

Accurate and efficient treatment of solid boundaries for the Vortex Particle-Mesh method

Doctoral dissertation presented by

Thomas GILLIS

in partial fulfillment of the requirements for
the degree of Doctor in Engineering Sciences

June 2019

Thesis committee

Pr. Philippe CHATELAIN (co-supervisor, UCLouvain)

Pr. Grégoire WINCKELMANS (co-supervisor, UCLouvain)

Pr. Thomas PARDOEN (chairperson, UCLouvain)

Dr. Yves MARICHAL (secretary, UCLouvain)

Pr. Jean-François REMACLE (UCLouvain)

Pr. Philippe PONCET (UPPA)

Pr. Wim VAN REES (MIT)

*“Research is to see what everybody has seen
and think what nobody has thought.”*

Albert Szent-Györgyi (1957)

Remerciements

Je désire entamer ces remerciements en exprimant ma profonde reconnaissance à mes deux promoteurs, Philippe Chatelain et Grégoire Winckelmans, de m'avoir soutenu et guidé tout au long de ce travail. Merci Grégoire pour ta rigueur et pour les heures passées à revoir mes publications, à discuter des méthodes vortex ou encore à me transmettre avec enthousiasme ton expérience en mécanique des fluides. Merci Philippe pour ton imagination et ta créativité débordante, ta vision à long terme dans mon travail et ton positivisme à toute épreuve. Merci aussi de m'avoir incité et aidé à partir vers de nouveaux projets à l'issue de ces quatre années.

J'aimerais également remercier mon comité d'encadrement, Yves Marichal et Jean-François Remacle, pour leur patience et leurs réponses à mes questions, ainsi que les membres de mon jury, Philippe Poncet, Wim van Rees et Thomas Pardoën, pour leur lecture attentive de mon manuscrit et la discussion constructive lors de la défense privée.

J'ai également eu la chance de recevoir un soutien infailible de la part de mes collègues, devenus amis. Merci Matthieu Duponcheel pour ta disponibilité constante, ta curiosité et ton intérêt dans ma recherche. Passer dans ton bureau m'a beaucoup aidé à me remettre en question et à prendre du recul. Merci Denis-Gabriel Caprace (DG) pour ton soutien, ton aide au jour le jour et ton envie de travailler ensemble. J'espère que le wiki et les makefiles que nous avons écrits pourront perdurer au-delà de nos thèses. Je n'oublierai pas non plus les vols au-dessus de Namur, l'ambiance dans notre bureau, les nuits passées dans ton canapé, etc. Merci aussi à Philippe Parmentier et Olivier Thiry pour les week-ends organisés, les soirées et les repas mais surtout pour le temps que vous avez passé à répondre à mes questions. Ce séjour au TFL n'aurait

pas été pareil sans les académiques, les jeunes furieux et toutes les activités qui ont ponctué ces quatre années: le vol de canapé, les dessins furtifs sur les tableaux, les barbecues, les banquets méca, les Saint Éloi, etc. Merci à vous d’avoir fait du TFL un endroit où coexistent la bonne humeur et la très haute qualité scientifique.

J’aimerais aussi remercier tous ceux qui ont contribué plus discrètement à mon travail. Tout d’abord Damien François et Thomas Keutgen qui ont eu la patience de recueillir toutes mes questions techniques et tous mes mails concernant les clusters. Merci aussi à Danielle Coulon et Greg Hernandez qui m’ont aidé à plonger dans le monde du HPC. Merci également à Isabelle Hennau pour son aide quotidienne dans les démarches administratives. De plus, j’aimerais remercier le F.R.S-FNRS et le CÉCI d’avoir cru dans mon projet et m’avoir permis de le réaliser à travers une bourse FRIA et la mise à disposition des ressources de calculs nécessaires.

Je n’aurais pu réaliser ce travail sans le soutien de mes colocos et mes amis. Merci de m’avoir encouragé et accompagné au jour le jour. Merci pour votre bonne humeur, pour votre intérêt pour mon travail en dépit de l’aspect technique de celui-ci et pour tous nos échanges bien éloignés des méthodes vortex. Merci également à ma famille pour sa présence et son envie de comprendre ma recherche. Merci particulièrement à mes parents de m’avoir offert l’opportunité de faire ce qui me tient à cœur.

Finalement, merci à vous qui vous hasardez à parcourir cet ouvrage.

Bonne lecture,

Thomas¹

¹ alias TG ou Gru, Fondateur du CFD

Contents

Introduction	1
1 Methodological context	9
1.1 Introduction	9
1.2 The Lattice Green's Function	9
1.3 The Hockney-Eastwood algorithm	12
1.3.1 Real space analysis	12
1.3.2 Fourier analysis	14
1.4 The Immersed Interface method	15
2 Iterative Penalization	23
2.1 Introduction	23
2.2 Methodology	25
2.2.1 The Brinkman penalization	25
2.2.2 Krylov-based iterative methods	30
2.2.3 Recycling methods	32
2.2.4 Recycling for the iterative penalization method	33
2.2.5 Convergence criterion for the iterative penalization method	33
2.2.6 Force computation	35
2.3 Validation and analysis	35
2.3.1 Flow past a cylinder at $Re = 550$	36
2.3.2 Computational cost	36
2.4 Results on impulsively started flows	40
2.4.1 Flow past a cylinder at $Re = 9500$	40

2.4.2	Flow past a thin plate at $Re = 1000$	41
2.5	Conclusions	47
3	Immersed Interface Poisson solver	49
3.1	Introduction	49
3.2	Methodology	51
3.2.1	A second order unbounded, body-free, 3D Poisson solver	51
3.2.2	A second order Immersed Interface method	52
3.2.3	The combined problem	52
3.3	Validation: potential flows past a body	54
3.3.1	Potential flow past a sphere	56
3.3.2	Potential flow past an ellipsoid	56
3.3.3	Sensitivity analysis: R_{min}/h	59
3.3.4	Sensitivity analysis of the boundary problem tolerance	61
3.4	Conclusions	62
4	IIVPM 2D	65
4.1	Introduction	65
4.2	Methodology	66
4.2.1	The 2D Vortex Particle-Mesh method	66
4.2.2	Time step sequence	67
4.2.3	Poisson equation	68
4.2.4	Vorticity sheet treatment	69
4.2.5	M2P and extension inside the body	71
4.2.6	P2M and ghost particles	73
4.3	Validation	73
4.3.1	Temporal convergence	75
4.3.2	Spatial convergence of the method	76
4.4	The impulsively started flow past a cylinder	78
4.4.1	Flow at $Re = 550$	79
4.4.2	Flow at $Re = 3000$	80
4.4.3	Flow at $Re = 9500$	82
4.4.4	Flow at $Re = 40000$	84
4.5	Conclusions	88
5	IIVPM 3D	89
5.1	Introduction	89
5.2	Methodology	90

5.2.1	Poisson solver	90
5.2.2	Wall vorticity	91
5.2.3	Diffusion and stretching	92
5.2.4	Particle-Mesh Interpolation	92
5.2.5	Reprojection	93
5.2.6	Force computation	94
5.2.7	Numerical evaluation of surface integrals	98
5.3	Validation of the wall quantities	101
5.3.1	Stokes flow past a sphere	102
5.3.2	Impulsively started flow past a sphere at $Re_D = 300$	106
5.4	Flow past a sphere at $Re_D = 300$	109
5.4.1	Drag and lift coefficients	110
5.4.2	Computational time and GMRes iterations	111
5.4.3	Comparison with the Iterative Penalization VPM method	113
5.5	Impulsively started flow past an ellipsoid at $Re_c = 300$	120
5.6	Conclusions	122
6	A Multiresolution Poisson solver	123
6.1	Introduction	123
6.2	Methodology	124
6.3	Validation: 2D unbounded Gaussian source-term	127
6.4	Conclusions	130
	Conclusions	133
	Bibliography	139
A	Analytical formulations	153
A.1	Signed Distance Function of a sphere	153
A.2	Signed distance function of an ellipsoid	153
A.3	Analytical solution of the potential flow past an ellipsoid	154
B	The Fourier transform	157
B.1	Definitions	157
B.2	Properties	159
C	Lattice Green's Function computation	161
C.1	Two dimensions Lattice Green's function	161
C.2	Three dimensions Lattice Green's function	162

D Immersed Interface stencils	165
D.1 Polynomial interpolation	165
D.2 Summary of the Immersed Interface stencil used in 2D	166
E Wavelet theory and multiresolution analysis	169
E.1 Multiresolution analysis	169
E.2 Wavelet Decomposition	170
E.3 Bi-orthogonal wavelets	171
E.4 Interpolating wavelets	172

Introduction

Among fluid dynamics phenomena, wakes stand out as crucial in many engineering and scientific domains: wind farms and energy production, air-traffic management, aircraft and helicopter design, bio-locomotion, are a few examples of devices and applications that produce and/or exploit wake phenomena. When looking at Nature, they are several examples of energy harvesting through wake exploitation. Birds flying in formation or fish swimming in schools are constantly moving and deforming their bodies to gain energy from the ambient fluid and from the wakes generated by others. In order to replicate such techniques and then to adapt them in our engineered devices, we have to understand, predict and control such phenomenon.

The wake is the flow region impacted by the interaction between an object and the ambient fluid. It carries the signature of the force exchange happening through very fine flow phenomena, at the interface between the fluid and the immersed object. A faithful prediction of the wake thus depends upon the accurate capture of the flow physics around the body; this is even more critical for the so-called Fluid-Structure Interaction (FSI) problems where the object is moving and deforming due to the forces exerted by the flow.

For an accurate study and prediction of wakes, the recourse to Computational Fluid Dynamics (CFD) is inescapable and it relies on rapidly growing resources. The computational power of supercomputers is constantly increasing and boosted by the use of accelerators, such as GPU's. As an example, the world top supercomputer, Summit (USA), is made of 2 397 824 cores and able to perform, theoretically, more than $200 \cdot 10^{16}$ floating point operations per second (flops)^{2,3}. Even if these resources may

²A floating point operation is the simplest operation between two numbers: +, -, *

³Data from the top 500 list issued in November 2018

seem enormous, they are still finite and it is difficult to design a simulation tool able to use them efficiently together and for a single purpose. In the case of Summit, the maximum performance achieved on this cluster (using the LINPACK test) is $143 \cdot 10^{16}$ flops, which is $\simeq 70\%$ of the peak performance. We would like to stress that in practice such peak performances are almost impossible to reach due to a communication overhead inescapable in realistic applications. These two limitations, the non-infinite computational power and the performance loss inherent to massively distributed processing, currently bounds the size and the complexity of the problems tackled in the computational fluid dynamics community. Alternatively, one may also want to make the most of smaller computational units. Therefore, there is a constant effort to increase the accuracy and the efficiency of the algorithms used, which in turn reduces the computational cost and pushes further the boundaries of the possible applications.

Context and equations

We here consider an incompressible flow, a typical assumption in wake studies and external aerodynamics. The incompressible flow dynamics are governed by the Navier-Stokes equations, given in the velocity-pressure formulation as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \frac{D\mathbf{u}}{Dt} = -\nabla P + \nu \nabla^2 \mathbf{u} , \quad (1)$$

where $\frac{D}{Dt}$ stands for the Lagrangian derivative, $\mathbf{u}$ denotes the velocity, $\nu = \mu/\rho$ is the kinematic viscosity and $P = \frac{p}{\rho}$ is the reduced pressure, defined from the pressure p . Equivalently, a vorticity-velocity formulation can be obtained by taking the curl of (1) as

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \nabla) \boldsymbol{\omega} = \frac{D\boldsymbol{\omega}}{Dt} = \nu \nabla^2 \boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla) \mathbf{u} , \quad (2)$$

where and $\boldsymbol{\omega} = \nabla \wedge \mathbf{u}$ denotes the vorticity. The velocity is related to the vorticity through its Helmholtz decomposition, based on the stream and potential functions, Ψ and Φ respectively,

$$\mathbf{u} = \mathbf{U}_\infty + \nabla \wedge \Psi + \nabla \Phi . \quad (3)$$

where

$$\nabla^2 \Psi = -\boldsymbol{\omega} \quad \nabla^2 \Phi = \nabla \cdot \mathbf{u} . \quad (4)$$

This last condition is null due to the incompressibility of the flow. In fluid mechanics, any flow configuration is characterized by its Reynolds number, defined as the ratio between the inertial and the viscous forces,

$$\text{Re} = \frac{U_\infty D}{\nu} , \quad (5)$$

where D is the characteristic size of the flow. If the Reynolds number is small, $\text{Re} < 1$, the flow is dominated by the diffusion and hence the advection can be neglected. If the Reynolds number is moderate, $1 < \text{Re} < 10^3$, the flow is laminar, i.e. driven by the advection and stabilized by the diffusion. For larger Reynolds numbers, $\text{Re} > 10^3$, the flow becomes turbulent as the advection phenomenon is dominant at the larger scales. For such high Reynolds number flows, the Direct Numerical Simulation (DNS) approach captures both the larger and the smaller scales. The computational resources of this approach have been shown to scale⁴ as Re^3 . To reduce this tremendous computational cost, one has to discard the smallest scales and model their influence on the larger ones. This is the so-called the "Large Eddy Simulation" (LES) approach. Another alternative, the Reynolds Averaged Navier-Stokes (RANS), considers the averaged flow behavior and hence reduces the computational cost. In this work, we consider moderate Reynolds number; the DNS approach is thus adequate.

The Vortex Particle-Mesh method

Vortex methods discretize the vorticity field by means of particles, characterized by a position $\mathbf{x}_p$, a volume V_p and a strength $\alpha_p = \int_{V_p} \boldsymbol{\omega} dV$. One can integrate Eq. (2) over a particle volume through a mid-point quadrature with the assumption of the particle discretization and then finds a set of ordinary differential equations for the particle position and strength (see [2, 3] for a detailed analysis of particle discretization)

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}(\mathbf{x}_p) \quad (6a)$$

$$\frac{d\alpha_p}{dt} = \int_{V_p} [(\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega}] d\mathbf{x} \quad (6b)$$

$$\simeq [(\boldsymbol{\omega} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega}]_p V_p, \quad (6c)$$

where we identify the role of the velocity field in the advection of the particles, and the role of the stretching and diffusion terms for the evolution of vorticity.

Among vortex methods, the purely particle-based approaches, such as the Vortex Particles (VP) method or Smoothed Particle Hydrodynamics (SPH) [4], promise a straightforward implementation of Eqs. (6) and robust adaptivity. A closer inspection will show that the latter comes at the price of accuracy [5, 6], as particles deplete regions of the flow or cluster in others. Several remedies have been proposed at the price of artificial bulk and shear viscosities [7].

⁴As detailed by Thiry [1, p.5], the total simulation cost, $N_x N_y N_z N_t$, scales as $(L/\eta)^4 \propto \text{Re}^3$.

Another way to prevent particles depletion and clustering, used by many [3, 8–10] in the context of the vortex particle method, is the use of a procedure called remeshing or redistribution, which consists in the periodic regularization of the particle set onto a mesh using high order interpolation formulas [11–13]. One predicts that such a technique will introduce dissipation and dispersion; the interpolation kernel used in this work for instance, M'_4 [11, 12], adds to leading order a hyper-viscous type term, which was quantified to behave as

$$\frac{h^4}{N_{\text{redistr}} \Delta t} \nabla^4 \boldsymbol{\omega} , \quad (7)$$

where h is the mesh spacing, Δt the time step and N_{redistr} the redistribution period in number of time steps [3]. This behavior was more formally quantified by Marichal [14], both in terms of dispersion and dissipation. These results indicate that a particle method with a too frequent redistribution performs essentially like a Eulerian method; if one wants to preserve its intrinsic qualities, redistribution has to be used as seldom as possible (through monitoring of the deformation in the particle arrangement for instance). Within this work, this frequency was chosen between 5 and 10 time-steps.

The remeshing procedure, coupled with a combination of the Lagrangian and the Eulerian formulations, is at the heart of the most recent vortex method variant, called the Vortex Particle-Mesh (VPM) or Vortex in Cell (VIC) approach. In this class of methods, the presence of a Cartesian mesh is also exploited to improve the computational efficiency, by the use of finite differences to evaluate the right hand side of Eq. (6b), i.e. to evaluate the velocity, the diffusion and the stretching term. The Poisson problem of Eq. (4), typically an unbounded problem, can also be solved on the mesh using a Fast Fourier Transform (FFT) based approach [15–17] or from a source/target particles point of view using a Fast Multipole Method (FMM) [18–23] or using a combination of both approaches [24]. The right hand side is then interpolated back to the particles location and particles evolve according to Eqs. (6).

There are several advantages that come with the use of a vortex method, compared to other classical formulations (finite-volume, finite elements formulations, for example). Among others, we note the low numerical errors, the relaxed stability constraint for advection dominated flows and the use of a structured grid for the Poisson equation. As a result, vortex methods have gained popularity through several fundamental and/or applied contributions. As a non-exhaustive list, let us mention the lifting line theory applied to the VPM framework [25], the coupling of the VPM framework with wall dedicated approaches [14, 24, 26–29], the study of bluff body flows [30–35], the massively parallel architecture implementation of the vortex methods [36] and its ap-

plications to the wake dynamics [37–39], to the wind energy production [40, 41], to the energy harvesting in biolocomotion [42–44], to biological flows [45], etc.

Immersed boundaries

Taking an immersed obstacle into account in the vorticity formulation of the Navier-Stokes equations, Eq. (2), is not straightforward as the wall value is not readily accessible. Indeed, it corresponds to the gradient of the velocity at the wall which is driven by the body motion and the flow behavior in the very close vicinity of the interface. Considering the case of an interface matching with the Cartesian grid points, Thom and Ingram [46] proposed a first order boundary condition for the vorticity, relying on the stream function. Over the years, many variants of the Thom’s formula have been proposed and among them two are particularly famous: the Briley’s formula [47], a third order generalization of the Thom’s approach based on the stream function; and the Wood’s formula [48], which reaches a second order accuracy by considering the vorticity field in addition to the stream function. We refer the reader to Weinan and Liu [49], Napolitano et al. [50] and references therein for further details.

The use of a Cartesian mesh to take arbitrary solid obstacles into account increases the complexity of the immersed boundary treatment, as the obstacle will intersect the grid in arbitrary locations. It still constitutes an open challenge in the vorticity-based methods, and especially in the VPM framework which relies on such a grid. Many techniques have been proposed in the structured mesh framework and, following Mittal and Iaccarino [51], they are usually classified into two categories: the “continuous forcing” and the “discrete forcing” approaches.

Continuous forcing approach

The continuous forcing approach relies on the addition of a mathematical artifact inside the Navier-Stokes equations to impose the boundary condition. This artificial source term, $\mathbf{f}_b$, represents the force on the fluid exerted by the body. As an example, in the vorticity formulation, the momentum equation Eq. (2) is modified as

$$\frac{D\boldsymbol{\omega}}{Dt} = \nu \nabla^2 \boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla) \mathbf{u} + \nabla \wedge \mathbf{f}_b . \quad (8)$$

There are two well-known formulations for this additional source term: the Immersed Boundary method and the penalization approach.

The first and widely used method in this category is the Immersed Boundary (IB) approach, initially proposed in the pioneering work of Peskin [52] and renowned for

its proven robustness [53, 54]. The IB method discretizes the singular forcing term due to the immersed body surface, $\mathbf{f}_b$, on several grid points using regularized Dirac functions, hence mollifying the inner interface. The second approach in this class of methods is the Brinkman penalization, which uses a Lagrangian relaxation of the body constraint [55, 56] to impose the inner boundary condition. This type of method has been transposed to the Vortex Particle-Mesh method and is widely used throughout the works of Mimeau et al. [32], Verma et al. [44], Coquerelle and Cottet [57], Gazzola et al. [58], van Rees [59]. Still in the VPM framework, some more recent works have proposed to iterate on the penalization constraint to improve its enforcement inside the body, see Hejlesen et al. [60], Gillis et al. [61]. However, this approach is only first order at the boundary and, in the case of the iterative penalization, implies multiple resolutions of the Poisson equation in the computational domain and inside the inner body.

Discrete forcing approach

The second class of methods, the discrete forcing approach, rather modifies the discretization of the Navier-Stokes equations in order to take an immersed body into account. Hence, no additional source term is added to the Navier-Stokes equation but the changes made to the discretization usually increase the implementation complexity.

In this category, let us first mention the Ghost-Cell Method, that uses ghost cells to extend the fields inside the body by means of a polynomial extrapolation, see [51] and references therein. We also refer to the Ghost-Fluid approach, which uses the combination of the Neumann and the Dirichlet boundary conditions to build a field extension inside the body, see [62] and references therein.

Still in the discrete forcing approach, the Boundary Element Method (BEM), also called panel method, has been coupled with the Vortex Particle framework, see e.g. [10, 63]. In this approach, the convection and diffusion of the vorticity field are first handled discarding the generation of vorticity at the wall. Then, following Lighthill [64], the vortex sheet associated with the residual slip velocity is diffused into the flow, this is the so-called “viscous splitting” approach.

Let us also mention the work of Cottet and Poncet [26] that proposes a similar diffusion splitting method, applied to body-fitted grids and immersed boundary technique. Finally, we mention the Immersed Interface method, introduced by Leveque and Li [65] and Wiegmann and Bube [66], which has been widely applied to grid-based approaches, see Linnick and Fasel [67], Mayo and Greenbaum [68], Hosseinverdi and Fasel [69], for example.

The improvement of the treatment of immersed boundaries in the Vortex Particle-Mesh methods is precisely the central goal envisioned of this work. This challenge has been first tackled in two dimensions in the work of Marichal [14]. Indeed, the author presented a first version of the Immersed Interface Vortex Particle-Mesh method, suffering from two main drawbacks. First, the Poisson solver was costly and difficult to extend to three dimensions. Second, the splitting approach of the diffusion operation limited the time convergence to $\mathcal{O}(\Delta t)$.

These two drawbacks constitute the starting point of our work, which aims to propose a fully scalable and efficient Immersed Interface treatment for the Vortex Particle-Mesh method.

Outline

In Chapter 1, we first present the different algorithms and methods used throughout this work: the Hockney-Eastwood algorithm, the Lattice Green's Function and the Immersed Interface method. This chapter is meant to propose a concise, yet complete description of the main computational techniques in order to help the reader when going through this document.

The first part of this work, in Chapter 2, focuses on the iterative penalization method, a non sharp, yet non-intrusive way to take an inner obstacle into account. This approach suffers from the requirement of solving a large number of Poisson equations in the iterative procedure. To reduce the computational cost and preserve the quality of the solution, we propose to recycle previously computed solutions through the use of a deflated BiCGStab linear solver. We demonstrate the computational gain and assess its performance on well known benchmarks such as the impulsively started flows past a cylinder and past a plate.

In the next chapter, Chapter 3, we present an efficient Poisson solver in the Immersed Interface framework. Our approach, based on the Sherman-Morrison-Woodbury formula, decomposes the problem into two sub-problems. The first one is a bulk Poisson equation, that can be solved using a standard Poisson solver (multigrid, FMM, FFT). The second problem is a very localized and smaller compared to the first one. This second problem is then solved only in the vicinity of the immersed surface based on the recycling approach proposed in Chapter 2.

In Chapter 4, we develop the two dimensional version of the Immersed Interface approach applied to the Vortex Particle-Mesh method. This proof-of-concept is a preparatory work to the three dimensional work. In this chapter, we propose an approach which relies on the computation of the wall vorticity using a boundary condition

similar to the Woods boundary condition. Hence we eliminate the splitting problem and retrieve the expected order in time convergence. We also generalize the Hamilton-Jacobi extension to the Immersed Interface framework, which is used to compute ghost particles required by the particle-mesh interpolation procedure.

This work in 2D lays the foundations for the three dimensional Immersed Interface method proposed in Chapter 5. It generalizes the work done in Chapter 4 and extends it to handle new challenges, such as the stretching term computation and the solenoidal reprojection technique. In this chapter, we also analyze and discuss the performance of the proposed approach, compared to the iterative penalization approach proposed in Chapter 2, using the benchmark of the flow past a sphere at $\text{Re}_D = 300$. We also detail the way to evaluate surface integrals around the surface and assess its convergence on the Stokes flow past a sphere.

Finally, in Chapter 6, we present a preparatory study of a two dimensional Poisson solver in a multiresolution framework, using a wavelet approach. This solver is designed for the Immersed Interface application and makes an extensive use of the level to level interpolation provided by the wavelet framework. We finally demonstrate the convergence of the proposed approach.

Methodological context

1.1 Introduction

The aim of this chapter is to summarize the methods and algorithms used throughout this work, the unbounded Poisson solvers, the lattice Green's function and the Immersed Interface method information, in order to facilitate the reading of this document. The applications of these methods can be found in the following chapters.

First, we present the Hockney-Eastwood algorithm in §1.3 and its Fourier analysis. In §1.2, we present the lattice Green's function, initially introduced in 2D by Martinsson [70] and Gillman and Martinsson [22]. Finally, we introduce the Immersed Interface method, originally proposed by Leveque and Li [65] and further improved by Linnick and Fasel [67] and Li and Kazufumi [71].

1.2 The Lattice Green's Function

In this section, we present the Lattice Green's Function, introduced by Martinsson [70]. We first focus our approach on the 2D case, as proposed by Gillman and Martinsson [22] and then we extend the development to 3D, as done by Liska and Colonius [72]. Let us first consider a Poisson equation and its associated Green's function, G , defined as

$$\nabla^2 G = \delta \quad . \quad (1.1)$$

The straightforward inversion of the ∇^2 operator is challenging and computationally intensive [73–76]. In its analytical form, the Green's function has a singularity at $\mathbf{x} = 0$, which requires an infinite series of wavenumbers. When considering a discretized domain with a grid spacing given by h , the maximum wavenumber representable on

the grid is $k = \pi/h$. The way to treat this issue and therefore to regularize the problem is crucial to obtain the desired convergence behavior.

A first solution is to regularize the Green's function analytically, typically using a Gaussian filter [16, 17, 23]. The number of moments conserved by the filter determines the degree of convergence of the global Poisson solver. This approach is widely used and studied throughout the literature in other contexts such as the vortex particle method [77] or the regularized Stokeslets method [78]. Here, we use another technique which considers the discretized version of Eq. (1.1) and therefore eliminates the singularity issue,

$$\nabla_h^2 G_h = \delta_h \quad , \quad (1.2)$$

where ∇_h^2 stands for the Laplacian operator expressed using a finite difference stencil.

Without loss of generality, we will consider the Green's function associated with the standard second order stencil.

2D Lattice Green's Function

In 2D, as proposed by Gillman and Martinsson [22], the Poisson equation Eq. (1.2) expressed in wave space, $\mathbf{k} = (k_x ; k_y) \in [-\pi/h ; \pi/h]^2$, becomes

$$\frac{1}{h^2} [4 \sin^2(k_x h/2) + 4 \sin^2(k_y h/2)] \hat{G}_h \triangleq \hat{\sigma}(\mathbf{k}h) \hat{G}_h = 1 \quad . \quad (1.3)$$

The associated Lattice Green's Function (LGF) is given by the unbounded inverse Fourier transform¹ of $1/(\hat{\sigma}(\mathbf{k}h))$ [22], see Eq. (B.6b),

$$G_h(\mathbf{x}_m) \triangleq \frac{1}{(2\pi)^2} \iint_{-\pi/h}^{\pi/h} \frac{h^2 \exp(i \mathbf{x}_m \cdot \mathbf{k})}{4 \sin^2(k_x h/2) + 4 \sin^2(k_y h/2)} d\mathbf{k} \quad . \quad (1.4)$$

The LGF can also be expressed, with respect to $\mathbf{m} = \frac{\mathbf{x}}{h} = \{m_x ; m_y\}$ only, as

$$G_h(\mathbf{x}_m) = \frac{1}{(2\pi)^2} \iint_{-\pi}^{\pi} \frac{\exp(i \mathbf{m} \cdot \boldsymbol{\xi})}{4 \sin^2(\xi_x/2) + 4 \sin^2(\xi_y/2)} d\boldsymbol{\xi} \quad , \quad (1.5)$$

with $\boldsymbol{\xi} = \mathbf{k}h$. For small $|\mathbf{m}|$, we evaluate this integral numerically, while for $|\mathbf{m}|$ sufficiently large (in practice, $|\mathbf{m}| \geq 32$), Eq. (1.5) is replaced by the following expansion

¹We here note that the convention used by [22, 79, 80] defines the inverse Fourier transform using $\exp(-i \mathbf{x}_m \cdot \mathbf{k})$. Since the Green function is real, the two conventions are equivalent.

[79] (given with more terms in Appendix C, Eq. (C.7))

$$G_h(\mathbf{x}_m) = \frac{1}{2\pi} \left[\log(\mathbf{m}) + \gamma + \frac{\log(8)}{2} \right] - \frac{1}{24\pi |\mathbf{m}|^6} \left[m_x^4 + m_y^4 - 6m_x^2 m_y^2 \right] + \dots + \mathcal{O}(1/|\mathbf{m}|^{10}) \quad (1.6)$$

3D Lattice Green's Function

The 3D case is a generalization of the 2D development. As proposed by Liska and Colonius [72], the body-free Poisson equation, Eq. (1.2), expressed in wave space, $\mathbf{k} = (k_x; k_y; k_z) \in [-\pi/h; \pi/h]^3$, becomes

$$\frac{1}{h^2} \left[4 \sin^2 \left(\frac{k_x h}{2} \right) + 4 \sin^2 \left(\frac{k_y h}{2} \right) + 4 \sin^2 \left(\frac{k_z h}{2} \right) \right] \hat{G}_h \triangleq \hat{\sigma}(\mathbf{k}h) \hat{G}_h = 1, \quad (1.7)$$

The associated Lattice Green's Function is given by the unbounded inverse Fourier transform of $1/(\hat{\sigma}(\mathbf{k}h))$, see Eq. (B.6b):

$$G_h(\mathbf{x}_m) = \frac{1}{(2\pi)^3} \iiint_{-\pi/h}^{\pi/h} \frac{h^2 \exp(i \mathbf{x}_m \cdot \mathbf{k})}{4 \sin^2(k_x h/2) + 4 \sin^2(k_y h/2) + 4 \sin^2(k_z h/2)} d\mathbf{k}. \quad (1.8)$$

The LGF can also be expressed, with respect to $\mathbf{m} = \frac{\mathbf{x}}{h} = \{m_x; m_y; m_z\}$ only, as

$$G_h(\mathbf{x}_m) = \frac{1}{(2\pi)^3 h} \iiint_{-\pi}^{\pi} \frac{\exp(i \mathbf{m} \cdot \boldsymbol{\xi})}{4 \sin^2(\xi_x/2) + 4 \sin^2(\xi_y/2) + 4 \sin^2(\xi_z/2)} d\boldsymbol{\xi}, \quad (1.9)$$

with $\boldsymbol{\xi} = \mathbf{k}h$. For small $|\mathbf{m}|$, we evaluate this integral, while for $|\mathbf{m}|$ sufficiently large, Eq. (1.9) is replaced by the following expansion (given with more terms in Appendix C, Eq. (C.10)),

$$G_h(\mathbf{x}_m) h = -\frac{1}{4\pi |\mathbf{m}|} - \frac{\left[m_x^4 + m_y^4 + m_z^4 - 3m_x^2 m_y^2 - 3m_x^2 m_z^2 - 3m_y^2 m_z^2 \right]}{16\pi |\mathbf{m}|^7} + \dots + \mathcal{O}(1/|\mathbf{m}|^9) \quad (1.10)$$

Using this LGF, we express the application of the inverse of ∇_h^2 as an unbounded convolution,

$$u = [\nabla_h^{-2} f]_{ij} = \sum_{kl} G_h(\mathbf{x}_{kl} - \mathbf{x}_{ij}) f(\mathbf{x}_{kl}), \quad (1.11)$$

which is computed using the Hockney-Eastwood algorithm (see §1.3) or the KiFMM algorithm as in Gillman and Martinsson [22].

1.3 The Hockney-Eastwood algorithm

In this section, we present the Hockney and Eastwood algorithm, first from a schematic point of view, then from the Fourier point of view. A short reminder of the Fourier transforms and of some helpful properties is given in Appendix B.

1.3.1 Real space analysis

The goal of the Hockney-Eastwood approach is to elude the periodicity imposed by the Fourier transform in order to compute the unbounded convolution between a kernel, G , and a source term with a compact support fully included inside the computational domain. We then aim at replacing the unbounded convolution by a bounded sum over the domain. In order to do that, we will consider a doubled domain where the Green's function is made periodic, G_e , and the source term is extended using a zero-padding f_e , see Figure 1.1. Then, one performs a periodic convolution on the doubled domain, $[0 ; 2L]$. This convolution is accelerated using the Fourier transform on the doubled domain. The result of this convolution is exactly the result of the unbounded convolution in the domain $[0 ; L]$ and has no meaning in the domain extension, from $[L ; 2L]$. Therefore, we then discard the solution in the domain extension and only consider the original computational domain.

The total cost of the method is then $\mathcal{O}(2N \log(2N))$, neglecting the memory overhead and the communication between processors if any. In order to reduce the computational cost, two improvements can be applied to the presented algorithm. Although we have not used them in this work, we present them for the sake of completeness. First, one could use a DCT instead of a domain doubling to compute the Green's function. Considering the convention used in the FFTW library, a DCT of type 1 could be used in the example in Figure 1.1. Second, the zero padding of the sources and the associated Fourier transform on the extended domain may be avoided. Indeed, the zero padding of the source is an interpolation in the Fourier space. The zero-padded DFT is given by, see Eq. (B.5a),

$$\hat{f}(k) = \frac{1}{2N} \sum_{j=0}^{N-1} f(x_j) \exp\left(-i k \frac{2\pi}{2L} x_j\right), \quad (1.12)$$

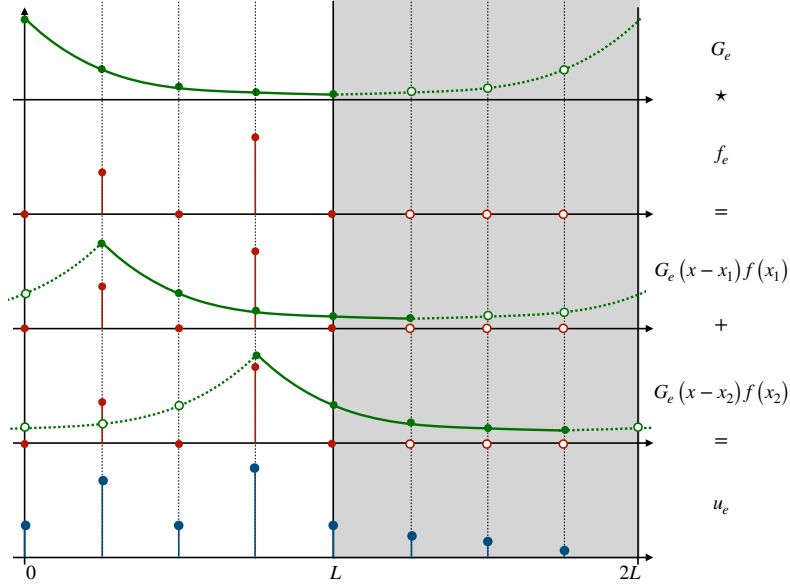


Figure 1.1: The Hockney-Eastwood algorithm illustrated.

where we distinguish two cases, whether k is even or odd, respectively,

$$\hat{f}(k_{\text{even}}) = \frac{1}{2N} \sum_{j=0}^{N-1} \left[f(x_j) \exp\left(-i \frac{k}{2} \frac{2\pi}{L} x_j\right) \right] \quad (1.13a)$$

$$\hat{f}(k_{\text{odd}}) = \frac{1}{2N} \sum_{j=0}^{N-1} \left[f(x_j) \exp\left(-i \frac{2\pi}{2L} x_j\right) \exp\left(-i \frac{(k-1)}{2} \frac{2\pi}{L} x_j\right) \right], \quad (1.13b)$$

When k is even, the Fourier transform can be computed using a standard real to complex DFT on a domain $[0; L]$. When k is odd, we first need to evaluate

$$f(x_j) \exp\left(-i \frac{2\pi}{2L} x_j\right) \quad (1.14)$$

before computing the complex to complex DFT. In practice, the cost of this double DFT is $\mathcal{O}(2N \log(N))$ instead of $\mathcal{O}(2N \log(2N))$, which represents a cost saving of $\mathcal{O}(2N \log(2))$. In terms of memory saving, there is no advantage since we still need to keep $2N$ values.

1.3.2 Fourier analysis

Given the unbounded discrete Green's function, G , and its Fourier Transform, $\hat{G}$, the Hockney-Eastwood algorithm is decomposed as follows:

1. It windows the Green's function G on the doubled computational domain (from $-L$ to L). This operation implies a convolution with the Fourier transform of the window function, $\hat{W}$,

$$G_w(x_n) = G(x_n) * W(x_n) \Leftrightarrow \hat{G}_w(k) = \hat{G}(k) \star \hat{W}(k) , \quad (1.15)$$

with $n = \{-\infty ; \dots ; -1 ; 0 ; 1 ; \dots ; +\infty\}$ and $k = [-\pi/h ; \pi/h]$.

2. It periodizes the function G_w , on the doubled computational domain, which is equivalent to sample the Fourier function, $\hat{G}_w$, with a frequency of $k_s = \frac{2\pi}{2L} = \frac{\pi}{L}$:

$$\hat{G}_w(nk_s) = \hat{G}_w(k) \delta(k - nk_s) . \quad (1.16)$$

The Green's function actually used in the Hockney-Eastwood algorithm is then given by

$$\hat{G}_w(nk_s) = [\hat{G}(k) \star \hat{W}(k)] \delta(k - nk_s) . \quad (1.17)$$

Let us stress that the sampling frequency, $k_s = \frac{\pi}{L}$, is half the sampling frequency we would have obtained by taking a standard DFT on the computational domain (0 to L), $\frac{2\pi}{L}$. Therefore, while computing the source term using Fourier series, it has to be interpolated at that half frequency. This Fourier space interpolation is performed by a zero-padding of the sources to the real domain.

As indicated by Eq. (1.17), the Fourier spectrum of the equivalent Green's function, obtained through the Hockney-Eastwood algorithm, is not equivalent to a sampling of the analytical Fourier representation of the unbounded Green's function. As a counter-example, let us apply the Hockney-Eastwood algorithm to the second order regularization function, proposed by Hejlesen et al. [17],

$$G_\sigma = \frac{1}{2\pi} \left[\log(r) + \frac{1}{2} E_1 \left(\frac{\rho^2}{2} \right) \right] , \quad (1.18)$$

where $\rho = \frac{r}{\sigma}$, with $\sigma = 2h$. First, we assume that the real form is known and given by Eq. (1.18). The Hockney-Eastwood algorithm windows the Green's function and doubles its domain, as shown in Figure 1.2. If the Fourier transform of the kernel is sampled without any windowing operation, it results in an "aliasing" phenomenon in

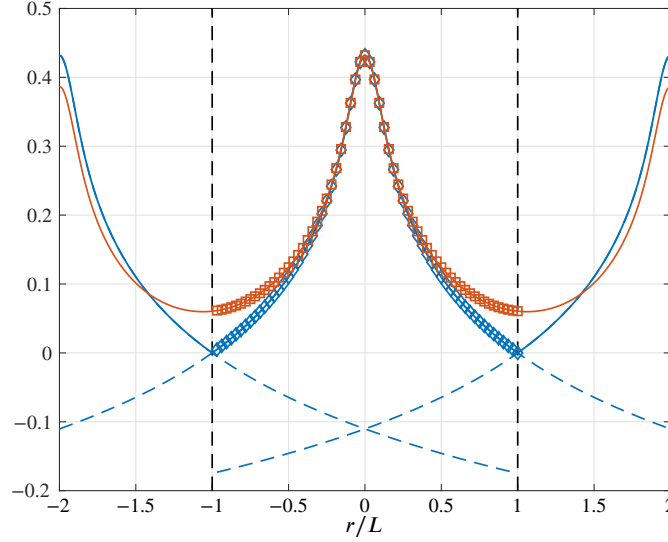


Figure 1.2: **Regularized Green function for the Hockney-Eastwood algorithm** Periodized Green's function from the sampling theorem applied in the Fourier domain (—); its evaluation with windowing, as performed by the Hockney-Eastwood algorithm (◇); equivalent Green's function obtained when sampling the Fourier transform and discarding the windowing (—) and its Hockney-Eastwood evaluation (□).

the real space if the sampling frequency is not small enough, illustrated in Figure 1.2. In this example, we have used a highly inadequate sampling frequency to highlight the different phenomena at play.

1.4 The Immersed Interface method

We here provide a brief presentation of the Immersed Interface Method (IIM), as applied in this work. For a more detailed information, we refer to [65, 67, 71, 76, 81, 82] and the references therein. We start with a general presentation of the method in 1D and then detail its application in 2D and 3D.

General considerations

The IIM relies on the handling of discontinuities in finite difference stencils. The discontinuities are incorporated into the stencil using a jump-corrected Taylor series. As an example, let us consider a one-dimensional function $u(x) = u^{(0)}(x) \in$

$C^\infty(\mathbb{R} \setminus \{x_\alpha\})$ and describe the IIM for the evaluation of the second derivative (required for the Laplacian evaluation in two and three dimensions), with second order accuracy. As shown in Figure 1.3, the function $u(x)$ and its derivatives, $u^{(\cdot)}(x)$, may

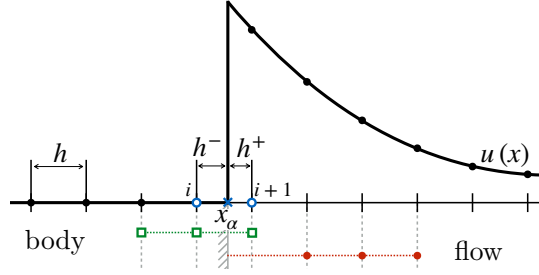


Figure 1.3: **1D Immersed interface scheme** The standard 3-point stencil (■) intersects the interface and generates a control point (✕), discontinuity affected points (○) and primary interpolation points (●).

be discontinuous at the interface, $x_\alpha \in]x_i ; x_{i+1}]$. Using a generalized Taylor series, we write a modified finite difference scheme at x_i and x_{i+1} for the second derivative, $u^{(2)}$, still valid across the interface (x_α),

$$u^{(2)}(x_i) = \frac{1}{h^2} [u(x_{i-1}) - 2u(x_i) + (u(x_{i+1}) - J_\alpha^+)] + \mathcal{O}(h^2) \quad (1.19a)$$

$$u^{(2)}(x_{i+1}) = \frac{1}{h^2} [(u(x_i) + J_\alpha^-) - 2u(x_{i+1}) + u(x_{i+2})] + \mathcal{O}(h^2) \quad (1.19b)$$

where we define the jumps as

$$J_\alpha^+ \triangleq [u^{(0)}]_\alpha + \frac{h^+}{1!} [u^{(1)}]_\alpha + \frac{(h^+)^2}{2!} [u^{(2)}]_\alpha + \frac{(h^+)^3}{3!} [u^{(3)}]_\alpha + \mathcal{O}(h^4) \quad (1.20a)$$

$$J_\alpha^- \triangleq [u^{(0)}]_\alpha + \frac{h^-}{1!} [u^{(1)}]_\alpha + \frac{(h^-)^2}{2!} [u^{(2)}]_\alpha + \frac{(h^-)^3}{3!} [u^{(3)}]_\alpha + \mathcal{O}(h^4) \quad (1.20b)$$

with $h^+ = (x_{i+1} - x_\alpha)$ positive, $h^- = (x_i - x_\alpha)$ negative and the jumps in the k th derivative:

$$[u^{(k)}]_\alpha \triangleq u^{(k)}(x_{\alpha+}) - u^{(k)}(x_{\alpha-}) \quad (1.21)$$

Without loss of generality, we assume that $x < x_\alpha$ represents the immersed obstacle, Ω_b . Inside Ω_b , all derivatives are supposed to be known since we know the desired solution inside the obstacle, for instance a null field as in Figure 1.3. Outside Ω_b , the evaluation of the value and the derivatives at the wall, required by Eq. (1.21), uses one-sided interpolation schemes and eventually the boundary condition (in this example, a

Dirichlet boundary condition: $u_\alpha = u(x_\alpha^+)$,

$$u^{(k)}(x_{\alpha^+}) = S_\alpha^k u_\alpha + S_1^k u(x_{i+2}) + S_2^k u(x_{i+3}) + S_3^k u(x_{i+4}) + \mathcal{O}(h^{4-k}) \quad , \quad (1.22)$$

which is in agreement with the global accuracy needed by the corrected scheme and the second derivative evaluation. The intersection between the mesh and the body surface is called “intersection point” (or “control point”), the points x_i and x_{i+1} are called “discontinuity affected points” (or “biased points”) and the mesh-points needed by Eq. (1.22) are called “primary interpolation points”.

Other variants of the presented methodology are also used in this work when we evaluate first degree operators, $(\nabla \cdot)$ or $(\nabla \wedge)$ for example, or when we do not have any boundary condition to impose. To evaluate first degree operators with second order accuracy, the number of derivatives in the jumps evaluation, Eq. (1.20), can be reduced,

$$J_\alpha^\pm = [u^{(0)}]_\alpha + \frac{h^\pm}{1!} [u^{(1)}]_\alpha + \frac{(h^\pm)^2}{2!} [u^{(2)}]_\alpha + \mathcal{O}(h^3) \quad , \quad (1.23)$$

and only two primary interpolation points are needed for the evaluation of the wall derivatives:

$$u^{(k)}(x_{\alpha^+}) = \tilde{S}_\alpha^k u_\alpha + \tilde{S}_1^k u(x_{i+2}) + \tilde{S}_2^k u(x_{i+3}) + \mathcal{O}(h^{3-k}) \quad , \quad (1.24)$$

for $k = 1, 2$, with $\tilde{S}$, the interpolation coefficients satisfying the needed accuracy. Following the same approach, when no boundary condition is provided, we rather extrapolate the value of the field up to the boundary, leading to another evaluation of the wall derivatives,

$$u^{(k)}(x_{\alpha^+}) = \tilde{T}_1^k u(x_{i+2}) + \tilde{T}_2^k u(x_{i+3}) + \tilde{T}_3^k u(x_{i+4}) + \mathcal{O}(h^{3-k}) \quad , \quad (1.25)$$

with $\tilde{T}$, the extrapolation coefficient satisfying the needed accuracy.

The Immersed Interface tool is applicable to a wide variety of boundary conditions. While we only presented three of them, one can quickly realize that a large number of combinations are available.

Generalization to higher dimensions

When considering higher dimensions, we use the dimension splitting approach, i.e. we apply, dimension by dimension, a 1D jump correction to standard 2D stencils, as proposed by Linnick and Fasel [67] and Marichal et al. [76].

Let us first consider a 2D intersection between the body and the mesh, and define the current local direction, ξ , as the direction of the grid line intersecting with the body

and the transverse local direction, η , see Figure 1.4. Following these definitions and assuming a Dirichlet boundary condition, the derivatives needed in the jump evaluation, Eqs. (1.20), are then expressed, using the notation of Figure 1.4 with $\xi_0 < \xi_\alpha \leq \xi_1$, as

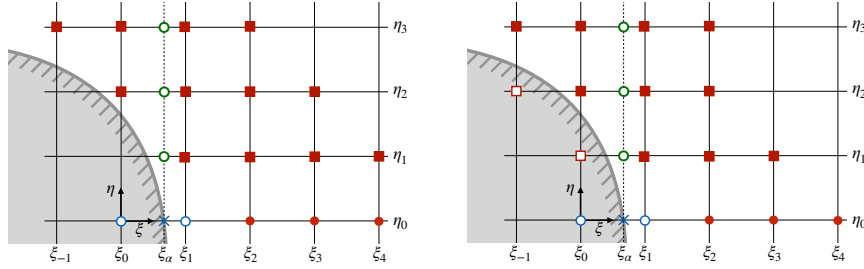
$$\frac{\partial^k u}{\partial \xi^k} \Big|_\alpha = S_\alpha^{\xi,k} u_\alpha + \sum_{i=1}^3 S_i^{\xi,k} u(\xi_{i+1}, \eta_0) + \mathcal{O}(h^{4-k}) \quad (1.26a)$$

$$\frac{\partial^k u}{\partial \eta^k} \Big|_\alpha = S_\alpha^{\eta,k} u_\alpha + \sum_{j=1}^3 S_j^{\eta,k} u(\xi_\alpha, \eta_j) + \mathcal{O}(h^{4-k}) \quad (1.26b)$$

for $k = 0, \dots, 3$ and the missing values in the η direction are interpolated from the neighbors,

$$u(\xi_\alpha, \eta_j) = \sum_{i=1}^4 R_i^0 u(\xi_{i-2+s_{\eta_j}}, \eta_j) + \mathcal{O}(h^4) \quad , \quad (1.27)$$

where s_{η_j} is a shift variable, computed for each level (see below). Following the notation in 1D, the grid points involved in Eqs. (1.26) are called “primary interpolation points”, while the points required in Eq. (1.27) are called “secondary interpolation points”. Finally, let us also note that Eq. (1.26a) is a generalization to higher dimensions of Eq. (1.22).



(a) Approach proposed by Marichal [14] and used in Chapter 3.

(b) Approach used in Chapters 4 and 5.

Figure 1.4: 2D Immersed Interface scheme - The mesh intersects the interface and generates one control point ($\times$), two discontinuity affected points (or biased points) ($\circ$) and primary interpolation points in the current direction, ξ , and in the transverse direction, η (represented as $\bullet$ and $\circ$, respectively). Primary interpolation points in the η direction are computed as a polynomial interpolated values using secondary interpolation points ($\blacksquare$ and $\square$).

The choice of the secondary interpolation points, i.e. the value of the shift coefficient s_{η_i} in Eq. (1.27), varies from one approach to another. Marichal [14] recommend to use the most centered scheme possible to reduce the interpolation error in Eq. (1.27).

We use this approach in Chapter 3 and discard any point inside the body and shift the interpolation stencil in the bulk when it crosses the boundary, see Figure 1.4 on the left side. In Chapters 4 and 5, we keep one point in the body when the secondary interpolation stencil crosses the inner boundary, in order to improve the consistency at the interface and reduce the number of the secondary interpolation points in the flow. The value of the inner point is computed by extrapolation in the ξ direction, using the jumps, Eq. (1.26a), at the interface, as it would have been computed when considering primary interpolation in the ξ direction at that level. Following the example in Figure 1.4, the point $u(\xi_0, \eta_1)$ is computed using Eqs. (1.20) and (1.26a), evaluated at level η_1 . With this choice, we aim at reducing the interpolation error and still consider the best possible information. Still following the example in Figure 1.4, the shift variables are then $s_{\eta_1} = 1$, $s_{\eta_2} = 0$ and $s_{\eta_3} = 0$.

Neumann boundary condition in higher dimensions

While the generalization of the Dirichlet boundary condition to higher dimensions is straightforward, see Eq. (1.27), imposing a 2D (or, equivalently a 3D) Neumann boundary condition increases the complexity. Indeed the dimensions are linked by the imposed boundary expression but are splitted in the IIM. To solve that problem in a coherent way, Marichal [14] proposed to use the so-called “compatible interpolation scheme”. We refer to Marichal [14] for the 2D scheme, as we here present its extension to 3D. First, we introduce a second direction to the local system of axes, defined at the intersection between the grid line and the body boundary, $\mathbf{x} = \mathbf{x}_\alpha$, as shown in 2D in Figure 1.4. We name the current direction, ξ and the two transverse directions, η_1 and η_2 . Following the 2D case, the values at grid points aligned with the current direction are given by $u_{\xi,p}$, while values aligned with the transverse directions, $u_{\eta_i,p}$, are not necessarily associated with grid points. The compatible interpolation scheme consists

of the following interpolation relations, formulated at the intersection,

$$u_\alpha = S_\alpha^{\xi,0} \frac{\partial u}{\partial \xi} \Big|_\alpha + \sum_{i=1}^3 S_i^{\xi,0} u(\xi_{i+1}, \eta_{1,0}, \eta_{2,0}) + \mathcal{O}(h^4) \quad (1.28a)$$

$$u_\alpha = S_\alpha^{\eta_1,0} \frac{\partial u}{\partial \eta_1} \Big|_\alpha + \sum_{j=1}^3 S_j^{\eta_1,0} u(\xi_\alpha, \eta_{1,j}, \eta_{2,0}) + \mathcal{O}(h^4) \quad (1.28b)$$

$$u_\alpha = S_\alpha^{\eta_2,0} \frac{\partial u}{\partial \eta_2} \Big|_\alpha + \sum_{j=1}^3 S_j^{\eta_2,0} u(\xi_\alpha, \eta_{1,0}, \eta_{2,j}) + \mathcal{O}(h^4) \quad (1.28c)$$

$$\frac{\partial u}{\partial n} = \mathbf{n}_\xi \frac{\partial u}{\partial \xi} + \mathbf{n}_{\eta_1} \frac{\partial u}{\partial \eta_1} + \mathbf{n}_{\eta_2} \frac{\partial u}{\partial \eta_2} \quad (1.28d)$$

where $S_\alpha^{\xi,0}$, $S_\alpha^{\eta_1,0}$ and $S_\alpha^{\eta_2,0}$ are the interpolation coefficients of Eq. (1.22) associated with the direction ξ , η_1 , η_2 , respectively. In order to make the notation more compact, we define the following abbreviations,

$$S^{\xi,0} \triangleq \sum_{i=1}^3 S_i^{\xi,0} u(\xi_{i+1}, \eta_{1,0}, \eta_{2,0}) \quad (1.29a)$$

$$S^{\eta_1,0} \triangleq \sum_{j=1}^3 S_j^{\eta_1,0} u(\xi_\alpha, \eta_{1,j}, \eta_{2,0}) \quad (1.29b)$$

$$S^{\eta_2,0} \triangleq \sum_{j=1}^3 S_j^{\eta_2,0} u(\xi_\alpha, \eta_{1,0}, \eta_{2,j}) \quad (1.29c)$$

Using this notation and solving the compatible interpolation system, formed by Eqs. (1.28), leads to an expression for $\frac{\partial u}{\partial \xi}$,

$$\begin{aligned} \frac{\partial u}{\partial \xi} \Big|_\alpha = \frac{1}{D} \left[S_\alpha^{\eta_1,0} S_\alpha^{\eta_2,0} \frac{\partial u}{\partial n} - \left(\mathbf{n}_{\eta_1} S_\alpha^{\eta_2,0} + \mathbf{n}_{\eta_2} S_\alpha^{\eta_1,0} \right) S^{\xi,0} \right. \\ \left. + \mathbf{n}_{\eta_1} S_\alpha^{\eta_2,0} S^{\eta_1,0} + \mathbf{n}_{\eta_2} S_\alpha^{\eta_1,0} S^{\eta_2,0} \right] \quad (1.30) \end{aligned}$$

where D is the determinant of the system,

$$D = \mathbf{n}_\xi S_\alpha^{\eta_1,0} S_\alpha^{\eta_2,0} + \mathbf{n}_{\eta_1} S_\alpha^{\xi,0} S_\alpha^{\eta_2,0} + \mathbf{n}_{\eta_2} S_\alpha^{\xi,0} S_\alpha^{\eta_1,0} \quad (1.31)$$

The values aligned with the transverse directions, $u(\xi_\alpha, \eta_{1,j}, \eta_{2,0})$ and $u(\xi_\alpha, \eta_{2,0}, \eta_{1,j})$, are retrieved by interpolation, using secondary interpolation points (see Figure 1.4), similarly to the two dimensional approach.

When considering a Neumann boundary condition, the jumps are then computed using Eqs. (1.20), (1.21) and the derivative of u at the interface in the ξ direction from Eq. (1.30),

$$\frac{\partial^k u}{\partial \xi^k} \Big|_\alpha = S_\alpha^{\xi,k} \frac{\partial u}{\partial \xi} \Big|_\alpha + \sum_{i=1}^3 S_i^{\xi,0} u(\xi_{i+1}, \eta_{1,0}, \eta_{2,0}) + \mathcal{O}(h^{4-k}) \quad (1.32a)$$

Evaluation of first degree differential operators

As already mentioned in 1D, there is a large number of combinations between the needed accuracy and type of boundary condition. We here present two of them in a 2D framework: the evaluation of a first degree operator and the extrapolation of the field when no information is available from the boundary condition. Following the same approach as in 1D, when evaluating first degree operators, we use a degraded version of the presented method: only two primary interpolation points are needed and the wall derivatives are then expressed as

$$\frac{\partial^k u}{\partial \xi^k} \Big|_\alpha = \tilde{S}_\alpha^{\xi,k} u_\alpha + \sum_{i=1}^2 \tilde{S}_i^{\xi,k} u(\xi_{i+1}, \eta_\alpha) + \mathcal{O}(h^{3-k}) \quad (1.33a)$$

$$\frac{\partial^k u}{\partial \eta^k} \Big|_\alpha = \tilde{S}_\alpha^{\eta,k} u_\alpha + \sum_{j=1}^2 \tilde{S}_j^{\eta,k} u(\xi_\alpha, \eta_j) + \mathcal{O}(h^{3-k}) \quad (1.33b)$$

with

$$u(\xi_\alpha, \eta_j) = \sum_{i=1}^3 \tilde{R}_i^0 u(\xi_{i-2+s_{\eta_j}}, \eta_j) + \mathcal{O}(h^3) \quad (1.34)$$

Again, s_{η_j} is chosen such that the interpolation error is reduced.

When no information is available from the boundary condition, the field is extrapolated up to the wall using the following formula,

$$\frac{\partial^k u}{\partial \xi^k} \Big|_\alpha = \sum_{i=1}^3 \tilde{T}_i^{\xi,k} u(\xi_{i+1}, \eta_0) + \mathcal{O}(h^{3-k}) \quad (1.35a)$$

$$\frac{\partial^k u}{\partial \eta^k} \Big|_\alpha = \sum_{i=1}^3 \tilde{T}_i^{\eta,k} u(\xi_\alpha, \eta_i) + \mathcal{O}(h^{3-k}) \quad (1.35b)$$

and Eq. (1.34). The way we compute the interpolation/extrapolation weights (S , R , T , $\tilde{S}$, $\tilde{R}$, $\tilde{T}$) is detailed in Appendix D.1 along with a detailed description of the different Immersed Interface stencils used in Appendix D.2 (in 2D).

Computational overhead

The computation cost of the construction of the stencil is small compared to an operation in the whole computation domain. Indeed, let us consider an obstacle of length D (in the case of a sphere, $D = 2R$) and an associated grid-resolution $N = D/h$ (the computation domain has a global resolution of N^d where $d = 2$ or $d = 3$ in 2D or 3D respectively). For a 2D problem, in each direction, the interface will cross the grid $2D/h$ times. It will then generate $4D/h$ control points and $8D/h$ biased points. For a 3D problem, we have to consider $2D/h$ slices in Z direction and each one of them will contain up to $4D/h$ control points, in the X and Y directions. We then slice the domain in the X direction to obtain $2D/h$ slices containing $2D/h$ intersections in Z direction. We will then have up to $12D^2/h^2$ control points and $24D^2/h^2$ biased points, in the worst case. For each biased point, the interpolation weights are computed locally by solving a few small linear systems (4×4 matrices).

The global computational overhead, then scales as $\mathcal{O}(N^{(d-1)})$. This is relatively small compared to the cost of evaluating a source term $\mathcal{O}(N^d)$ or to the cost of a Poisson solver ($\mathcal{O}(N^d \log(N^d))$). The real challenge in the Immersed Interface lies in the implementation on massively distributed architectures as the load balancing is not trivial and the communications may be expensive. The lack of a dedicated treatment of this issue can result in a high penalty as only a few processors would be concerned by the IIM jumps computations and its application.

An efficient iterative penalization method using recycled Krylov subspaces and its application to impulsively started flows

This chapter is based on the published article *An efficient iterative penalization method using recycled Krylov subspaces and its application to impulsively started flows*, T. Gillis, G. Winckelmans, P. Chatelain, *Journal of Computational Physics* 347 (2017) 490-505.

2.1 Introduction

Vortex methods use the vorticity-velocity formulation of the incompressible Navier-Stokes equations. Re-meshed vortex particle (VP) methods have been shown to efficiently and accurately simulate complex viscous vortex flows [2, 3, 10, 30, 63]. The Lagrangian treatment of the convection term, combined with the re-meshing operation leads to a method with accurate treatment of the convection. The re-meshed and hybrid approach, as in [2, 3, 24, 26, 36, 83, 84], here called the Vortex Particle-Mesh (VPM) method, sometimes also called the Vortex-In-Cell (VIC) method, combines the advantage of a particle method with those of a mesh-based approach. Several techniques have been proposed for embedding solid bodies with arbitrary geometries in VPM methods [10, 30]. Most recent efforts use Immersed Interface method (IIM) [76, 81] or penalization techniques [32, 55, 56]. For the

latter, an iterative penalization method has been proposed by Hejlesen et al. [34], in order to improve the accuracy at the boundary, avoid diffusion errors [85] and keep the time-step sufficiently large. However, this iterative procedure is highly expensive since, at each iteration, one needs to recover velocity from vorticity by solving a Poisson equation. When considering unbounded problems, this can be carried out on a small sub-domain around the body; yet it still remains the bottleneck operation of such method. Moreover, in a parallel computing context, this domain change implies some important data mappings in order to maintain acceptable load balancing within the penalization sub-problem. In this work, we propose to formulate the penalization problem as a linear system and to use advanced iterative methods to reduce the number of iterations needed by the penalization sub-step.

Iterative methods for generic linear systems as $Ax = b$ are widespread. When the matrix A is Hermitian, one of the prominent algorithms, the conjugate gradient method (CG) [86] uses short 3-term recurrences to build the solution iteratively. This implies that the storage overhead is low enough to be used in CFD applications. To solve more general systems, one usually uses the GMRes algorithm [86, 87]. Since this method is based on the construction of an orthogonal basis of the Krylov subspace at each iteration, the cost in memory and orthogonalization increases with the number of iterations. Therefore, another approach has also been proposed in the literature: transform short recurrences from the CG algorithm to make it suitable for arbitrary matrices. Two generalizations of CG based on bi-orthogonalization were proposed: BiCG and QMR. BiCG, however, requires two matrix - vector products involving A and A^T . To overcome this limitation, Sonneveld [88] proposed a transpose-free variant: the conjugate gradient squared algorithm (CGS) which improves the convergence by using the “wasted” matrix A^T - vector multiplication. Yet, this variation can exhibit a build-up of rounding errors and erratic convergence. An improved version has then been proposed by van der Vorst [89], BiCGStab, which stabilizes the convergence behavior. This method rapidly became famous due to its low computational cost and simplicity.

The integration of time-dependent PDEs very often involves a linear system solution at each time step; this can be due to either an elliptic sub-problem (as in the present fluid mechanics problem) or the use of implicit time integration. Linear solvers are thus ubiquitous in computational science and constitute a focal point of research in applied mathematics. When using an iterative solver, the computational cost will be directly governed by the number of iterations needed to converge. In the context

of a time-dependent problem, this motivates the recycling of the Krylov subspaces obtained during the previous solution. Generalized conjugate residual method with inner orthogonalization (GCRO) [90], GMRes and BiCGStab with deflated restarting [91–94] and GCRO with optimal truncation (GCROT) [95] are as many examples of such algorithms. Assuming a continuous behavior of the PDE and its solution in time, Amritkar et al. [96] proposed a recycling solver based on the combination of both GMRes with deflated restarting and recycling BiCGStab and applied it to standard CFD problems such as turbulent channel flows and flows through porous media. We further extend this latter work to the penalization method in the vorticity formulation of the incompressible Navier-Stokes equations. The computational gains are assessed on the salient and enduring challenge of taking complex geometries into account with VPM methods in an efficient and flexible manner.

In this paper, we present the transposition of the recycling BiCGStab to the penalization in VPM methods to reduce the number of iterations needed in order to converge. We assess its computational gains on benchmark problems: the number of calls to the Poisson solver, the convergence and the memory overhead. We first briefly recall the Vortex Particle-Mesh (VPM) method and the relevant algorithmic techniques, then we detail the proposed penalization algorithm in Section 2.2. In §2.3 and §2.4, we present an assessment of the computational efficiency of our method. We close this paper in Section 2.5 with a conclusion and perspectives for future work.

2.2 Methodology

2.2.1 Vortex Particle-Mesh (VPM) method and Brinkman penalization

We briefly present Vortex Particle-Mesh methods, the use of penalization techniques and the way the penalization step is solved using iterative methods. Particle-mesh methods are well suited for solving advection dominated problems. The VPM method is a prime example; it solves the vorticity formulation of the incompressible Navier-Stokes equations [2, 3, 24, 34, 36]. In 2D, we only have the out-of-plane vorticity, $\omega = \omega \mathbf{e}_z$, and

$$\frac{D\omega}{Dt} = \frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega , \quad (2.1)$$

where $\mathbf{u}$ is the velocity which has to be recovered through the Poisson problem,

$$\nabla^2 \mathbf{u} = -\nabla \wedge \omega . \quad (2.2)$$

The vorticity field, ω , is discretized by means of particles with positions $\mathbf{x}_p$, material surfaces S_p and integral of vorticity $\Gamma_p = \int_{S_p} \omega \, d\mathbf{x} \triangleq \omega_p S_p$. The mesh is used to carry out operations such as the solution of Eq. (2.2) and operator computations [2, 24], while the time integration is carried out on the particles. The 2D VPM method then solves the following set of equations

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \omega_p \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \nu \nabla^2 \omega \end{bmatrix}_p . \quad (2.3)$$

As a consequence, particle-mesh methods need to interpolate the vorticity field and the right-hand sides of (2.3) between the particles and the mesh.

Brinkman penalization methods [34, 56] embed the object in the fluid domain and enforce the velocity inside the obstacle, Ω_b , to be $\mathbf{u} = \mathbf{u}_b$, where $\mathbf{u}_b$ is the desired velocity inside the immersed body. It adds this constraint to the Navier-Stokes problem using an additional penalization term, which will behave as a relaxation of any spurious velocity inside the object. Since the constraint is imposed on the velocity, these methods are naturally applied to pressure-velocity formulation of the Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \nu \nabla^2 \mathbf{u} + \lambda \chi (\mathbf{u}_b - \mathbf{u}) , \quad (2.4)$$

where λ is the Lagrange multiplier and χ is the mask function taken as being

$$\chi(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_b, \\ 0 & \text{otherwise.} \end{cases} \quad (2.5)$$

Applying the curl operator to Eq. (2.4) leads to the vorticity formulation. In 2D, one then obtains the penalized form of the Navier-Stokes equations [57]:

$$\frac{\partial \omega}{\partial t} + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega + \lambda \nabla \wedge [\chi (\mathbf{u}_b - \mathbf{u})] . \quad (2.6)$$

For each particle, we then solve the following set of equations

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \omega_p \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \nu \nabla^2 \omega + \lambda \nabla \wedge [\chi (\mathbf{u}_b - \mathbf{u})] \end{bmatrix}_p . \quad (2.7)$$

These ordinary time-differential equations are typically time-integrated using a second or higher order scheme (possibly using the first order Euler scheme for the diffusion part) and the velocity field is recovered by solving Eq. (2.2). Solving this Poisson problem is the bottleneck of such methods. For the unbounded domain considered in this chapter, this is handled by using the Hockney-Eastwood algorithm which

works on the Fourier transform of the zero-padded doubled-domain source field, see Chapter 1.

The time integration of Eq. (2.7) is carried out through a splitting approach [42]. Given $\mathbf{u}^n$, ω^n and Δt , one goes through the following substeps:

1. Convection of the particles and interpolation of the vorticity onto the mesh (or remeshing).
2. Diffusion on the mesh (here presented using the Euler scheme)

$$\tilde{\omega}^{n+1} = \omega^n + \Delta t \left[\nu \nabla^2 \omega^n \right] . \quad (2.8)$$

3. Computation of the unpenalized velocity field, on the mesh, by solving

$$\nabla^2 \tilde{\mathbf{u}}^{n+1} = -\nabla \wedge \tilde{\omega}^{n+1} . \quad (2.9)$$

4. Computation of the correction vorticity, on the mesh

$$\delta\omega = \mathcal{F}(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}) , \quad (2.10)$$

where $\mathcal{F}$ is a function that varies from one penalization technique to another.

5. Update of the vorticity

$$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega \quad (2.11)$$

and interpolate on the particles.

The function $\mathcal{F}$ effectively produces the vortex sheet, $\delta\omega$, that cancels the velocity field inside the body. This leads to the observation that penalization is very similar to the problem solved using boundary elements methods (BEM) [10, 30, 63], as such BEM method also computes the vortex sheet required at each time step in order to cancel the slip velocity at the interface [30]. Both methods aim to capture the generation of the same vortex sheet but through different formulations and discretizations (explicitly, with panels or on a grid).

In the following, we give a short overview of all the penalization approaches proposed in the literature. Without loss of generality, let us consider the penalization subproblem in vorticity formulation,

$$\frac{\partial\omega}{\partial t} = \lambda \nabla \wedge \left[\chi(\mathbf{u}_b - \mathbf{u}) \right] . \quad (2.12)$$

The numerical integration of Eq. (2.12) will drive the accuracy of the flow around and inside the body. Due to the stability constraint, an explicit integration will bound the value of $\lambda\Delta t$, hence leading to very small time steps to ensure accuracy. Therefore, the implicit integration of Eq. (2.12), given by

$$\omega^{n+1} - \tilde{\omega}^{n+1} = \delta\omega = (\lambda\Delta t) \nabla \wedge [\chi (\mathbf{u}_b - \mathbf{u}^{n+1})] , \quad (2.13)$$

with an unconstrained value of $\lambda\Delta t$, will be considered as a better choice. The velocity $\mathbf{u}^{n+1}$ is associated with the unknown vorticity, ω^{n+1} , through the Poisson equation and is then also part of the unknowns. This formulation is therefore not usable in practice. Many alternative techniques have been proposed to circumvent this issue. A first widespread approach [32, 42, 57, 97] approximates Eq. (2.13) by the following function

$$\mathcal{F}(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}) = \nabla \wedge \left[\frac{\tilde{\mathbf{u}}^{n+1} + (\lambda\Delta t) \chi \mathbf{u}_b}{1 + (\lambda\Delta t) \chi} \right] - \tilde{\omega}^{n+1} , \quad (2.14)$$

which comes from an implicit Euler integration of the penalization sub-problem in velocity formulation,

$$\frac{\partial \mathbf{u}}{\partial t} = \lambda \chi (\mathbf{u}_b - \mathbf{u}) . \quad (2.15)$$

Because the implicit Euler scheme is unconditionally stable, one can use an arbitrary high value for $(\lambda\Delta t)$. One also verifies that this formulation is equivalent to

$$\begin{aligned} \omega^{n+1} &= \frac{\tilde{\omega}^{n+1} + (\lambda\Delta t) \nabla \wedge (\chi \mathbf{u}_b)}{(1 + (\lambda\Delta t) \chi)} \\ &\quad + [1 + (\lambda\Delta t) (\nabla \chi)]^{-1} \wedge (\tilde{\mathbf{u}}^{n+1} + (\lambda\Delta t) \chi \mathbf{u}_b) \end{aligned} \quad (2.16a)$$

$$\begin{aligned} &= \tilde{\omega}^{n+1} + (\lambda\Delta t) [\nabla \wedge (\chi \mathbf{u}_b) - \chi \nabla \wedge (\mathbf{u}^{n+1})] \\ &\quad + [1 + (\lambda\Delta t) (\nabla \chi)]^{-1} \wedge (\tilde{\mathbf{u}}^{n+1} + \lambda\Delta t \chi \mathbf{u}_b) . \end{aligned} \quad (2.16b)$$

The mathematical error associated to this integration scheme, with respect to the perfect correction Eq. (2.13) is then given by

$$(\nabla \chi) \wedge \mathbf{u}^{n+1} + [1 + \lambda\Delta t (\nabla \chi)]^{-1} \wedge (\tilde{\mathbf{u}}^{n+1} + \lambda\Delta t \chi \mathbf{u}_b) . \quad (2.17)$$

As claimed by Rasmussen et al. [85], this error term generates substantial diffusion. Let us mention another approximation which neglects the term $(\nabla \chi) \wedge (\mathbf{u}^{n+1})$ and leads to an implicit integration of the penalization equation Eq. (2.12), with an unconstrained value of $(\lambda\Delta t)$,

$$\mathcal{F}(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1}) = \frac{\omega^n + (\lambda\Delta t) \nabla \wedge (\chi \mathbf{u}_b)}{(1 + (\lambda\Delta t))} - \tilde{\omega}^{n+1} . \quad (2.18)$$

As explained by Hejlesen et al. [34], both approaches do not provide a global relation between vorticity and velocity which leads to a dramatically decreasing time step in order to provide accurate results.

Since getting the correct vortex sheet, i.e. the correct pair $(\mathbf{u}^{n+1}; \omega^{n+1})$, is crucial for capturing unsteady behaviors such as the added mass effects of accelerated objects, Hejlesen et al. [34] proposed another way to evaluate $\mathcal{F}(\mathbf{u}_b, \tilde{\mathbf{u}}^{n+1})$. It is proposed to solve the following problem: at time step n , given a non-penalized vorticity field, $\tilde{\omega}^{n+1}$, (and an associated non-penalized velocity field, $\tilde{\mathbf{u}}^{n+1}$); the iterative penalization method aims at finding the vorticity correction, $\delta\omega$, which generates the velocity correction, $\delta\mathbf{u}$, such that $\tilde{\mathbf{u}}^{n+1} + \delta\mathbf{u} = \mathbf{u}^{n+1} = \mathbf{u}_b$ in Ω_b . The method proposed by Hejlesen et al. [34] iterates on this vorticity correction to converge to consistent vorticity and velocity fields (see algorithm 1). We rather formulate this problem as a linear system

$$A(\delta\omega) = \nabla \wedge [\chi(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1})] \quad , \quad (2.19)$$

where $\delta\omega = \delta\omega \mathbf{e}_z$ and the linear operator A is composed as

$$A : \delta\omega_k \rightarrow \nabla \wedge [\chi \delta\mathbf{u}_k] = \nabla \wedge \left[\chi \left((\nabla^2)^{-1} (-\nabla \wedge \delta\omega_k) \right) \right] \quad . \quad (2.20)$$

We notice that this operator is non-symmetric because of the χ function. If we pursue the above analogy with the corresponding boundary integral equation in two dimensions, we expect the operator A to have a non-void kernel. The uniqueness of the solution is therefore not ensured. Indeed, the solution of Eq. (2.19) is defined up to a constant which sets the circulation around the object. Nevertheless, as reported by various authors [34, 42, 57, 85, 97] and observed here, one does not need to impose the vorticity integral explicitly.

Within this framework, the iterative method proposed by Hejlesen et al. [34] can be reinterpreted as an over-relaxed Jacobi iteration. The stability constraint reported by the authors is in fact the well known stability constraint for this kind of method $0 < \alpha = (\lambda \Delta t) < 2$.

Crucially, a single matrix-vector product, $A \delta\omega_k$, is expensive: therein lies the solution of a Poisson equation, as shown in Eq. (2.20). Because each iteration of the penalization involves such an expensive component, this potentially makes the boundary condition enforcement the bottleneck of the global algorithm. To reduce the cost, Hejlesen et al. [34] proposed to reduce the computational domain of the penalization method. When using a FFT-based Poisson solver, this domain restriction can only be performed in the unbounded direction. Still, it reduces the computational and memory overheads due to the penalization. Nevertheless, a crucial issue remains: a Jacobi method will require more iterations compared to more advanced algorithms, making

Given: $\tilde{\mathbf{u}}^{n+1}, \tilde{\omega}^{n+1}$,
Set: $\delta\omega_0 = 0; \delta\mathbf{u}_0 = 0$
Compute: $b = \nabla \wedge [\chi(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1})]; r_0 = b$
for $k = 1, 2, \dots$ **do**
 $\delta\omega_k = (1 - \alpha)\delta\omega_{k-1} + \alpha r_{k-1}$
 $\nabla^2(\delta\mathbf{u}_k) = -\nabla \wedge \delta\omega_k$
 $r_k = \nabla \wedge [\chi(\mathbf{u}_b - (\tilde{\mathbf{u}}^{n+1} + \delta\mathbf{u}_k))]$
end
 $\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$

Algorithm 1: Penalization algorithm - Jacobi(α) from Hejlesen et al. [34].

its cost significant with respect to the rest of the time-step. One needs then to reduce the number of iterations needed to reach convergence and hence, there is a strong motivation to reduce the number of iterations.

2.2.2 Krylov-based iterative methods

In this section, we briefly present the BiCGStab algorithm and the way to recycle Krylov subspaces with this method. We refer to [86] for the fundamentals on this class of methods.

We consider a non-Hermitian matrix, $A \in \mathbb{R}^{n \times n}$. One of the most famous algorithm for such matrices is GMRes. It consists in computing iteratively an orthogonal basis, V , of $\mathcal{K}(A)^1$, such that, at each iteration,

$$V^T A V = H \quad \text{where} \quad V^T V = I \quad (2.21)$$

and where H is a Hessenberg matrix (i.e $H_{ij} = 0$ if $i > j + 1$). Alternatively, there is the BiCGStab method. It is a member of the bi-conjugated algorithm family which replaces the Hessenberg matrix, H , by a tridiagonal one using bi-orthogonalization. In other words, we bi-orthogonalize A

$$W^T A V = T \quad \text{where} \quad \begin{cases} W^T V = I \\ AV = V T \\ A^T W = W T^T \end{cases}, \quad (2.22)$$

¹ $\mathcal{K}(A) = \text{span}\{b, A(b), A^2(b), A^3(b), \dots\}$ denotes the Krylov subspace of the operator A .

where T is the tridiagonal matrix. One of the first bi-orthogonalized variants of the Conjugated Gradients (CG) method has been proposed by Lanczos [98]: Bi-Conjugated Gradients (BCG). This method iteratively computes a solution of the system $Ax = b$ using 4 vectors r_k, r_k^*, p_k and p_k^* , defined as

$$\begin{aligned} r_k &= \phi_k(A) r_0 & r_k^* &= \phi_k(A^T) r_0^* \\ p_k &= \pi_k(A) r_0 & p_k^* &= \pi_k(A^T) r_0^* \end{aligned} \quad (2.23)$$

where ϕ_k is a certain polynomial of degree k which satisfies $\phi_k(0) = 1$ and π_k is another polynomial of degree k . The series of ϕ and π polynomials are initialized as $\phi_0 = \pi_0 = 1$. This algorithm then iteratively computes the solutions using short recurrences [86]

$$\begin{aligned} \alpha_k &= \frac{(\phi_k(A) r_0, \phi_k(A^T) r_0^*)}{(A \pi_k(A) r_0, \pi_k(A^T) r_0^*)} \\ \phi_{k+1}(x) &= \phi_k(x) - \alpha_k x \pi_k(x) \\ \beta_k &= \frac{(\phi_{k+1}(A) r_0, \phi_{k+1}(A^T) r_0^*)}{(\phi_k(A) r_0, \phi_k(A^T) r_0^*)} \\ \pi_{k+1}(x) &= \phi_{k+1}(x) + \beta_k \pi_k(x) . \end{aligned} \quad (2.24)$$

The drawback of this method is to use 2 matrix-vector products (one with A and one with A^T) at each iteration to compute α_k and β_k . To take advantage of this wasted matrix-vector product, Sonneveld [88] proposed the Conjugated Gradient Squared (CGS) algorithm which is based on the observation that α_k can be rewritten as

$$\alpha_k = \frac{(\phi_k^2(A) r_0, r_0^*)}{(A \pi_k^2(A) r_0, r_0^*)} . \quad (2.25)$$

Therefore it seeks for a residual defined by

$$r_k = \phi_k^2(A) r_0 . \quad (2.26)$$

However, this method shows poor convergence behavior. To stabilize CGS, van der Vorst [89] introduced BiCGStab, a stabilized version where the residual is defined as

$$r_k = \psi_k(A) \phi_k(A) r_0 , \quad (2.27)$$

where $\psi_k(A)$ is a polynomial assembled iteratively and chosen to smooth the convergence. This new ψ polynomial avoids the use of the squared ϕ , which damages the convergence behavior. It endows the method with an improved convergence behavior but, its convergence is not guaranteed to be monotonic anymore.

2.2.3 Recycling methods

The recycling of information from one solution attempt to the next, on a perturbed system, has become a much pursued area of research in linear algebra. All recycling methods use the same framework: they first compute the best solution within a known space and then solve the remaining part of the system [92, 94, 96]. In order to do so, they rely on 2 subspaces: $\mathcal{U}$, and $\mathcal{C}$, both of dimension l , related by $A : \mathcal{U} \rightarrow \mathcal{C}$. Recycling methods store 2 bases which span those 2 subspaces: $U \in \mathbb{R}^{n \times l}$ and $C \in \mathbb{R}^{n \times l}$, respectively. Given those two bases, an initial solution, x_{-1} and its associated residual, $r_{-1} = b - Ax_{-1}$, recycling methods first seek for the best initial guess, $x_0 = x_{-1} + Uy$, where $y \in \mathbb{R}^l$, that verifies

$$x_0 = \arg \min_y \|r_0\|_2, \quad (2.28)$$

subject to the constraints

$$AU = C \quad \text{and} \quad C^T C = I. \quad (2.29)$$

This problem becomes

$$y_0 = \arg \min_y \|r_{-1} - AUy\|_2 = \arg \min_y \|r_{-1} - Cy\|_2 = C^T r_{-1}. \quad (2.30)$$

The optimal solution, x_0 , and its residual are then given by the following expressions [91]

$$x_0 = x_{-1} + UC^T r_{-1} \quad r_0 = r_{-1} - CC^T r_{-1}. \quad (2.31)$$

The original problem $Ax = b$ has then become

$$A(x_0 + \zeta) = b \quad \Rightarrow \quad A\zeta = r_0, \quad (2.32)$$

which we still have to solve. This is done using widespread algorithms like BiCGStab. However, since x_0 is the best solution in $\mathcal{U}$, we know that $r_0 \in \mathcal{C}^\perp$. To enforce this on Eq. (2.32) and inside the iterative solver, we orthogonalize the remaining problem with respect to $\mathcal{C}$ and solve the modified problem

$$A_C \bar{\zeta} = [I - CC^T] A \bar{\zeta} = \bar{r}_0 = r_0. \quad (2.33)$$

Since we solved Eq. (2.33) instead of Eq. (2.32) we have to recover ζ from $\bar{\zeta}$, which is done using

$$x = x_0 + \zeta = x_0 + [I - UC^T A] \bar{\zeta}, \quad (2.34)$$

as presented in [91].

2.2.4 Recycling for the iterative penalization method

Instead of using a Jacobi iteration to solve the penalization part of Eq. (2.6), we propose to use the rBiCGStab algorithm of Section 2.2.3, that recycles solutions from previous time steps. We observe that the solution of the penalization problem, Eq. (2.19), does not change much from one time step to another and therefore previous solutions are accurate initial guesses. This rBiCGStab(l) algorithm will then store up to l previous solutions and use them to compute the best initial iterate possible (Algorithm 2). What might seem like an expensive memory overhead is actually limited as the support of the corrective vorticity consists in the body and its direct surroundings. Additionally, for an unbounded flow, we can also reduce the computational domain, as done in [34].

Algorithm 2 has two major advantages. First, the memory cost is constant with the number of iterations, which is not the case for a GMRes-like method. Second, performing the small orthogonalization is cheaper than solving a Poisson equation ($\mathcal{O}(l n)$ versus $\mathcal{O}(n \log n)$ where l is chosen such that $l \ll n$). Furthermore, in this version of the algorithm, the orthogonalization is carried out using a modified Gram-Schmidt algorithm.

We note some drawbacks, however. First, it uses two matrix vector products per iteration which makes it more expensive than the Jacobi method per iterate. Still, the number of iterations needed to reach convergence will be much lower. Second, the residual does not converge monotonically as with a GMRes-like method. This is a well-known behavior of this method and it is due to the use of bi-orthogonalization [86].

2.2.5 Convergence criterion for the iterative penalization method

Every iterative method needs a stopping criterion. We recall that the stopping criterion should be an estimator of the solution quality. In Hejlesen et al. [34] the authors propose a way to compute the error at each iteration as the correction vorticity field first moment integral (i.e. the impulse). This stopping criterion is however not general as it allows spurious solutions, which are even-symmetric with respect to the inflow direction. For these reasons, we rather use a dimensionless penalization residual. At a given iteration, k , we compute it as follows

$$E_2^k(r_k) = \frac{1}{\omega_\infty} \sqrt{\frac{1}{S_{\Omega_b}} \|r_k\|_2^2}, \quad (2.35)$$

where $\omega_\infty = U_\infty / L_{\Omega_b}$, S_{Ω_b} is the obstacle reference surface, L_{Ω_b} is the obstacle reference length and the 2 norm is taken as $\|a\|_2^2 = \sum_{i,j} a_{i,j}^2 h^2$ and $r_k = \nabla \wedge [\chi(\mathbf{u}_b - (\tilde{\mathbf{u}}^{n+1} + \delta \mathbf{u}_k))]$.

Given: $\tilde{\mathbf{u}}^{n+1}$, $\tilde{\omega}^{n+1}$, $U \in \mathbb{R}^{n \times l}$, $C \in \mathbb{R}^{n \times l}$ such that $AU = C$ and $C^T C = I$

Set: $k = 0$; $\rho_{old} = \rho = 0$

Compute:
 $b = \nabla \times [\chi(\mathbf{u}_b - \tilde{\mathbf{u}}^{n+1})]$; $\xi_0 = -[C^T b]$; $r_0 = b - C\xi$; $\beta = \|r_0\|$

Choose: $\tilde{r} = r_0$

for $k = 1, 2, \dots$ **do**

$\rho = \tilde{r}^T r_k$

if $\rho == 0$ **then** Breakdown occurred. **exit**

if $i == 0$ **then**

$p = \tilde{r}_k$

else

$\beta = \frac{\rho}{\rho_{old}} \frac{\alpha}{\omega}$

$p = r_k + \beta(p - \omega v)$

end

$v = Ap$; $\eta = C^T v$; $v = v - Cv$

$\alpha = \frac{\rho}{\tilde{r}^T v}$

$\xi_{k+1} = \xi_k + \alpha \eta$; $x_{k+1} = x_k + \alpha p$; $r_{k+1} = r_k - \alpha v$

if $E_2^{k+1} \leq \text{tol}$ **then exit; else** $k = k + 1$

$t = Ar_k$; $\eta = C^T t$; $t = t - C\eta$

$\omega = \frac{t^T r_k}{t^T t}$

$\xi_{k+1} = \xi_k + \omega \eta$; $x_{k+1} = x_k + \omega r_k$; $r_{k+1} = r_k - \omega t$

$\rho_{old} = \rho$

if $E_2^{k+1} \leq \text{tol}$ **then exit; else** $k = k + 1$

end

$\delta\omega = x_{k_{end}} - U\xi_{k_{end}}$

if $\delta\omega \neq U_{1:l}$ **then**

$C_{1:l} = [C_{2:l} \quad b - r_{k_{end}}]$; $U = [U_{2:l} \quad \delta\omega^{n+1}]$

$C_{1:l} = QR$

$U_{1:l} = UR^{-1}$ $C_{1:l} = Q$

end

$\omega^{n+1} = \tilde{\omega}^{n+1} + \delta\omega$

Algorithm 2: Penalization algorithm - rBiCGStab(l).

2.2.6 Force computation

To compute the force exerted by the fluid on the immersed body, we use the momentum formulation from Noca et al. [99], as proposed by Mimeau et al. [32]: considering a control volume, Ω_c , that contains the obstacle, we have that

$$\mathbf{F} = -\frac{d}{dt} \int_{\Omega_c} \mathbf{u} \, d\mathbf{x} + \int_{\partial\Omega_c} \boldsymbol{\gamma} \cdot \mathbf{n} \, d\mathbf{x} \, , \quad (2.36)$$

where $\mathbf{n}$ is normal to $S = \partial\Omega_c$ and

$$\begin{aligned} \boldsymbol{\gamma} = & \frac{1}{2} |\mathbf{u}|^2 \mathbf{I} - \frac{1}{N-1} \mathbf{u} (\mathbf{x} \wedge \boldsymbol{\omega}) + \frac{1}{N-1} \boldsymbol{\omega} (\mathbf{x} \wedge \mathbf{u}) \\ & - \frac{1}{N-1} \left[\left(\mathbf{x} \cdot \frac{\partial \mathbf{u}}{\partial t} \right) \mathbf{I} - \mathbf{x} \frac{\partial \mathbf{u}}{\partial t} \right] \\ & + \frac{1}{N-1} [[\mathbf{x} \cdot (\nabla \cdot \mathbf{T})] \mathbf{I} - \mathbf{x} (\nabla \cdot \mathbf{T})] + \mathbf{T} \, . \end{aligned} \quad (2.37)$$

in which N is the spatial dimension (2 or 3), $\mathbf{I}$ is the unit tensor and $\mathbf{T}$ is the viscous stress tensor, given by $\mathbf{T} = \mu [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]$. In our case, Ω_c is a rectangular box containing the body, Ω_b .

2.3 Validation and analysis

In this section, we use the flow past a cylinder at low Reynolds number ($\text{Re} = 550$) to validate our approach and analyze its behavior. We use an adaptive time step, well suited for the impulsively started flow considered here. The time step, Δt , is constrained by the following accuracy conditions:

$$\Delta t \|\mathbf{D}\|_1 \leq \text{LCFL}_D \quad \text{and} \quad \Delta t \|\boldsymbol{\omega}\|_1 \leq \text{LCFL}_\omega \, , \quad (2.38)$$

where $\mathbf{D} = \frac{1}{2} [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]$ is the strain tensor and $\text{LCFL}_D \simeq \text{LCFL}_\omega \simeq 0.15 \dots 0.25$ are typical values. Throughout this work, we consider a cylinder of diameter D in an computational domain of size $[5D \times 5D]$ and we use the following dimensionless variables:

$$\omega^* = \frac{\omega D}{U_\infty} \quad t^* = \frac{t U_\infty}{D} \quad \text{Re} = \frac{D U_\infty}{\nu} \quad C_D = \frac{\mathbf{F} \cdot \mathbf{e}_x}{\frac{1}{2} \rho U_\infty^2 D} \, . \quad (2.39)$$

Furthermore, we compute the penalization residual as being

$$E_2^k(r_k) = \frac{D}{U_\infty} \sqrt{\frac{4}{\pi D^2} \|r_k\|_2^2} \, , \quad (2.40)$$

2.3.1 Flow past a cylinder at $Re = 550$

For this validation, we consider that the iterative procedure reached convergence when $E_2^k < 10^{-4}$. To put this convergence criterion in perspective, one should keep in mind that, following the Falkner-Skan theory, the momentum thickness of the boundary layer near the stagnation point is given by $\theta/D \simeq 0.1462/\sqrt{Re}$. Using this thickness instead of D in the scaling coefficient $\frac{D}{U_\infty}$ would give a new convergence criterion of $< 6.23 \cdot 10^{-7}$, which is more representative of the physical interpretation behind the residual, but it requires to have a priori information about the flow solution. The resolution of the full domain is $\frac{h}{5D} = \frac{1}{1024}$ and the subdomain used for the penalization problem, $\left[\frac{5}{4}D ; \frac{5}{4}D\right]$, is centered on the cylinder. The time step coefficients are taken as $LCFL_\omega = LCFL_D = 0.25$. We compare our drag coefficient to three references in the literature: Koumoutsakos and Leonard [63] (KL), Ploumhans and Winckelmans [10] (PW) and Marichal et al. [76, 81] (M). An analytic solution valid for short times also exists [100] and is given by

$$C_D = 4 \left(\frac{\pi}{Re \, t^*} \right)^{1/2} + 2\pi \left(9 - \frac{15}{\sqrt{\pi}} \right) \frac{1}{Re} . \quad (2.41)$$

Figure 2.1 shows the evolution of the drag coefficient, defined at Eq. (2.39), for short times.

As expected, we underestimate C_D at the very early times (the theoretical behavior is singular at $t = 0^+$, corresponding to an infinitely thin boundary layer). Soon after, our simulation is well-resolved and the C_D curve falls back onto the theoretical curve, (momentarily though, as this curve is only valid for short times). Our results also compare favorably with all the references results of PW [10], KL [63] and M [76, 81]. The present method also shows a good agreement with the IIM Vortex Method of Marichal et al. [76], which performs the highest order treatment of the wall boundary condition among the reference results. It is important to note that our results are not filtered and still exhibits a very low noise signal, unlike those of PW [10] who used filtering (visible in the zoom $t^* \in [0.15 ; 0.25]$).

2.3.2 Analysis of the computational cost

In this section, we compare the computational costs required by Jacobi(1.9) and rBiCGStab. The resolution of the full domain is $\frac{h}{5D} = \frac{1}{256}$, the penalization subdomain and the time step coefficients are as in §2.3.1. The error allowed inside the body (and therefore the norm of the residual) will lead to an incorrect vorticity flux at

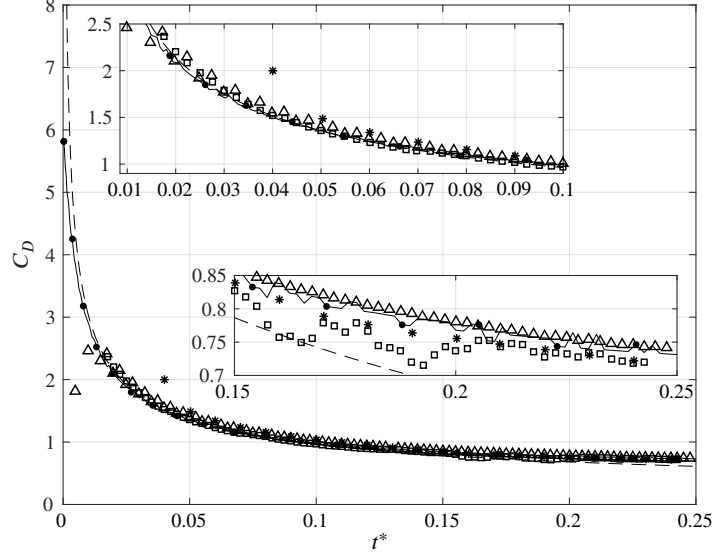


Figure 2.1: **Impulsively started cylinder at $Re = 550$:** C_D , penalization rBiCGStab(4) (—●—), KL [63] (*), PW [10] (□), M [76, 81] and PhD thesis [14] (△) and Eq. (2.41) (---). The first time step is $\Delta t^* = 10^{-6}$.

the wall and therefore will potentially pollute the flow solution outside the body. Following Eq. (2.35) and given a fixed time t^* , we compute the so-called “external error” as

$$\tilde{E}_2^{t^*}(m, l) = \frac{D}{U_\infty} \sqrt{\frac{4}{\pi D^2} \left\| \left(\omega^{t^*}(m, l) - \omega_{ref}^{t^*} \right) (1 - \chi) \right\|_2^2}, \quad (2.42)$$

where m is the number of matrix-vector products, l is the dimension of the recycled subspace and $\omega_{ref}^{t^*}$ is the reference solution at the given time t^* . We chose $\omega_{ref}^{t^*}$ as the solution obtained by iterating till breakdown or convergence to a tolerance of 10^{-10} . Breakdown refers to a situation where the algorithm finds itself stuck: such a condition arises when the residual r is aligned with the conjugated residual $\hat{r}$. In the test performed for this case, breakdown solutions are still well converged ($E_2 \leq 10^{-8}$). However, this could be easily solved by restarting the algorithm and therefore picking another $\hat{r}$. Since we are interested only in starting times, we define the “short-time total

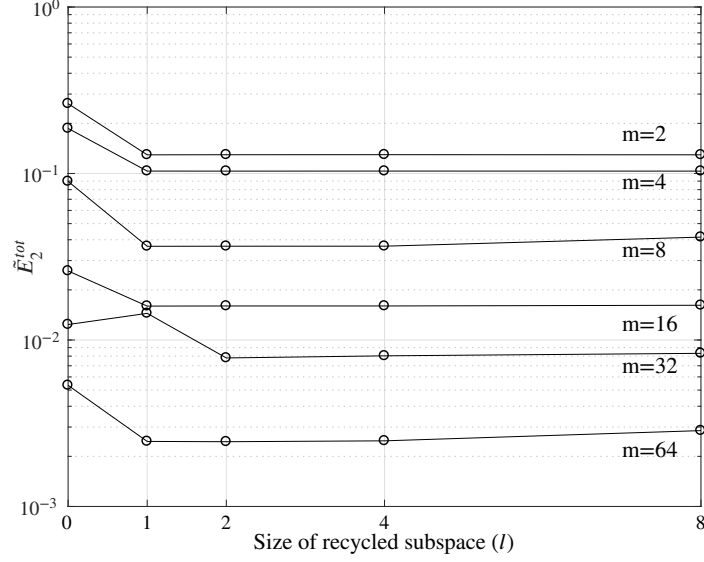


Figure 2.2: **Impulsively started cylinder at $Re = 550$:** $\tilde{E}_2^{tot}$ as a function of the memory overhead.

external error" as being

$$\tilde{E}_2^{tot}(m, l) = \sum_{n=0}^{10} \tilde{E}_2^{(n\Delta t^*)}(m, l) \quad \text{with} \quad \Delta t^* = 0.01 \quad . \quad (2.43)$$

Influence of the number of iterations performed on the flow accuracy

The CPU-cost is directly driven by the number of matrix-vector (matvec) products computed in the iterative process. In Fig. 2.3, we further analyze the relationship between the computational cost invested in the penalization step and the error level in the flow, as defined by Eq. (2.42). The convergence slopes for $\tilde{E}_2^{0+}$, $\tilde{E}_2^{0.1}$ and $\tilde{E}_2^{tot}$ are shown in Table 2.1. There are several notable differences between the behaviors of Jacobi and rBiCGStab-based methods. First we observe that the slopes of Jacobi are $\simeq 0.7$, while the slopes for rBiCGStab are $\simeq 1.1$, highlighting a real computational gain for rBiCGStab (cfr Table 2.1). We also notice, on the left part of Figure 2.3, that the recycled subspace is very useful since it allows to reduce the mean error in the flow with non-CPU overhead. Of course, storing additional solutions on the subdomain slightly increases the memory footprint, an overhead quite acceptable in view

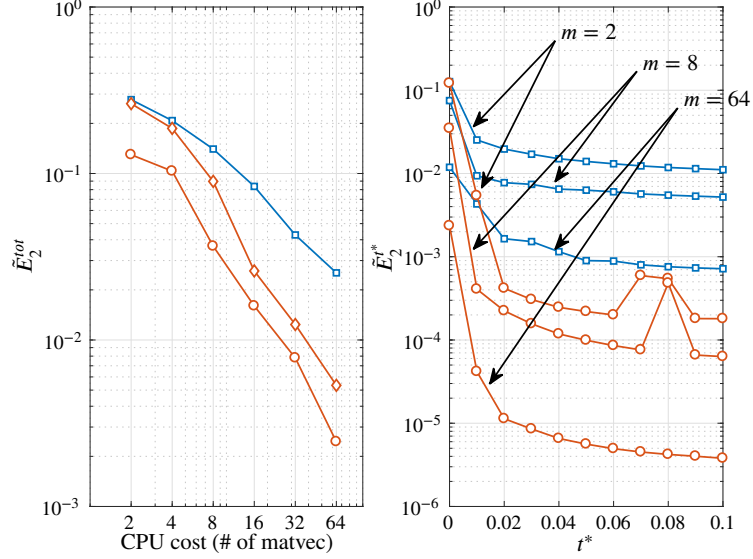


Figure 2.3: **Impulsively started cylinder at $Re = 550$:** (left) $\tilde{E}_2^{tot}$ as a function of the CPU cost (which is directly proportional to the number of matvecs). (Right) $\tilde{E}_2^*$ as a function of time for Jacobi and rBiCGStab(2). Jacobi(1.9) ($\square$), rBiCGStab(0) ($\diamond$) and rBiCGStab(2) ($\circ$).

	Jacobi(α)	rBiCGStab(l)				
	$\alpha = 1.9$	$l = 0$	$l = 1$	$l = 2$	$l = 4$	$l = 8$
$\tilde{E}_2^{0+}$	-0.68	-1.16	-1.16	-1.16	-1.16	-1.16
$\tilde{E}_2^{0.1}$	-0.80	-1.21	-1.02	-1.14	-0.89	-0.41
$\tilde{E}_2^{tot}$	-0.71	-1.19	-1.09	-1.17	-1.17	-1.14

Table 2.1: **Impulsively started cylinder at $Re = 550$:** convergence slopes for different algorithms based on the $\tilde{E}_2^*$ error defined by Eq. (2.42) and the $\tilde{E}_2^{tot}$ error defined by Eq. (2.43).

of the accuracy and CPU gains. Second, we observe, on the right part of Figure 2.3, that reusing subspaces and using rBiCGStab can lead to non monotonic convergence behavior (which, we recall, is inherent to the bi-conjugated methods). However, the error is still lower than the error performed by Jacobi. As expected, we also see that the gain in accuracy is more important at the very beginning of the simulation than when the flow is more developed.

Influence of the memory overhead on the recycled solution

The next parameter of interest is the dimension of the subspace to recycle, as presented on Figure 2.2. We see that the advantage of taking l previous solutions into account quickly decreases with l . In the present case, choosing a value of $l = 2$ puts us in the plateau area and guarantees that we are not involved in the non-monotonic behavior of BiCGStab. We conclude that, even when using a “large” time-step (i.e. $C_1 = C_2 = 0.25$), two previous solutions are enough for computational acceleration and do not imply a large memory overhead. However, this conclusion needs to be put in perspective since we consider very early times with a marked unsteadiness at $t = 0^+$. After this strong transient however, the flow development is smoother and therefore easily captured by the previous solution.

2.4 Results on impulsively started flows

2.4.1 Flow past a cylinder at $Re = 9500$

Vortex methods are well suited for moderately high Reynolds flows. In this section, we present results for the flow past a cylinder at $Re = 9500$ and assess the computational gain. We consider the algorithms Jacobi(1.9) and rBiCGStab(2). The resolution of the full domain is $\frac{h}{5D} = \frac{1}{4096}$. The penalization subdomain stays the same as in §2.3.1 and the time step coefficients satisfies Eq. (2.38) with the same coefficients as in §2.3.1. We consider that both methods have to reach an error of $E_2 \leq 10^{-2}$, as defined by Eq. (2.40), which allows a relatively converged flow solution at reasonable CPU-cost.

Figure 2.4 compares the flow obtained for the two methods considered. The drag coefficients, C_D , shown on Figure 2.5, both exhibit a non-smooth behavior, typical of the method we used to compute the drag force (see §2.2.6). One way to increase the smoothness consists in performing more iterations, thereby decreasing the residual error within the object. Furthermore, one notes that our results compare favorably with those of the references [34, 63].

Fig. 2.6 shows the number of matrix-vector products needed to converge the solution down to the chosen tolerance. We observe that using rBiCGStab(2) reduces the average number of iteration needed by a factor of $\simeq 48$ ($\simeq 22.6$ for Jacobi(1.9) and $\simeq 0.47$ for rBiCGStab(2)). Recycling solutions is therefore very effective at speeding up the computation, even for high Reynolds number flows.

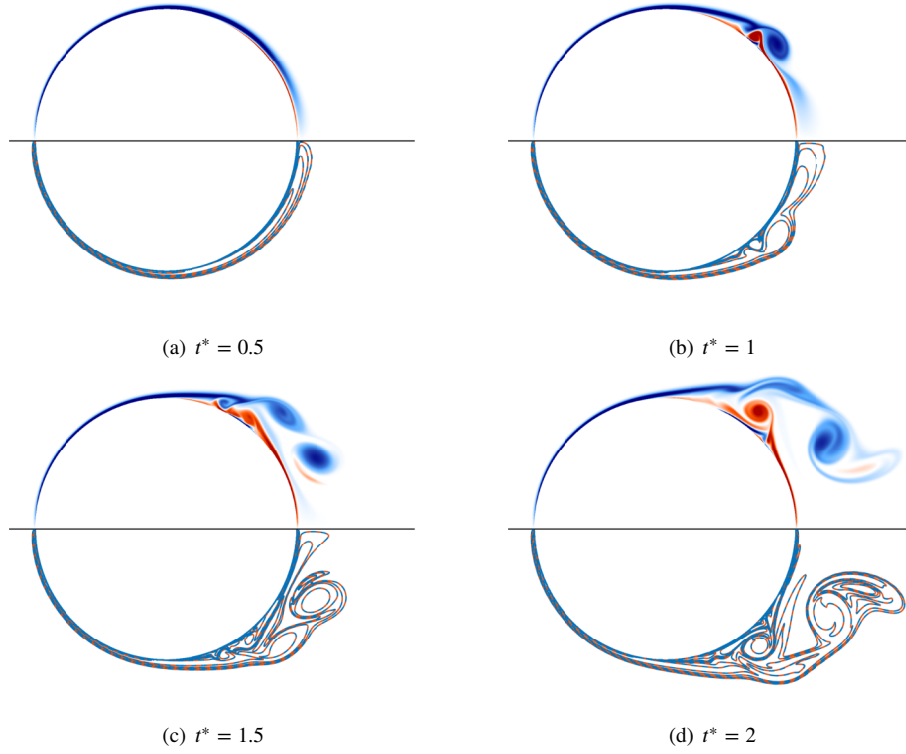


Figure 2.4: **Impulsively started cylinder at $Re = 9500$** : (up) vorticity obtained with rBiCGStab(2); (down) vorticity contour obtained with rBiCGStab(2) (—) and Jacobi(1.9) (—), both with tolerance 10^{-2} .

2.4.2 Flow past a thin plate at $Re = 1000$

In this section, we assess the computational gains on a challenging benchmark characterized by sharp corners and non-negligible acceleration effects, due to impulsively start, that, typically, can pose problems to a penalization method. The resolution of the full domain is $\frac{2h}{5L} = \frac{1}{2048}$, where L is the height of the plate; the penalization subdomain is $(32h \times 1024h)$; the time step coefficients are taken as $LCFL_\omega = LCFL_D = 0.15$ and the flat plate thickness is $T = 8h$, as in [34] ($L/T = 512/5 \simeq 100$). The flat plate constitutes a stiff problem to solve, especially at early times. It is indeed challenging for the penalization method to respond to the very important temporal variations of this flow, when, furthermore, the penalization term support is a thin region: Hejlesen et al. [34] reported that more than 100 iterations are needed in order to converge. Although this convergence criterion is based on the drag coefficient (the iteration stops

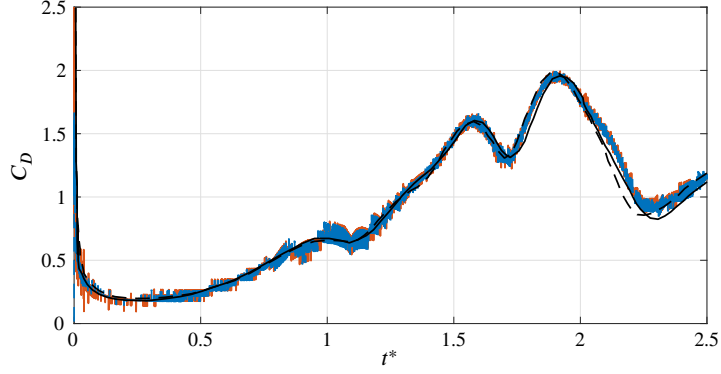


Figure 2.5: **Impulsively started cylinder at $Re = 9500$:** C_D , Jacobi(1.9) (—•—), rBiCGStab(2) (—•—), Hejlesen et al. [34] (---) and Koumoutsakos and Leonard [63] (—).

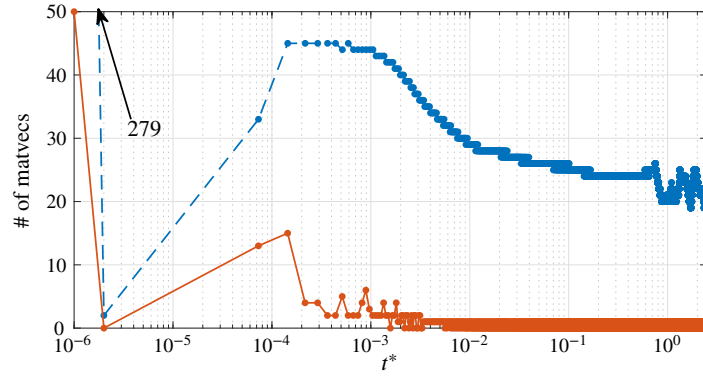


Figure 2.6: **Impulsively started cylinder at $Re = 9500$:** comparison of the number of matvec needed to converge to a tolerance of 0.01 for Jacobi(1.9) (—•—) and rBiCGStab(2) (—•—). The first time step of Jacobi(1.9) needs 279 matvecs.

when the increment of the drag coefficient is less than 0.001 of the drag itself), one could argue that this is too computationally expensive to make it usable within a realistic framework. In our case, we define the error E_2^k , as

$$E_2^k(r_k) = \frac{L}{U_\infty} \sqrt{\frac{1}{LT} \|r_k\|_2^2}. \quad (2.44)$$

The tolerance on this error for both rBiCGStab(2) and Jacobi(1.9) is 10^{-2} . The Reynolds number is defined as $Re = \frac{LU_\infty}{\nu} = 1000$.

The large scales of the flow are shown in Figure 2.7 for both solvers, while Figure 2.8 shows the evolution of the unfiltered drag coefficient, C_D .

At early times, the use of the rBiCGStab method leads to drag coefficient results that are a bit more spread than those obtained using the Jacobi method. Let us recall that this problem is very stiff and especially at $t = 0^+$. We observe that iterating through the penalization method (and therefore performing more matrix-vector products, like the Jacobi method does) helps to smooth the drag coefficient behavior. Furthermore, one notes that our results for both methods compare favorably with the reference [34], except at early times, where we observe a faster-decreasing drag than Hejlesen et al. [34].

Moreover, one must keep in mind that the drag coefficient is obtained through a control volume integration, which involves the first moment of vorticity. If the edges of the penalized vorticity, which actually are in the control volume, oscillate (which is very likely in the present very stiff problem), we expect to observe noise within the drag behavior at early times.

As shown in Figure 2.9, the computational costs are noticeably different between the two approaches: Jacobi(1.9) uses $\simeq 70$ matrix-vector products on average while rBiCGStab rapidly falls to an average of 0.55 matrix-vector products on average after the transient ($\text{tol} = 10^{-2}$). rBiCGStab only needs a relatively large number of iterations during the first few time steps, which cover the early times of the impulsive start. Yet, we note that this number remains one order of magnitude below that needed by Jacobi(1.9). The computational gain is thus significant and comes at the cost of a moderate memory overhead. It is indeed here quite limited due to the uses of a penalization-specific subdomain and of the small dimension of the recycled subspace ($l = 2$).

A closer inspection of the solutions inside the body reveals some differences between the two solvers. Figure 2.10 shows the solutions in the vicinity of a plate tip. Both solvers exhibit a spurious vorticity within the plate thickness (which only covers 8 grid points). Yet this spurious vorticity is clearly more intense in the case of Jacobi(1.9); it also gets dampened more slowly.

In other tests (not shown here) performed with a larger time-step (i.e. $C_1 = C_2 = 0.25$), we observe oscillations in the drag coefficient results obtained using the rBiCGStab method. These oscillations indicate that the recycling method is then sensitive to the Lagrangian deformation of the particles which is large due to the corners of the plate, especially at early times. For such stiff test-cases, the time-step should therefore be wisely chosen, or a minimum number of iterations (i.e. resolutions of the Poisson equation) should be enforced to damp any spurious oscillation. Nevertheless,

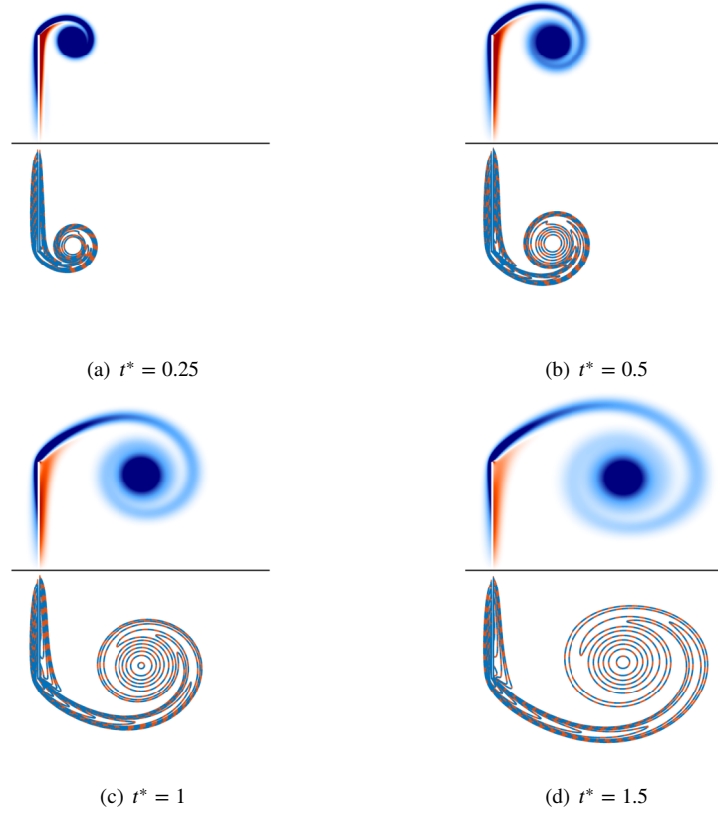


Figure 2.7: **Impulsively started flat plate at $Re = 1000$:** (up) vorticity obtained with $rBiCGStab(2)$; (down) vorticity contour obtained with $rBiCGStab(2)$ (—) and Jacobi(1.9) (—), both with tolerance 10^{-2} .

the flat plate problem indicates that sharp geometrical features and important unsteady effects exacerbate the differences between the two solvers, leading to even higher computational gains.

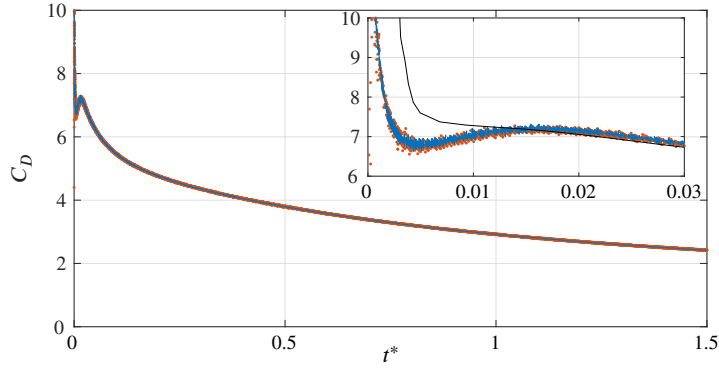


Figure 2.8: **Impulsively started flat plate at $Re = 1000$:** C_D , rBiCGStab(2) (●) and Jacobi(1.9) (—) both with tolerance 10^{-2} . For comparison purpose, we also represent, on the zoom, the drag curve from [34] ($L/T \approx 100$) (—).

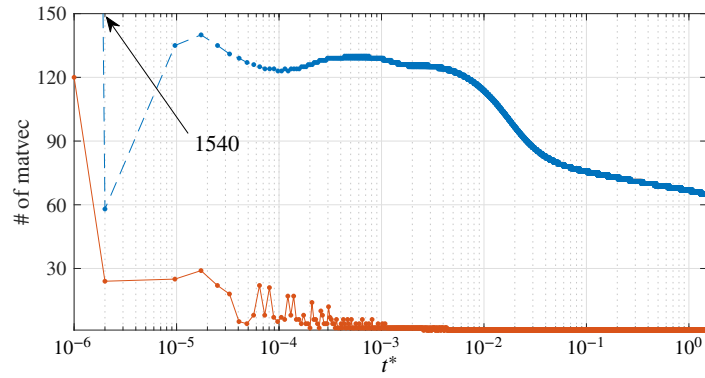


Figure 2.9: **Impulsively started flat plate at $Re = 1000$:** CPU cost to converge to a tolerance of 10^{-2} , Jacobi(1.9) (—) and rBiCGStab(2) (—).

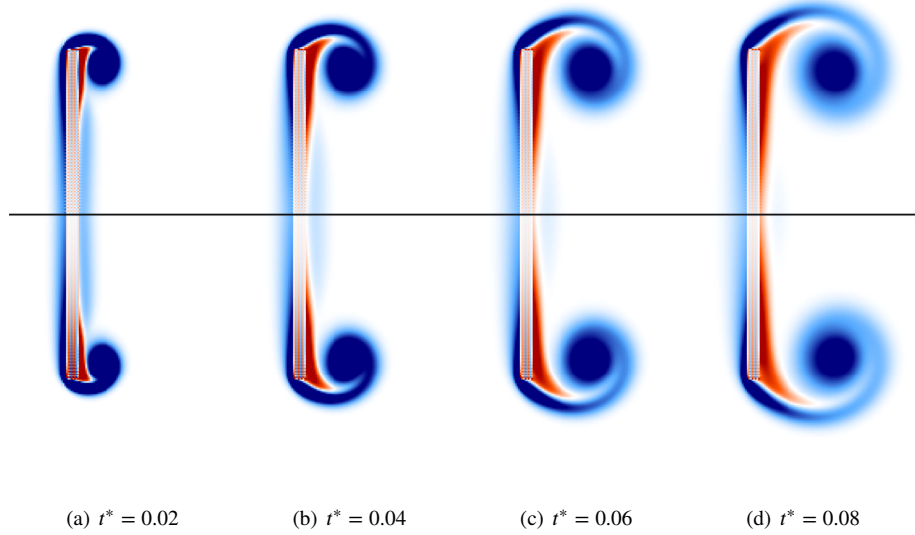


Figure 2.10: **Impulsively started flat plate at $Re = 1000$:** tip vorticity obtained with rBiCGStab(2) (up) and Jacobi(1.9) (down) both with tolerance 10^{-2} .

2.5 Conclusions

Penalization methods in vortex methods are very well suited for the use of recycling iterative solvers. The unknown penalization vorticity is restricted to the boundaries of the body, thus keeping the storage of previous solutions affordable. Solving for this penalization step is also potentially expensive as it entails an elliptic problem: velocities induced by the correction vorticity have to be obtained, which amounts to solving a Poisson equation. Because the right-hand side changes only slightly between two time steps, previous solutions can be efficiently recycled. We developed the approach and motivated the adoption of the rBiCGStab (recycled Bi-Conjugate Gradient Stabilized) as the underlying solver for this application.

The resulting algorithm was assessed on benchmark problems: the flow past an impulsively started cylinder at low and moderate Reynolds numbers and the flow past an impulsively started flat plate. The convergence with respect to the external error is found to be twice as fast as that of the competing approach [34]. Recycling previous solutions also saves many iterations: it divides the cost of iterative penalization by a factor of $\simeq 50$ for the case of the cylinder ($Re = 9500$) and $\simeq 120$ for the case of the flat plate ($Re = 1000$). Furthermore, two previous solutions are found to be sufficient in order to maximize the computational gain and keep the memory overhead low enough to be usable for large problems. However, the higher computational costs encountered during the early phases of the impulsive start hint at the possible revision of this conclusion for flows with strong unsteadiness and/or fast body deformations: more stored previous solutions might be useful to reduce the cost.

In a directly related work, one could consider to study the use of a preconditioner, analyse the reason for a breakdown or take advantage of the recycling of eigenvectors in order to further improve the convergence. We could also develop a criterion in order to use an adaptative recycled subspace's dimension or transport the recycled solution to address accelerating problems. Finally, we note that the generalization of this work to 3D is relatively straightforward since the method considers the domain fields as one dimensional vectors. This is a topic of ongoing work.

Acknowledgments

We would like to acknowledge the helpful discussion with Pr. P. Van Dooren (ICTEAM: Institute of Information and Communication Technologies, Electronics and Applied Mathematics, Université catholique de Louvain).

Fast Immersed Interface Poisson solver for 3D unbounded problems around arbitrary geometries

This chapter is based on the published article *Fast immersed interface Poisson solver for 3D unbounded problems around arbitrary geometries*, T. Gillis, G. Winckelmans, P. Chatelain, Journal of Computational Physics 354 (2018), 403-416.

3.1 Introduction

Solving the Poisson equation is ubiquitous in many computational disciplines such as electromagnetism, fluid mechanics and theoretical physics. Therefore, the problem has been deeply studied and is a recurrent subject within the literature. Still, two major challenges remain: solving the problem in an unbounded framework, and taking into account the presence of a body within the computational domain. These two remaining challenges are widespread and often coupled in many applications, such as computational fluid dynamics applied to the simulation of flows past bodies, like blades and wings.

One may solve the first challenge, i.e. computing unbounded solutions, using two classes of methods. The first kind, based on James [73] and Lackner [74] ideas, solves two problems to determine coherent outer boundary conditions, as proposed

by Marichal et al. [76, 81], McCorquodale et al. [101]. This kind of approach is not very convenient and has an extra cost due to the undetermined boundary condition. In this work, we use a second type of method based on the convolution of a Green's function, sometimes also called kernel function. This convolution is performed using fast summation algorithms: Fast Multipole Methods (FMM) [18–23]), Fast Fourier transforms (Hockney-Eastwood technique) [15–17, 102], or a combination of both [54, 72, 80]. The major drawback of this approach is to handle the singularity of the analytical Green's function at the origin. Two commonly approaches are usually considered: the kernel function is either built through a regularization of the Green's function [16, 17, 23], or by using the Lattice Green's Function (LGF) [21, 22, 54, 72, 79, 80], which take advantage of the mesh to avoid the singularity. This two approaches have already been extensively studied and in this work, we chose the second one for its compatibility with the Immersed Interface Method, see Chapter 1 for more details. The second challenge, i.e. taking immersed bodies into account within an arbitrary grid, can be solved using several methods: Immersed Boundary methods (IB) [52–54, 103], Boundary Element Methods (BEM) [10, 30] or Immersed Interface Methods (IIM) [65, 67, 69, 71, 76]. As in Hosseinverdi and Fasel [69], we choose to take the body into account using the Immersed Interface Method, that offers a proven high order treatment of the boundaries and does not require the introduction of a mollification. Originally developed in a multidimensional way [65] but subject to stability issues, it has been modified to work with one-dimensional dimension splitting [67, 71] and recently applied to 2D problems [67, 76].

Within a regular grid framework, coupling a fast summation based and unbounded solver and immersed bodies is still an open challenge. The BEM already performs such a coupling of the FMM solvers with panel methods in order to compute the additional sources. Let us also mention the work of Hosseinverdi and Fasel [69] that coupled the Immersed Interface Method with a multi-grid Poisson solver (Multi-Scale Multi-Grid, MSMG approach). They also extended the IIM to stretched grid but they are still limited by the determination of the computational domain boundary condition, due to the multi-grid solver. However, none of the above techniques achieves a fast, grid-based solution of the unbounded Poisson problem around a body. For example, the Miller iteration [75], applied by Marichal et al. [76] to the IIM framework, needs to compute the outer boundary conditions iteratively, together with the inner boundary conditions: this tends to increase dramatically the computation cost. Furthermore, fast convolution solvers based on the Lattice Green's Functions [72] were never applied together with the Immersed Interface Method.

This work focuses on coupling the Immersed Interface method with a Lattice Green's Function solver, in order to solve unbounded Poisson problems around bodies. The coupling is performed in an efficient manner through the use of the Sherman-Morrison-Woodbury formula [104–106]. In this chapter, we first present the three different used technologies and further explain how to couple them together (§3.2). Second, we assess the second order of convergence on two benchmarked potential flow applications, and we analyze further the convergence behavior (§3.3). Finally, we close this work by a short conclusion and perspectives for future work.

3.2 Methodology

We consider the unbounded 3D scalar Poisson problem around a body, Ω_b :

$$\nabla^2 u(\mathbf{x}) = f(\mathbf{x}), \quad \text{supp}(f) \in \Omega \setminus \Omega_b, \quad (3.1)$$

where $\mathbf{x} \in \mathbb{R}^3$, Ω is the computational domain which encapsulates the body and the support of the source term, $f(\mathbf{x})$, and the unbounded solution, $u(\mathbf{x})$, decays at least as $1/|\mathbf{x}|$ at infinity. Without loss of generality we assume a Neumann boundary condition on $\partial\Omega_b$. Using a standard 3-point grid discretization for the Laplacian operator, ∇_h^2 , we aim at a second order solver, which uses IIM in order to take the inner boundary condition into account.

We decompose the problem (3.1) into two sub-problems. The first one is the body-free problem, expressed by the unbounded elliptic operator, ∇_h^2 . The second one is the boundary problem, the perturbation of this operator by the body contribution, expressed as an additional source term, f_b . In this section we successively address these two problems and the method used to couple them.

3.2.1 A second order unbounded, body-free, 3D Poisson solver

The straightforward inversion of ∇_h^2 operator is challenging and computationally intensive [73–75, 107]. For the sake of efficiency, we rather follow an approach of grid-based fast convolution of the Green's function with the source term. Because the IIM tools, discussed in Chapter 1, consist in finite differences modifications, we use the lattice Green's function (LGF) associated with the standard second order cross stencil ($3 \times 3 \times 3$). We refer the reader to Chapter 1, and especially §1.2, for a detailed explanation of the Lattice Green's Function.

We use a Hockney-Eastwood FFT-based fast convolution technique [15, 16] to evaluate the unbounded convolution, since it requires the evaluation of the LGF only

once on the domain (it only depends of the size of the domain). We note that a FMM-based approach could also have been used, as proposed in [72].

3.2.2 A second order Immersed Interface method

We here follow the approach and notations as describe in Chapter 1, Linnick and Fasel [67] and Marichal et al. [76, 81]. We invite the reader to refer especially to §1.4 for more information in the second order Immersed Interface Method (IIM) we have used in this work.

However, for the sake of clarity of this section, we recall that the correction brought by the Immersed Interface method rewrites as

$$\nabla_h^2 u - \frac{1}{h^2} J_\alpha(u) = f \quad \Leftrightarrow \quad \nabla_h^2 u = f + f_b(u) \quad , \quad (3.2)$$

where u is the unknown field, f is the source field and the jump coefficient, $J_\alpha = J_\alpha^+ + J_\alpha^-$, depends on the solution itself, see Eqs. (1.20), u and can be interpreted as an additional source term, $f_b(u)$, for the body-free equation. From an operators perspective, one can also identify the IIM as a low-rank perturbation of the body-free Laplacian ∇_h^2 since it only affects mesh points in the vicinity of the body interface.

3.2.3 The combined problem

The IIM scheme, that accounts for inner boundaries, can be considered as a low rank perturbation of the discretized second order Laplacian operator (see §1.4), ∇_h^2 . We propose to couple the body-free problem and the IIM perturbation through the Sherman-Morrison-Woodbury (SMW) formula, that precisely allows to exploit the low-dimensionality of the perturbation. This kind of approach has already been proposed in the context of hybrid grid-particle methods and penalization, by Chatelin and Poncet [45]. The SMW formula [104–106] expresses the inverse of a low-rank perturbed matrix as the exact following decomposition

$$[A + UV]^{-1} = A^{-1} - A^{-1}U [I + VA^{-1}U]^{-1} VA^{-1} \quad . \quad (3.3)$$

We note a slight difference Considering this formula in our particular application, one can associate the unperturbed matrix A to the unbounded and body-free Laplace operator, $A = \nabla_h^2$, while the low rank perturbation is matched to the IIM jumps. To do so, we define the following linear operators

- $E_a \in \mathbb{R}^{n \times k_a}$ is a mapping matrix from the small subspace of discontinuity-affected points (the external vicinity of the body, i.e. the k_a nodes concerned by the IIM corrections) to the global domain of size $n = N_x \times N_y \times N_z$,

- $E_i \in \mathbb{R}^{n \times k_i}$ is a mapping matrix from the primary and secondary interpolation points to the global space,
- $C \in \mathbb{R}^{k_a \times k_i}$ is the matrix that transforms the interpolation points values and the inner boundary condition into IIM jumps, i.e. the Eqs. (1.20) expressed as a linear operator.

Using this definitions, we obtain the following linear system to solve,

$$(\nabla_h^2 + E_a C E_i^T) u = f \quad . \quad (3.4)$$

Applying the SMW formula, Eq. (3.3), to the Immersed Interface unbounded Poisson equation, Eq. (3.4), we obtain two different possible decompositions

$$u = \nabla_h^{-2} \left\{ I_n - E_a C [I_i + E_i^T \nabla_h^{-2} E_a C]^{-1} E_i^T \nabla_h^{-2} \right\} f \quad (3.5a)$$

$$u = \nabla_h^{-2} \left\{ I_n - E_a [I_a + C E_i^T \nabla_h^{-2} E_a]^{-1} C E_i^T \nabla_h^{-2} \right\} f \quad (3.5b)$$

where $I_n \in \mathbb{R}^{n \times n}$, $I_i \in \mathbb{R}^{k_i \times k_i}$ and $I_a \in \mathbb{R}^{k_a \times k_a}$. This allows us to identify two associated boundary problems (and their solutions, x_b and y_b) as the low-rank linear systems to solve,

$$[I_i + E_i^T \nabla_h^{-2} E_a C] x_b = E_i^T \nabla_h^{-2} f \quad (3.7a)$$

$$[I_a + C E_i^T \nabla_h^{-2} E_a] y_b = C E_i^T \nabla_h^{-2} f \quad . \quad (3.7b)$$

At this point, let us note that both problems, Eq. (3.7a) and Eq. (3.7b), are equivalent but the number of unknowns is smaller when considering the problem Eq. (3.7b)². However, in this work, we considered the first formulation Eq. (3.7a), while both decompositions could have been applied. One of the two linear systems Eq. (3.7) is then solved to compute the additional source term due to the body, $f_b = E_a C x_b = E_a y_b$. A GMRes algorithm [86] is typically used for this resolution, since it is a non-symmetric problem, as the IIM operator generates sources, using the interpolation data,

¹We note that our decomposition is slightly different from the one proposed in Chatelin and Poncet [45]. Indeed, they rather use

$$u = \nabla_h^{-2} \left\{ I_n - E_a [C^{-1} + E_i^T \nabla_h^{-2} E_a]^{-1} E_i^T \nabla_h^{-2} \right\} f \quad . \quad (3.6)$$

Our formulation corresponds to solving the system $[I_i + E_i^T \nabla_h^{-2} E_a C] x = b$ and then considering $E_a C x$, while the other formulation is equivalent to solving the system $[C^{-1} + E_i^T \nabla_h^{-2} E_a] \tilde{x} = b$ and considering $E_a \tilde{x}$. The equivalence between both approaches is obtained by considering $\tilde{x} = C x$. Nevertheless, we prefer our approach as it does not involve the inverse of the matrix C .

²We acknowledge the private communication with James Gabbard (MIT) for the second decomposition.

at discontinuity-affected points and not at interpolation points. In practice, a non-zero boundary condition is usually applied at the body surface, i.e. the linear operator C is then affine. In other words, the IIM jumps are given by

$$[E_a C E_i^T] u = [E_a \tilde{C} E_i^T u + E_a \tilde{C}_{bc}] \quad , \quad (3.8)$$

with $\tilde{C} \in \mathbb{R}^{k_a \times k_i}$ the contribution of the interpolation points and $\tilde{C}_{bc} \in \mathbb{R}^{k_a \times 1}$ the boundary condition contribution. The Poisson problem to solve then becomes

$$(\nabla_h^2 + E_a \tilde{C} E_i^T) u = f - E_a \tilde{C}_{bc} \quad . \quad (3.9)$$

The SMW decomposition can be exploited towards computational efficiency. Indeed, the resolution of the boundary problem, restricted to the primary and secondary interpolation points, is more affordable and faster to perform than a resolution of the global problem. Nevertheless, it still involves the body-free problem, which is solved using a fast convolution solver (see §1.2). Rather than using the full-field solver (FFT) used in this work, one might use a sparse solver (FMM) for additional efficiency (thus avoiding the computation of the unused locations). Finally, it is interesting to observe that, unlike many other methods, the decomposition in Eq. (3.7a) works from the perspective of the solution rather than that of the source.

To perform the computations, we use the highly distributed “PPM” library [36, 108]. The Fourier solver is based on 3D FFT’s [16] where data transfers are performed using the PPM library provided mappings. The data movements involved in the IIM interpolation rely on the ghost particles functionality of the PPM library in order to transfer the information from one process to another.

3.3 Validation: potential flows past a body

This section validates our approach and assesses its performance on the solution of the potential flow past a body, Ω_b . As in Poncet [109], we use the Helmholtz decomposition of the velocity,

$$\mathbf{u} = \mathbf{U}_\infty + \nabla \times \boldsymbol{\psi} + \nabla \Phi \quad \text{with} \quad \nabla \cdot \boldsymbol{\psi} = 0 \quad , \quad (3.10)$$

where $\mathbf{U}_\infty = U_\infty \mathbf{e}_z$ represents the free-stream velocity, $\boldsymbol{\psi}$ is the stream function of the velocity induced by the bulk vorticity ($\nabla^2 \boldsymbol{\psi} = -\boldsymbol{\omega}$) and Φ represents the potential required to satisfy the boundary condition $\mathbf{u} \cdot \mathbf{n} = 0$ on $\partial\Omega_b$, where $\mathbf{n}$ is the outgoing normal to $\partial\Omega_b$.

To compute the velocity field, one first computes the bulk stream function, neglecting the presence of the body, by solving $\nabla^2 \boldsymbol{\psi} = -\boldsymbol{\omega}$ in the whole computational domain. Then, the body is taken into account by solving the second problem,

$$\nabla^2 \Phi = 0 \quad \frac{\partial \Phi}{\partial \mathbf{n}} = - [\mathbf{U}_\infty + \nabla \times \boldsymbol{\psi}] \cdot \mathbf{n} \text{ on } \partial\Omega_b . \quad (3.11)$$

In the context of the present potential flows validation, the bulk vorticity is nil, hence $\boldsymbol{\psi} = 0$, without loss of generality. Therefore, we are left with the following problem

$$\nabla^2 \Phi = 0 \quad \text{with} \quad \frac{\partial \Phi}{\partial \mathbf{n}} = - [\mathbf{U}_\infty] \cdot \mathbf{n} \text{ on } \partial\Omega_b . \quad (3.12)$$

For our application, we define the local residual for the boundary problem, Eq. (3.7a), as

$$r_b = E_i^T \nabla_h^{-2} [f_\infty - E_a \tilde{C}_{bc}] - [I_i + E_i^T \nabla_h^{-2} E_b \tilde{C}] x_b , \quad (3.13)$$

where x_b is the solution of the boundary problem, whose support is limited to the vicinity of the body $\mathcal{I}_b$, i.e. the primary and secondary interpolation points. The global residual is then taken, in dimensionless form, as

$$\tilde{r}_b = \frac{1}{U_\infty L^{5/2}} \sqrt{\sum_{\mathcal{I}_b} |r_b|^2 h^3} , \quad (3.14)$$

with L the length of the computational domain and h the spatial step of the mesh. The tolerance on $\tilde{r}_b$ is chosen to be 10^{-14} in order to avoid any influence of a non-fully converged solution of the boundary problem.

Furthermore, we define the local error as $e(\mathbf{x}) = \phi(\mathbf{x}) - \phi_{exact}(\mathbf{x})$, in the bulk, and the associated dimensionless 2-norm and infinite norm, respectively as,

$$E_2 = \frac{1}{U_\infty L^{5/2}} \sqrt{\sum_{\Omega \setminus \Omega_b} e^2(\mathbf{x}) h^3} \quad \text{and} \quad E_\infty = \frac{1}{U_\infty L} \max_{\Omega \setminus \Omega_b} |e(\mathbf{x})| . \quad (3.15)$$

We finally define the local convergence orders, r_∞ and r_2 as

$$[r_q]_i = \log([E_q]_i / [E_q]_{i-1}) / \log(N_{i-1} / N_i) , \quad (3.16)$$

with $q = \infty$ and $q = 2$, respectively and i , the resolution level (here, $h_i = \frac{L}{N_i}$ and $N_{i+1} = 2N_i$).

3.3.1 Potential flow past a sphere

We first consider the problem of the potential flow past a sphere. The analytical solution of this flow is given by

$$\Phi_{exact} = U_{\infty} z \frac{R^3}{2 |\mathbf{x}|^3} , \quad (3.17)$$

where R is the radius. We use a box $[L \times L \times L]$, with $L = 2.5 R$ and N grid points in each direction (in total N^3 unknowns). The global and local convergence slopes are given in Table 3.1 and Figure 3.1, while the solution is visualized in Figure 3.3 in terms of streamlines, i.e. tangent lines to $\mathbf{U}_{\infty} + \nabla \phi$.

Our method achieves the expected second order convergence for both the 2-norm and the infinite norm. We stress that the velocity inside the body remains nil, as imposed by the IIM, without any a posteriori modification. Therefore, there is no need to cancel it and no spurious flow appears across the boundary. Figure 3.2 also shows the number of iterations performed while solving the boundary problem using the GMRes algorithm. We observe that increasing the number of points does not require many additional iterations since we already resolve the body boundary accurately, even with $N = 128$ ($R/h = [51.2, 102.4, 204.8, 409.6]$ for $N = [128, 256, 512, 1024]$, respectively).

Figure 3.4 shows the distribution of the time spent in each part of the key steps: initialization, Poisson equation and other operations (communication and memory movements). We observe that we spend a non-negligible time to initialize the solver. This time is used to detect the intersections with the level set function and pre-compute the interpolation coefficients, i.e. the C matrix from Eq. (3.5a). The remaining time is mostly spent in the body-free Poisson solver. One also notices on Figure 3.4 that the percentage of time spent in communication is decreasing with the size of the problem. This is the expected behavior of such a method when one subjects it to a strong scalability test. We use a constant number of CPU's (1024 Intel Haswell's) and increase the computational domain size: each processing entity thus has a decreasing communication load, in relation to the computational work.

3.3.2 Potential flow past an ellipsoid

Following the same approach as before, we now consider the potential flow around an ellipsoid of semi-axes $[a, b, c]$. The analytical solution of such a problem is given

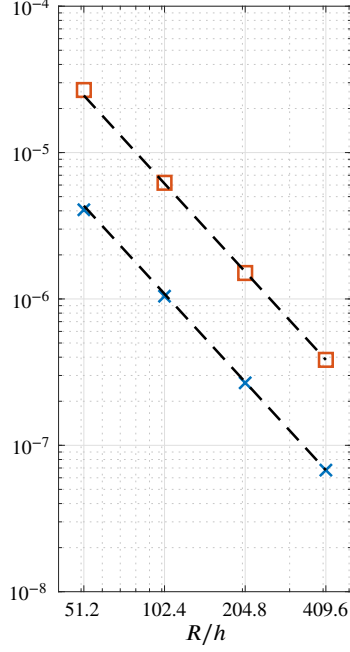


Figure 3.1: **Potential flow past a sphere:** Grid convergence of the solution to the analytic expression, E_1 (□), E_2 (×) and $\mathcal{O}(h^2)$ (—).

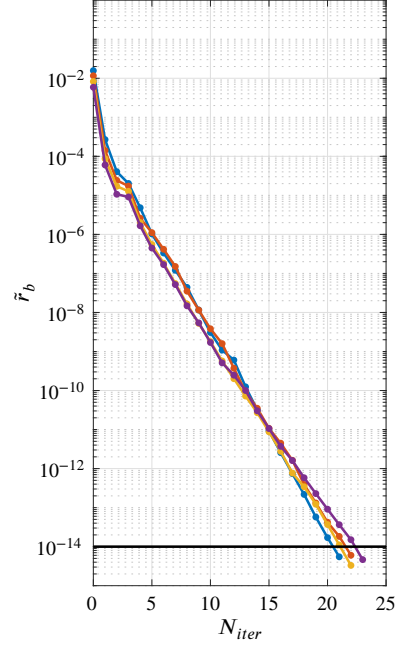


Figure 3.2: **Potential flow past a sphere:** Number of iteration needed to solve the boundary problem, Eq. (3.7a), up to the chosen tolerance, 10^{-14} with $R/h = [51.2, 102.4, 204.8, 409.6]$ (—, —, —, —).

R/h	E_∞	r_∞	E_2	r_2	t_i [s]	t_{bf} [s]	t_o [s]
51.2	$2.68e-05$		$4.07e-06$				
102.4	$6.22e-06$	2.11	$1.04e-06$	1.96	0.83	4.50	1.63
204.8	$1.50e-06$	2.05	$2.67e-07$	1.97	4.82	26.81	2.18
409.6	$3.84e-07$	1.97	$6.77e-08$	1.98	61.76	239.26	4.86

Table 3.1: **Potential flow past a sphere:** convergence rate for E_∞ and E_2 and time spent for initialization (t_i), body-free problem (t_{bf}) and other operations (t_o) in seconds while using 1024 Intel Haswell CPU's.

by

$$\Phi_{exact} = \frac{abc U_\infty z}{3 - abc R_D(a^2, b^2, c^2)} \left[\frac{R_D(\epsilon - 1, \epsilon - k^2, \epsilon)}{(c^2 - a^2)^{3/2}} \right], \quad (3.18)$$

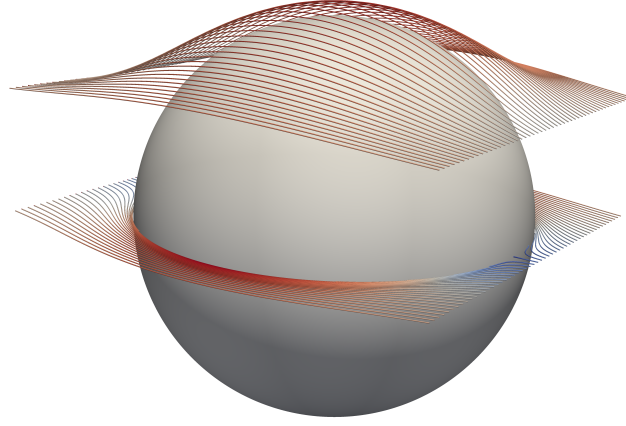


Figure 3.3: **Potential flow past a sphere:** Streamlines view colored by the norm of the velocity, $\nabla\Phi + \mathbf{U}_\infty$ ($N = 512$).

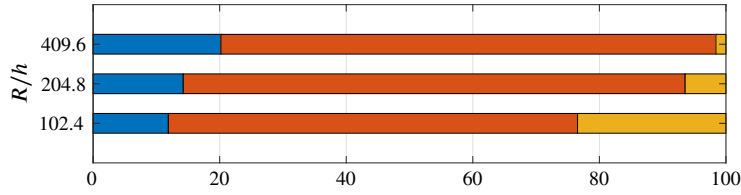


Figure 3.4: **Potential flow past a sphere:** % of the computational time elapsed in initialization (■), Poisson solver (■) and communication (■); timings for $R/h = [102.4, 204.8, 409.6]$.

with ϵ , k and R_D defined in Appendix A.3, where we present this analytical solution. We choose the computational domain to be $[L \times L \times L]$ with an ellipsoid given by $[a = 0.2L, b = 0.3L, c = 0.4L]$ and a tolerance for the boundary problem of 10^{-14} . Since $a < b < c$, the smallest radius of curvature, on the surface of the ellipsoid is $R_{min} = \frac{a^2}{c} = 0.1L$.

The convergence slope of this study case is given in Figure 3.5 and Table 3.2, while the solution is shown in Figure 3.7. We conclude that our method achieves again the expected second order convergence both for the 2-norm and infinite norm. The number of iterations to perform in order to converge to the chosen tolerance is almost constant (see Figure 3.6). Again, this is due to the fact that the problem is already well resolved ($R_{min}/h = [12.8, 25.6, 51.2, 102.4]$ for $N = [128, 256, 512, 1024]$),

respectively). In §3.3.3, we further investigate the evolution of the number of iterations with respect to the resolution of the obstacle boundary.

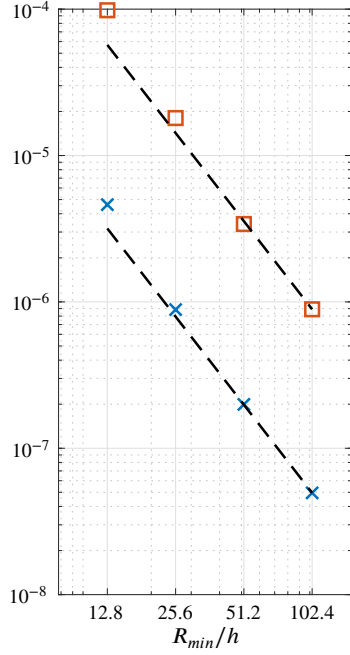


Figure 3.5: **Potential flow past an ellipsoid:** Grid convergence of the solution to the analytic expression, E_i (□), E_2 (×) and $\mathcal{O}(h^2)$ (---).

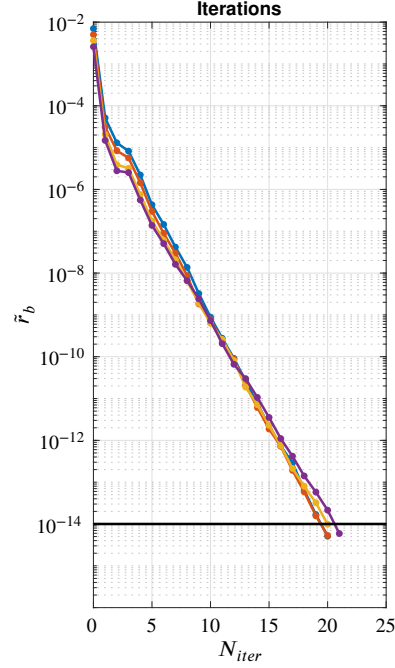


Figure 3.6: **Potential flow past an ellipsoid:** Number of iteration needed to solve the boundary problem, Eq. (3.7a), up to the chosen tolerance, 10^{-14} with $R/h = [51.2, 102.4, 204.8, 409.6]$ (●, ●, ●, ●).

Figure 3.8 shows the time spent in each part of the key steps. Again, the time spent in the body-free problem is prevailing, which is the expected behavior of the method.

3.3.3 Sensitivity analysis: R_{min}/h

The number of iterations and the evolution of the error varies with the number of points used to represent the shape and its finer scales in particular. We now consider an ellipsoid of semi-axes $[a = \beta L, b = 0.3 L, c = 0.4 L]$ and keep the computational domain constant. Figure 3.9 shows the evolution of the error and the number of it-

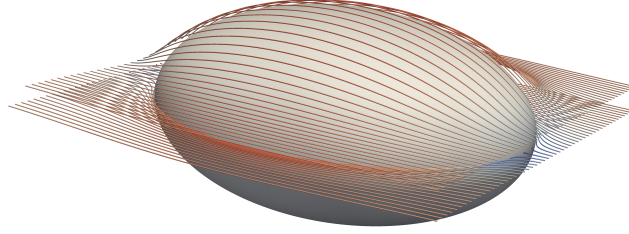


Figure 3.7: **Potential flow past an ellipsoid:** Streamlines view colored by the norm of the velocity, $\nabla\Phi + \mathbf{U}_\infty$ ($R_{min}/h = 51.2$).

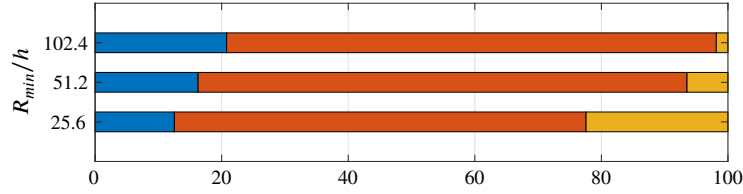


Figure 3.8: **Potential flow past an ellipsoid:** % of the computational time elapsed in initialization (■), Poisson solver (■) and communication (■); timings for $R_{min}/h = [25.6, 51.2, 102.4]$.

R_{min}/h	E_∞	r_∞	E_2	r_2	t_i [s]	t_{bf} [s]	t_o [s]
12.8	$9.82e-05$		$4.62e-06$				
25.6	$1.80e-05$	2.45	$8.86e-07$	2.38	0.81	4.21	1.45
51.2	$3.40e-06$	2.40	$1.99e-07$	2.15	5.22	24.77	2.07
102.4	$8.90e-07$	1.94	$4.96e-08$	2.01	58.65	218.38	5.19

Table 3.2: **Potential flow past an ellipsoid:** convergence rate for E_∞ and E_2 and time spent for initialization (t_i), body-free problem (t_{bf}) and other operations (t_o) in seconds while using 1024 Intel Haswell CPU's.

erations to perform ($\text{tol} = 10^{-14}$) for various $R_{min}/h = \frac{\beta^2 L}{0.4 h}$, where β varies and h remains constant.

We identify three different behaviors in Figure 3.9. When the geometry is over-resolved ($R_{min}/h \geq 30$), the 2-norm exhibits a plateau. However, the decreasing infinite norm shows that an increasing resolution still improves the solution at the tip of the ellipsoid.

For a well-resolved geometry ($30 > R_{min}/h \geq 10$), both errors are decreasing when one increases the resolution. Furthermore, the number of iterations to perform stays constant ($\simeq 20$). At coarser resolutions, the tip of the ellipsoid is not well-captured, this affects the conditioning of the system and the number of iterations requested to converge increases as the resolution decreases.

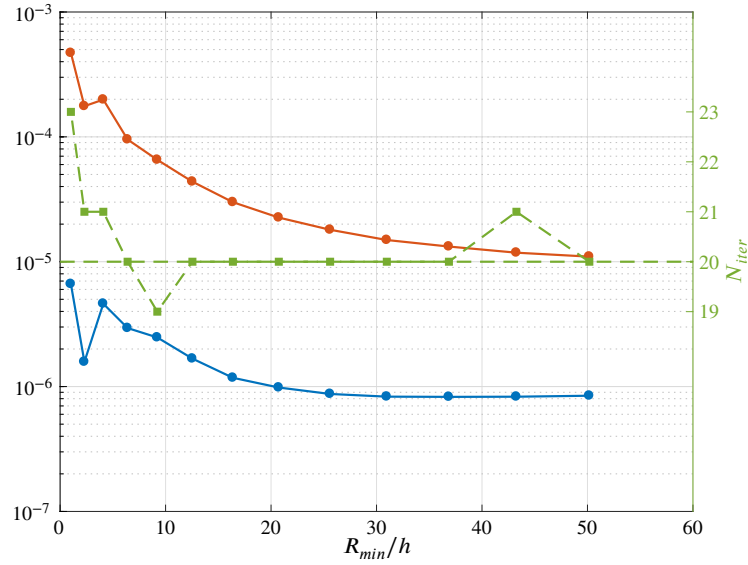


Figure 3.9: **Potential flow past an ellipsoid:** Analysis of the sensibility for various R_{min}/h coefficients (with constant h): the 2-norm (—●—), infinite norm (—●—) and the number of iterations required to reach $\text{tol} = 10^{-14}$ (—■—).

3.3.4 Sensitivity analysis of the boundary problem tolerance

A tolerance parameter has to be chosen for the GMRes iterations when solving the boundary problem. In this section, we study its influence to the expected second order convergence of the solution, as shown on Table 3.3. It shows that one should iterate enough to solve the boundary problem, Eq. (3.7a), to guarantee a converging solution.

The previous examples (§3.3.1 and §3.3.2) used a tolerance of 10^{-14} , but a tolerance of 10^{-10} would have been sufficient. We also notice that, at this tolerance, the GMRes convergence is fast and it only costs 13 to 14 iterations to converge a problem of 1024^3 unknowns.

tol	R/h	N_{iter}	r_∞	r_2
$1e-06$	204.8	5	non convergent	non convergent
	409.6	5		
$1e-08$	204.8	9	0.40	1.68
	409.6	9		
$1e-10$	204.8	13	1.99	1.98
	409.6	14		
$1e-12$	204.8	17	1.97	1.98
	409.6	18		
$1e-14$	204.8	22	1.97	1.98
	409.6	23		

Table 3.3: **Potential flow past a sphere**: Influence of the tolerance in the iterative process with respect to the convergence behavior.

3.4 Conclusions

In this chapter, we have presented a novel way to use a fast and efficient unbounded elliptic solver in order to take an inner obstacle into account. The method proposed is based on a combination between a Lattice Green's Function convolution carried out by the Hockney-Eastwood method or an FMM-based algorithm and the Immersed Interface technique. This coupling is performed through the Sherman-Morrison-Woodbury (SMW) formula, which involves a low-rank system to solve, i.e. the boundary problem. This formulation reduces the problem to the vicinity of the body and therefore to a boundary problem to solve instead of a volume problem. Through the SMW formula and the boundary problem, we find the additional source term which, combined with the unbounded source and the application of the body-free Laplacian, produces the correct solution satisfying the boundary condition. Through the SMW formula, our solver preserves the efficiency of the fast convolution-based algorithms and uses the immersed interface method to achieve a second order convergence, up to the boundary.

We prove the second order accuracy of our method on two benchmark cases (the potential flows past a sphere and an ellipsoid). We then study the robustness of our method with respect to the resolution, R_{min}/h and the number of iterations performed within the iterative process for a chosen tolerance. We show that keeping $R_{min}/h \geq 10$ will lead to a well-resolved solution and that the boundary problem solution has to be accurate enough to guarantee the convergence.

This work, so far limited to the Poisson equation, could be extended to the Helmholtz problem. Moreover, we stress that our method is compatible with any solver based on the second order standard stencil for the Laplacian. The method is applicable

to a wide range of problems involving an elliptic equation around inner bodies, and it goes beyond the applications in computational fluid dynamics.

A 2D Immersed Interface Vortex Particle-Mesh method

This chapter is based on the published article *A 2D Immersed Interface Vortex Particle-Mesh method*, T. Gillis, Y. Marichal, G. Winckelmans and P. Chatelain, *Journal of Computational Physics* 394 (2019).

4.1 Introduction

Vortex methods are gaining popularity for many advection-dominated simulations in flows where the vorticity is compact such as wake studies at high Reynolds numbers and external flow aerodynamics [36, 39, 43, 44]. However, taking the presence of a body into account remains a challenge. The current state of the art in the treatment of inner bodies for flow problems, on regular grids, is split between the *discrete* and *continuous forcing* approaches, a differentiation introduced by Mittal and Iaccarino [51]. To the best of the authors' knowledge, none of the proposed approaches, transposed to the VPM framework, offers a sharp and accurate treatment of the immersed boundary, while also preserving the temporal convergence order. Indeed, the above approaches either smear the immersed object interface, as in the iterative Brinkman penalization technique, which degrades the spatial accuracy at the wall or they split the diffusion operator, which limits the temporal convergence to first order, as in the VPM-BEM approach.

Nevertheless, some recent works have extended the Immersed Interface grid-based approach to the VPM approach. Marichal et al. [76] have developed an IIM-based unbounded Poisson solver around complex geometries and they have extended the particle-mesh interpolation to the Immersed Interface tool in [81]. In Chapter 3, we

have also presented an efficient unbounded FFT-based Poisson solver for inner boundaries using IIM, that decomposes the problem using the Sherman-Morrison-Woodbury formula [45, 104, 105].

These works have paved the way for an Immersed Interface Vortex Particle-Mesh (IIVPM) solver for incompressible flows around inner bodies. Here we aim at a sharp and second order IIVPM solver which additionally conserves the temporal convergence order. This chapter is structured as follows: in §4.2, we first briefly recall the Vortex Particle-Mesh method as a whole, including the specific aspects of the particle-mesh interpolation and of the Biot-Savart law solver. Then, we introduce the Immersed Interface techniques and we expose their use in the present context. Finally, we validate the resulting IIVPM method in §4.3 and assess its performances on the benchmark case of an impulsively started cylinder in §4.4. A conclusion of the presented work and future perspectives are presented in §4.5.

4.2 Methodology

This work presents a combination of the Vortex Particle-Mesh (VPM) method with Immersed Interface (II) tools, see Chapter 1, in order to take an immersed body into account. Hereunder, we first briefly describe the VPM approach (§4.2.1) and then we present the IIVPM methodology as a whole in §4.2.2 and then, step by step: the Poisson solver in §4.2.3, the handling of the diffusion in §4.2.4 and the particle-mesh interpolation in §4.2.5 and §4.2.6.

4.2.1 The 2D Vortex Particle-Mesh method

We briefly present the Vortex Particle-Mesh method as used in this work, based on the velocity ($\mathbf{u}$)-vorticity ($\omega = \omega \mathbf{e}_z = \nabla \wedge \mathbf{u}$) formulation of the 2D Navier-Stokes equations for an incompressible flow ($\nabla \cdot \mathbf{u} = 0$):

$$\frac{D\omega}{Dt} = \nu \nabla^2 \omega, \quad (4.1)$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$ denotes the Lagrangian derivative and ν is the kinematic viscosity, see [2, 3] and references therein. We here have $\mathbf{u} = \mathbf{U}_\infty + \mathbf{u}_\omega$, where $\mathbf{u}_\omega$ is recovered from the vorticity field through a Poisson equation

$$\nabla^2 \psi_\omega = -\omega \quad \mathbf{u}_\omega = \nabla \wedge (\psi \mathbf{e}_z). \quad (4.2)$$

A Vortex Particle-Mesh (VPM) method discretizes the vorticity field by means of particles, characterized by a position $\mathbf{x}_p$, a material surface S_p and a strength

$\alpha_p = \int_{S_p} \omega d\mathbf{x}$. An integration of Eq. (4.1) over a particle surface, through a mid-point quadrature with the assumption of the particle discretization, produces the following system

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x}_p \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \mathbf{u}_p \\ \nu (\nabla^2 \omega)_p S_p \end{bmatrix} . \quad (4.3)$$

In the framework of VPM methods, the right-hand side of Eq. (4.3) is evaluated on the mesh, therefore improving the computational efficiency. The particle-mesh interpolation typically relies on high order interpolation formulas [11, 110]. Once the particle quantities have been interpolated onto the mesh, one can use standard finite differences to evaluate the differential operators on the right hand side of Eq. (4.3). The Poisson problem of Eq. (4.2) can also be solved using the mesh, by means of highly efficient FFT-based solvers [17, 36]. Afterwards, the right hand side is interpolated back on the particles to integrate Eq. (4.3). The time integration scheme, on the particles, is a standard Runge-Kutta scheme; the low-storage RK3 scheme proposed by Williamson [111] is used in this work.

The time step is first stability-constrained by the diffusion; this is embodied by the Fourier number, i.e. $r = \frac{\nu \Delta t}{h^2} < r_{max}$, where r_{max} is given by the chosen time and space discretization scheme. The particle treatment of the convection imposes an accuracy condition instead, based on the Lagrangian deformation, usually measured using two criterions

$$\Delta t \|\mathbf{D}\|_1 \leq \text{LCFL}_D \quad \text{and} \quad \Delta t \|\omega\|_1 \leq \text{LCFL}_\omega , \quad (4.4)$$

where $\mathbf{D} = \frac{1}{2} [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]$ is the strain tensor and $\text{LCFL}_D \simeq \text{LCFL}_\omega \simeq 0.15 \dots 0.3$ are typical values. To prevent particle clustering and depletion, due to the Lagrangian deformation, the VPM method relies on a procedure called remeshing [2, 3], which consists in the periodic redistribution of the particle set onto the mesh. Finally, we refer to [36] for an extensive description of the VPM method and its implementation for large parallel architectures.

4.2.2 Time step sequence

In this section we present a global view of the successive computations to be performed, for the evaluation of the right hand side of Eq. (4.3), during one time step. It can also be seen as a sub-step when using multi-step time integration schemes (such as Runge-Kutta schemes).

1. **Poisson - §4.2.3:** Given the bulk vorticity field, ω_{bulk}^n , we compute the no-through-flow velocity, $\mathbf{u}^n = \nabla \wedge (\psi^n \mathbf{e}_z)$, by solving the Poisson equation

$$\nabla^2 \psi^n = -\omega^n, \quad (4.5)$$

such that $\mathbf{u} \cdot \mathbf{n} \Big|_{\delta\Omega_b} = 0$, i.e. the no-through-flow condition is satisfied.

2. **Wall vorticity - §4.2.4:** Given $\mathbf{u}^n$, we compute the ω_{wall}^n for each control point, imposing $\mathbf{u}_{wall} = 0$, i.e we compute the vorticity value at the wall in order to impose the no-slip condition.
3. **Diffusion - §4.2.4:** Given ω^n and ω_{wall}^n , we compute the diffusion term, $\nabla^2 \omega^n$ that takes the wall vorticity into account.
4. **Extension - §4.2.5:** Given $\mathbf{u}^n$ and $\nabla^2 \omega^n$, we extend those fields into the body using a Hamilton-Jacobi extension.
5. **M2P - §4.2.5:** Given the extended fields, we perform the mesh to particles interpolation of the computed rhs of Eq. (4.3) and assign a value to the ghost particles (including ghost particles).
6. **Advection and diffusion:** We integrate in time the system of Eq. (4.3) on the particles.
7. **P2M - §4.2.6:** We perform the particle to mesh interpolation (and eventually a redistribution of the particles, if required), and we also erase the spurious extended fields inside the body.

Again, we would like to stress that this sequence actually constitutes an evaluation of the right hand side of Eq. (4.3). Indeed, this method is not a time splitting algorithm, which consequently allows to reach a global order as high as the consistency of the discretization of the right hand side, Eq. (4.3), here at third order.

4.2.3 Poisson equation

The 2D Poisson equation is solved on $\psi = \psi \mathbf{e}_z$ using the bulk vorticity $\omega_{bulk} = \omega_{bulk} \mathbf{e}_z$ and a no-through flow condition on the body, i.e. a Dirichlet boundary condition,

$$\nabla^2 \psi = -\omega_{bulk} \quad \text{such that} \quad \psi \Big|_{\delta\Omega_b} = \psi_b. \quad (4.6)$$

In 2D, this is not a simply connected domain and one needs to close the system by imposing an additional constraint. We impose the total circulation of the flow, Γ , which

sets the constant ψ_b . To solve the Poisson equation, Eq. (4.6), we use the efficient Immersed Interface Poisson solver proposed in Chapter 3. It decomposes the problem using the Sherman-Morrison-Woodbury formula [104, 105] and actually solves the following equation,

$$\psi = -(\nabla_h^2 + E_a C E_i^T)^{-1} \omega_{bulk} \quad (4.7a)$$

$$= -\nabla_h^{-2} \left\{ I_n - E_a C [I_i + E_i^T \nabla_h^{-2} E_a C]^{-1} E_i^T \nabla_h^{-2} \right\} \omega_{bulk} \quad (4.7b)$$

where E_a and E_i stand for the domain restriction operators to the discontinuity affected points and primary interpolation points, respectively. I_n is the identity matrix of size n (n is the size of the set of the interpolation points) and C represents the jumps computed using Eqs. (1.26). This kind of approach has been initially proposed in the context of penalization problems by Chatelin and Poncet [45] and is detailed in the context of the Poisson solver Chapter 3.

The operator $[\nabla_h^2]^{-1}$ is applied through a FFT-based Poisson solver [16] using the 2D lattice Green's function as a convolution kernel (for more details, see Chapter 1, Martinsson and Rodin [79] and Gillman and Martinsson [22]). Let us also observe that the circulation constraint, Γ , is actually imposed through a constraint on the IIM-generated correction, $[E_a C E_i^T] \psi$, which amounts to a vortex sheet; its strength has to respect $\Gamma_{body} = \Gamma - \Gamma_{bulk}$. The number of iterations required to solve the sub-system is reduced thanks to the use of recycling based on the previous solutions, as proposed by Amritkar et al. [96] and in Chapter 2, among others. Finally, the no-through flow enforcing velocity field is obtained by differentiating the stream function and injecting the Dirichlet boundary condition, $\psi = \psi_b$, into Eq. (1.26a).

4.2.4 Vorticity sheet treatment

The no-through flow enforcing velocity field still has a spurious slip velocity at the wall. This will generate a vorticity sheet, not representable on the grid points since it exists only at the body surface. Following Lighthill's approach [64], this vorticity sheet is the newly created vorticity necessary to satisfy the no-slip condition at the wall and therefore is part of the flow.

A classical way to do so, proposed by Marichal [14] in the framework of IIM and broadly used in vortex methods [10, 63, 64], is to first compute the no-through flow enforcing velocity field and use it to advect particles. The diffusion is performed using a zero-flux boundary condition, hence discarding the generation of vorticity at the wall. One then evaluates the error committed by these two operations, i.e. the spurious residual slip velocity which corresponds to the flux of vorticity that has to enter the

flow by another diffusion step. This methodology has two major drawbacks. First, it splits the diffusion operation and therefore bounds the time convergence to $\mathcal{O}(\Delta t)$; second, in order to ensure a high consistency in the slip velocity evaluation, it requires the solution of two Poisson problems per substep.

The present work uses a different technique that merges the two operations previously required into a single diffusion operation, carried out with a Dirichlet boundary condition. We compute the value of the vorticity field at the wall, using the Immersed Interface framework, in order to impose the no-slip behavior as

$$\omega_{wall} = \frac{\partial u_y}{\partial x} \Big|_{wall} - \frac{\partial u_x}{\partial y} \Big|_{wall} , \quad (4.8)$$

where the derivatives are computed by injecting $u_x|_{wall} = u_y|_{wall} = 0$ into Eqs. (1.26) for each control point at the body surface.

The diffusion field, $\nabla^2 \omega$, is computed in the fluid domain, and corrected along the boundary by each control point, injecting the value of the wall vorticity, ω_{wall} , into Eq. (1.26a), see Figure 4.1

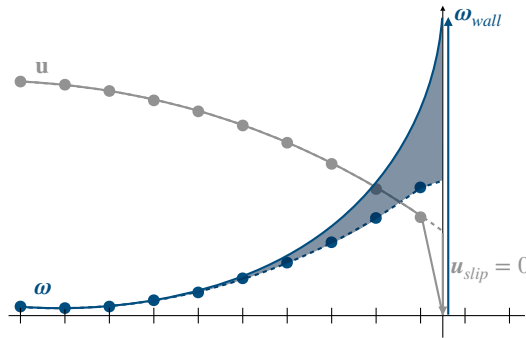


Figure 4.1: **Wall vorticity computation** - schematic view of the boundary layer region: the no-through flow velocity field, $\mathbf{u}$, (—●—) and the bulk vorticity, ω (—●—). The imposition of the no-slip velocity at the wall, $\mathbf{u}_{slip} = 0$, generates a wall vorticity value used as a boundary condition in the vorticity evolution, e.g. the diffusion illustrated here (■).

The proposed method differs significantly from the approach used in BEM [10, 30, 63]: it also diffuses a newly generated vorticity, obtained from the no-slip condition, into the flow. Yet, it uses the no-slip boundary condition to obtain ω_{wall} , and does the diffusion step using that value, together with the convection step. This differs from doing the diffusion step with zero flux, together with the convection step, and

then, afterwards, computing the vorticity flux to enforce the no-slip condition, using a Neumann boundary condition, and diffusing it into the flow.

4.2.5 Mesh to Particle (M2P) interpolation and extension inside the body

The mesh to particles interpolations use the M'_4 interpolation kernel [11, 110], which is equivalent to a stencil of width $4h$, see Figure 4.2. In the neighborhood of the interface, one could use a decentered interpolation in order to use only the information in the flow; or use extended field values to reproduce the effect of a decentered scheme. To keep the interpolation process unchanged, we rather choose the second option by extending the field inside the body, as initially proposed by Marichal et al. [81].

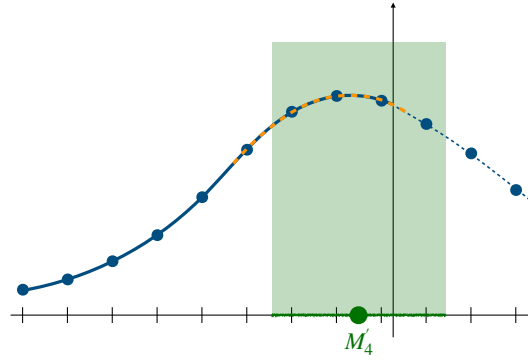


Figure 4.2: **Hamilton-Jacobi extension** - schematic view of the interface vicinity: the vorticity field, ω , (—●—) has to be interpolated on the particle (●). We could construct a third order polynomial (---) or extend the field inside the body (---), hence filling the M'_4 kernel support (■).

For this work, we identified two different extensions of the considered field inside the body, which generalizes the work proposed by Aslam [112] to the Immersed Interface framework. For the sake of completeness, we present them below. Both approaches consider the body level-set function, ϕ , i.e. a signed distance function such that $\phi < 0$ describes the region inside the body. For each point inside the body, we define the inner normal

$$\mathbf{n}_{in} = -\frac{\nabla\phi}{|\nabla\phi|}, \quad (4.9)$$

and, in order to extend the field in a small band around the body interface only (since we only need a few ghost points inside the body), the cutting function, C ,

$$C = \begin{cases} 0 & \text{if } \phi \geq 0 \\ \frac{1}{1 + \exp\left(10^3 \frac{|\phi - h_c|}{L}\right)} & \text{if } \phi < 0 \end{cases}, \quad (4.10)$$

where L is the characteristic length of the body and h_c is the cutting distance. The extrapolated field, f , is obtained as the stationary solution of the following system

$$\frac{\partial}{\partial \tau} \begin{bmatrix} f \\ f' \end{bmatrix} + C \begin{bmatrix} \mathbf{n}_{in} \cdot \nabla & -1 \\ 0 & \mathbf{n}_{in} \cdot \nabla \end{bmatrix} \begin{bmatrix} f \\ f' \end{bmatrix} = 0, \quad (4.11)$$

where the unknowns are extended with the normal derivative of f , $f' \triangleq \mathbf{n}_{in} \cdot \nabla f$. This system is solved in pseudo time τ until convergence of the solution, using an up-wind scheme coupled with a RK2 integration, a classical approach in levelset methods.

The two identified approaches differs in the way they take the boundary into account and in the way they compute the normal derivative. The first approach decomposes the problem of the extrapolation into two steps. First, knowing a field f outside the body, we compute its value and its normal derivative, $f' \triangleq \mathbf{n}_{in} \cdot \nabla f$, at the boundary using Eqs. (1.26), as in [81]. Then, once all the needed information is known at the boundary, we extend it from the boundary into the body using a Hamilton-Jacobi system, coupled with the Immersed Interface technique and the pre-computed surface information to evaluate the right hand side of Eq. (4.11). This approach has two drawbacks: first, one needs to use the IIM to correct the evaluation at each iteration and second, in order to use a second order up-wind discretization of the gradient, more than one grid point on each side of the interface need to be affected by the Immersed Interface correction.

The second approach is much simpler, as we do not have to take the interface explicitly into account while integrating the system Eq. (4.11). Indeed, before integrating the Hamilton-Jacobi system, we pre-compute f and $f' = \mathbf{n}_{in} \cdot \nabla f$ in the fluid-side vicinity of the wall, using centered finite differences, hence second order, and correcting the discontinuity affected points with the Immersed Interface jumps. Since we do not want to consider any boundary condition for the extended field, we extrapolate the value and the derivatives (using third order extrapolation) up to the boundary, using Eqs. (1.35), which is in agreement with the global accuracy of the scheme, $\mathcal{O}(h^2)$. This way, we build a second order evaluation of f' and still keep the number of discontinuity affected points to one, on each side of the interface.

In this work, we chose to use a linear extension in the normal direction. A higher degree extension (e.g. using $\frac{\partial}{\partial \tau} f''$ in Eq. (4.11)) could in principle lead to an increased accuracy. This is however not the case in practice, due to the cross derivatives not taken into account and the use of higher order polynomials for the extrapolation, which become poorly conditioned when considering the extension of stiff fields.

The proposed Hamilton-Jacobi extension is then used on the velocity and diffusion fields to extend them into the body and provide enough grid-based information for a M2P operation near the boundary.

4.2.6 P2M and ghost particles

The Particle-to-Mesh (P2M) operation follows the same approach as for the M2P operations. Ghost particles are constructed in a narrow band in the vicinity of the interface, $\simeq 5h$, in order to provide a particle support for the standard P2M interpolation kernels at grid points close to the interface. The way to create a ghost particle relies on the following observation: the field that needs to be interpolated is the diffused strength of the particles and there is no easy and cheap way to create ghost particles from the flow particle status. Therefore, we propose to build ghost particles using the same extension of the vorticity inside the body, during the M2P operation. Ghost particles are then advected and diffused along with the flow particles, using the extension of the velocity, vorticity and diffusion fields (see §4.2.5).

4.3 Validation

In order to validate the spatial and temporal convergences of the proposed method, we consider the diffusion of a Gaussian vorticity field, centered at $\mathbf{x} = \mathbf{x}_c$:

$$\omega_G(x, y, t) = \frac{\Gamma}{\pi} \frac{1}{(\sigma^2 + 4\nu t)} \exp\left(-\frac{r^2}{(\sigma^2 + 4\nu t)}\right), \quad (4.12)$$

with $r = |\mathbf{x} - \mathbf{x}_c|$. The velocity, $\mathbf{u} = (u, v)$, and diffusion, $\nabla^2 \omega$, are analytically obtained:

$$u = -\frac{\Gamma}{2\pi} \frac{y}{r^2} \left[1 - \exp\left(-\frac{r^2}{(\sigma^2 + 4\nu t)}\right) \right] \quad (4.13)$$

$$v = \frac{\Gamma}{2\pi} \frac{x}{r^2} \left[1 - \exp\left(-\frac{r^2}{(\sigma^2 + 4\nu t)}\right) \right] \quad (4.14)$$

$$u_\theta = \frac{\Gamma}{2\pi r} \left[1 - \exp\left(-\frac{r^2}{(\sigma^2 + 4\nu t)}\right) \right] \quad (4.15)$$

$$\nabla^2 \omega = \frac{4\Gamma}{2\pi} \frac{1}{\sigma^4} \left[\frac{r^2}{\sigma^2} - 1 \right] \exp\left(-\frac{r^2}{(\sigma^2 + 4\nu t)}\right) \quad (4.16)$$

To validate the proposed method, we clip that Gaussian vorticity field, by removing its core, $r < R = \sigma$, and we replace it with a rotating cylinder. The flow lies outside, $r \geq R = \sigma$, and has a vorticity field given by $\omega = \omega_G$. The removed core effect is ensured by the adequate rotating velocity of the cylinder, i.e. the one that reproduces that of the diffusing Gaussian vortex at $r = R$: $u_{\theta \text{ wall}} = u_\theta(r = R)$. The corresponding circulation of the body is then

$$\Gamma_{body} = \Gamma \left[1 - \exp\left(-\frac{R^2}{(\sigma^2 + 4\nu t)}\right) \right] . \quad (4.17)$$

The analytical solution of the diffusing Gaussian vorticity thus remains unchanged and is still valid in the fluid domain. The Reynolds number is defined using the characteristic velocity, $U = \frac{\Gamma}{2\pi R}$, and the diameter, $D = 2R$, as

$$\text{Re} = \frac{U D}{\nu} = \frac{1}{\pi} \frac{\Gamma}{\nu} . \quad (4.18)$$

Finally, we also define the dimensionless time, $t^* = t \Gamma / R^2$.

We run three different validation tests on this benchmark, using the analytical expression of the vorticity given by Eq. (4.12), as an initial condition, and the analytical wall velocity given by Eq. (4.13) and Eq. (4.14) at every time. The first two results prove the third order time convergence of the diffusion alone and of the full method (without remesh), respectively; the last one shows the second order spatial convergence of the full method (without remesh). For every run, we use a very narrow tolerance for the Poisson iteration, 10^{-14} , 6 vectors in the recycling subspace and $\text{Re} = 100$.

We study the convergence of the 2- and ∞ -norms of the error. Given a field of reference, the dimensionless norms are defined as

$$E_2 = \frac{R}{\Gamma} \sqrt{\sum_{i,j} \left(\omega_{ij}^{ref} - \omega_{ij} \right)^2 h^2} \quad E_\infty = \frac{R^2}{\Gamma} \max_{i,j} |\omega_{ij}^{ref} - \omega_{ij}| \quad . \quad (4.19)$$

Considering two successive resolutions, (Δ_k) and (Δ_{k+1}) , and their associated errors, we define the local convergence rate, r_2 and r_∞ , as

$$r_2 = \frac{\log [E_2^k / E_2^{k+1}]}{\log [\Delta_k / \Delta_{k+1}]} \quad r_\infty = \frac{\log [E_\infty^k / E_\infty^{k+1}]}{\log [\Delta_k / \Delta_{k+1}]} \quad . \quad (4.20)$$

4.3.1 Temporal convergence

Diffusion

The diffusion problem presented earlier is purely axisymmetric and hence we first do not advect the particles, in order to highlight the diffusion treatment presented in §4.2.2. Other operations of the proposed time-step remain unchanged. This special treatment for the advection avoids the errors produced by the particle-mesh interpolation and particle distortion. The runs are made using $R/h = 51.2$ (i.e. a grid of 512×512 for a domain of $10R \times 10R$) and for three Fourier numbers: $r \triangleq \nu \Delta t / h^2 = \{0.2 ; 0.1 ; 0.05\}$. We use these runs in a self-convergence test, as the error is computed with respect to a reference solution computed very accurately, using a much smaller Fourier number $r = 0.005$. Indeed, the spatial discretization produces an error much higher than the time integration and, therefore, does not allow to compare to the analytical vorticity solution. We compare the obtained results at two dimensionless times, $T_1^* = 0.2 \pi \text{Re} (h/R)^2 \simeq 0.024$ and $T_2^* = 4T_1^* \simeq 0.096$. The results of the conver-

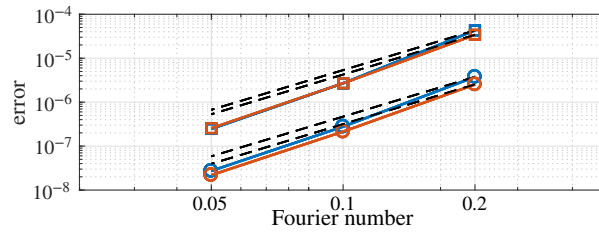


Figure 4.3: **Temporal convergence** - diffusion: $E_\infty(T_1^*)$: ■ and T_2^* : ■), $E_2(T_1^*)$: ○ and T_2^* : ○) and $\mathcal{O}(\Delta t^3)$ (---).

T_1^*					
Fourier number	time step	E_∞	r_∞	E_2	r_2
0.2	1	4.29e-05		3.75e-06	
0.1	2	2.69e-06	4.0	2.77e-07	3.8
0.05	4	2.46e-07	3.5	2.73e-08	3.3
T_2^*					
Fourier number	time step	E_∞	r_∞	E_2	r_2
0.2	4	3.40e-05		2.53e-06	
0.1	8	2.69e-06	3.7	2.14e-07	3.6
0.05	16	2.55e-07	3.4	2.19e-08	3.3

Table 4.1: **Temporal convergence** - diffusion: E_∞ , E_2 and the corresponding convergence order r_∞ and r_2 at T_1^* and T_2^* .

gence are presented in Figure 4.3 and Table 4.1 and exhibit a third order convergence, which validates the proposed approach for the diffusion. Indeed, the treatment of the diffusion term on the particles does not include any splitting and hence does not affect the third order Runge-Kutta integration scheme.

Diffusion and advection

In this section, we add the advection of the particles in order to highlight the influence of the particle-mesh interpolation and the particles distortion on the accuracy of the method. We then use every sub-step of the algorithm described in §4.2.2. However, we do not remesh the particles during the simulation. We use a time-step fixed by the Fourier number, $r \triangleq \nu \Delta t / h^2 = \{0.2 ; 0.1 ; 0.05\}$ and the same mesh resolution as in the previous section. Again, this analysis is based on a self-convergence approach as the spatial discretization produces much higher errors than the time integration. The results are presented in Figure 4.4 and in Table 4.4. The third order convergence is reached for T_1^* . The change in convergence rate observed for T_2^* is mainly due to the Lagrangian deformation that begins to dominate. It could also be partially attributed to the non-linear behavior of the extension methodology, which depends on both Δx and Δt and hence deteriorates the self-convergence order.

4.3.2 Spatial convergence of the method

To study the spatial convergence of the method (without remesh), we use the same Gaussian benchmark and consider this time the convergence to the analytical solution. The convergence is run at constant Fourier number, $r = 0.15$ and using three

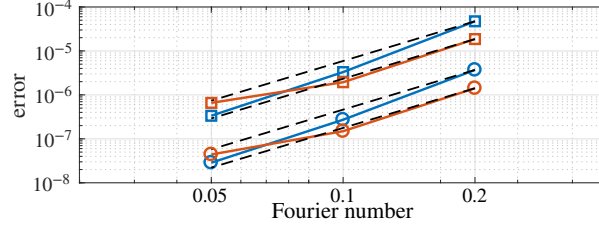


Figure 4.4: **Temporal convergence** - advection and diffusion: $E_\infty(T_1^* : \text{---}\square\text{---})$ and $T_2^* : \text{---}\square\text{---}$), $E_2(T_1^* : \text{---}\circ\text{---})$ and $T_2^* : \text{---}\circ\text{---}$) and $\mathcal{O}(\Delta t^3)$ (---).

T_1^*					
Fourier number	time step	E_∞	r_∞	E_2	r_2
0.2	1	4.73e-05		3.70e-06	
0.1	2	3.30e-06	3.8	2.72e-07	3.8
0.05	4	3.35e-07	3.3	2.86e-08	3.2
T_2^*					
Fourier number	time step	E_∞	r_∞	E_2	r_2
0.2	4	1.86e-05		1.42e-06	
0.1	8	1.94e-06	3.3	1.49e-07	3.2
0.05	16	6.58e-07	1.6	4.41e-08	1.8

Table 4.2: **Temporal convergence** - advection and diffusion: E_∞ , E_2 and the corresponding convergence order r_∞ and r_2 at T_1^* and T_2^* .

T^*					
R/h	time step	E_∞	r_∞	E_2	r_2
25.6	4	1.15e-03		2.66e-04	
51.2	16	2.00e-04	2.5	6.47e-05	2.0
102.4	64	7.54e-05	1.4	1.52e-05	2.1

Table 4.3: **Spatial convergence** - E_∞ , E_2 and the corresponding convergence order r_∞ and r_2 at T^* .

different grid sizes $R/h = [25.6 ; 51.2 ; 102.4]$. We measure the convergence at $T^* = 0.6 \pi \text{Re}(h_0/R)^2 \simeq 0.29$, when the Lagrangian deformation is still acceptable and no remeshing operation is required. The results are shown in Figure 4.5 and they present the expected second order convergence for both the 2- and ∞ -norms.

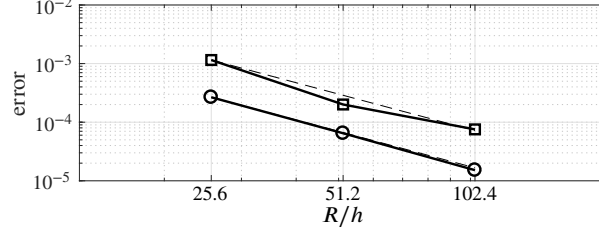


Figure 4.5: Spatial convergence - E_∞ (—□—), E_2 (—○—) and $\mathcal{O}(h^2)$ (---).

4.4 The impulsively started flow past a cylinder

To illustrate the presented approach, we present its application to the benchmark of the impulsively started flow past a cylinder, and we compare our results to several published works. We perform the comparison between the grid resolutions of several runs by quantifying the resolution in the boundary layer. Therefore, the scaling concerning the spatial resolution h relatively to the characteristic boundary layer thickness δ is here defined as

$$N_\delta \triangleq \frac{\delta}{h} = \frac{1}{\sqrt{\text{Re}_D}} \left(\frac{D}{h} \right), \quad (4.21)$$

with $\text{Re}_D = \frac{U_\infty D}{\nu}$ and $\delta \propto \frac{D}{\sqrt{\text{Re}_D}}$, for the flow near the stagnation point, as also used in Mimeau et al. [32]. Moreover, the accuracy of numerical methods in regions with sharp gradients is driven by the mesh Reynolds number based on the vorticity:

$$\text{Re}_{\omega,h} = \frac{|\omega| h^2}{\nu}. \quad (4.22)$$

So, a second “figure of merit” to assess the quality, within the boundary layer, of simulations relatively to each other can be defined as

$$Q \triangleq \frac{\omega_{wall} h^2}{\nu} = \left(\frac{U_\infty}{D} \sqrt{\text{Re}_D} \right) \frac{h^2}{\nu} = \frac{\sqrt{\text{Re}_D}}{N_\delta^2} = \text{Re}_D^{3/2} \left(\frac{h}{D} \right)^2, \quad (4.23)$$

where we have used that the wall vorticity is proportional to $\frac{U_\infty}{\delta}$, thus $\omega_{wall} \propto \frac{U_\infty}{D} \sqrt{\text{Re}_D}$.

We also define the following dimensionless quantities,

$$t^* = \frac{t U_\infty}{D} \quad C_D = \frac{F_x}{\frac{1}{2} D U_\infty^2} \quad \omega^* = \frac{\omega D}{U_\infty}, \quad (4.24)$$

where F_x is the drag computed using a control volume approach by Noca [113], as also proposed by Mimeau et al. [32],

$$\mathbf{F} = -\frac{d}{dt} \int_{\Omega_c} \mathbf{u} d\mathbf{x} + \int_{\partial\Omega_c} \boldsymbol{\gamma} \cdot \mathbf{n} d\mathbf{x} , \quad (4.25)$$

where Ω_c is the control volume, $\mathbf{n}$ is normal to $\partial\Omega_c$ and

$$\begin{aligned} \boldsymbol{\gamma} = & \frac{1}{2} |\mathbf{u}|^2 \mathbf{I} - \frac{1}{N-1} \mathbf{u} (\mathbf{x} \times \boldsymbol{\omega}) + \frac{1}{N-1} \boldsymbol{\omega} (\mathbf{x} \times \mathbf{u}) \\ & - \frac{1}{N-1} \left[\left(\mathbf{x} \cdot \frac{\partial \mathbf{u}}{\partial t} \right) \mathbf{I} - \mathbf{x} \frac{\partial \mathbf{u}}{\partial t} \right] \\ & + \frac{1}{N-1} [[\mathbf{x} \cdot (\nabla \cdot \mathbf{T})] \mathbf{I} - \mathbf{x} (\nabla \cdot \mathbf{T})] + \mathbf{T} . \end{aligned} \quad (4.26)$$

in which N is the spatial dimension (2 or 3), $\mathbf{I}$ is the unit tensor and $\mathbf{T} = \mu [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]$ is the viscous stress tensor. This approach is well suited for the vorticity formulation as it does not involve the pressure field.

Finally, we define $\theta \in [-\pi ; \pi]$, the angle along the cylinder surface such that the stagnation point is at $\theta = 0$.

To alleviate the computational cost, we use a subdomain, $[N_x^{sub} \times N_y^{sub}]$ that fully includes the immersed body and where the Poisson sub-system is solved with a relative tolerance of 10^{-6} and 6 recycling solutions. The cylinder is centered at $(0.75D ; 1.25D)$ and we use the following accuracy parameters: $C_{LCFL,D} = 0.1$, $C_{LCFL,\omega} = 0.2$ and $r = 0.3$. The remaining parameters are presented in Table 4.4.

Re	R/h	N_δ	Q	L_x	L_y
550	102.4	8.73	0.31	$5D$	$2.5D$
3000	204.8	7.48	0.98	$5D$	$2.5D$
9500	819.2	16.81	0.34	$2.5D$	$2.5D$
9500	409.6	8.40	1.38	$5D$	$2.5D$
40000	819.2	8.19	2.98	$2.5D$	$2.5D$

Table 4.4: **Impulsively started flow past a cylinder** - setup used for different Reynolds numbers. The computational domain spans $[L_x ; L_y]$.

4.4.1 Impulsively started flow past a cylinder at $Re = 550$

The flow around a cylinder at $Re = 550$ is presented in Figure 4.8. The simulation parameters are presented in Table 4.4. The obtained drag coefficient is shown in Figure 4.6, where we match the results already obtained by Koumoutsakos and Leonard

[63] and Marichal [14]. We also compare favorably to the theoretical drag curve [100], valid only for short times,

$$C_D = 4 \sqrt{\frac{\pi}{(\text{Re}_D t^*)}} + 2\pi \left(9 - \frac{15}{\sqrt{\pi}} \right) \frac{1}{\text{Re}_D} . \quad (4.27)$$

In Figure 4.7, we compare the vorticity at the wall used in our method with the one obtained by Koumoutsakos and Leonard [63] and the one provided by Marichal [14]. We observe a behavior consistent with the sharp treatment of the Immersed Interface techniques. We are indeed closer to the results of Marichal [14] than to those of Koumoutsakos and Leonard [63]. Nevertheless, we predict slightly higher wall vorticity values for $\theta \simeq \pm 0.9\pi \simeq \pm 80^\circ$ compared to the former. Finally, the small oscillations present in the wall vorticity are due to the varying distance between the surface and the first point in the bulk in the X and Y directions: this distance governs the computation of the wall quantities, as described in §4.2.4. These oscillations tend to disappear at higher resolutions, as they are driven by the Q quality criterion, see Figs. 4.7, 4.10 and 4.13.

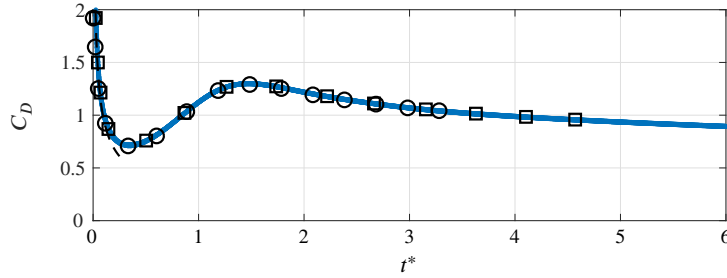


Figure 4.6: **Drag coefficient at $\text{Re}_D = 550$** - Present approach (—●—), theoretical drag curve (---), Koumoutsakos and Leonard [63] (□) and Marichal [14] (○)

4.4.2 Impulsively started flow past a cylinder at $\text{Re}_D = 3000$

The flow around a cylinder at $\text{Re}_D = 3000$ is presented in Figure 4.11. The simulation parameters are presented in Table 4.4. In Figure 4.9, we compare the obtained drag coefficient with the results of Koumoutsakos and Leonard [63]. The agreement with the reference is excellent, in particular the capture of the peak around $t^* \simeq 2$. The dimensionless wall vorticity is exposed in Figure 4.10 and compared with the results from Koumoutsakos and Leonard [63]. It can be seen to have more slight oscillations than at $\text{Re}_D = 550$, which are due to the fact that we kept N_δ constant from one run

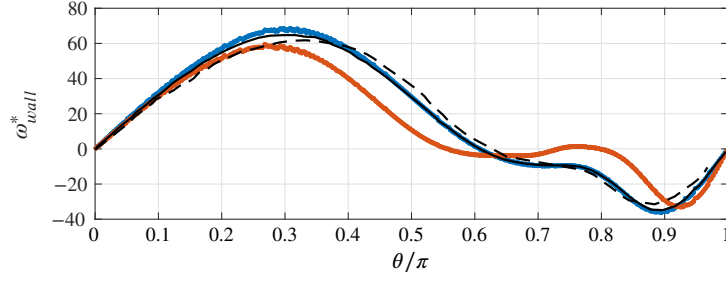


Figure 4.7: **Wall vorticity at $\text{Re}_D = 550$** - Present approach ($t^* = 1$: $\bullet$, $t^* = 3$: $\circ$), Koumoutsakos and Leonard [63] ($t^* = 1$: $-\cdot-$) and Marichal [14] ($t^* = 1$: —).

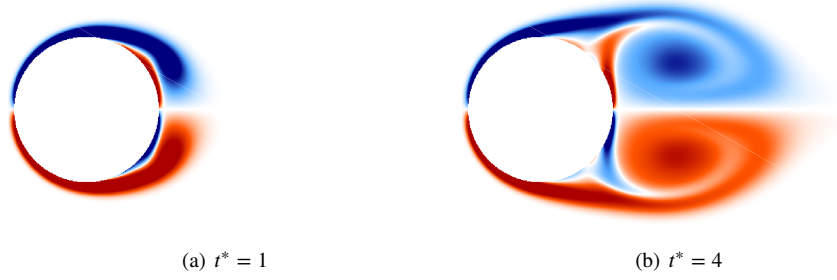


Figure 4.8: **Vorticity contours at $\text{Re}_D = 550$**

to another, hence decreasing the accuracy at the wall (as we increase the value of Q). Moreover, we note that our approach predicts higher values around wall vorticity peaks than the latest reference; this reflects the better sharpness of the IIVPM method at the wall.

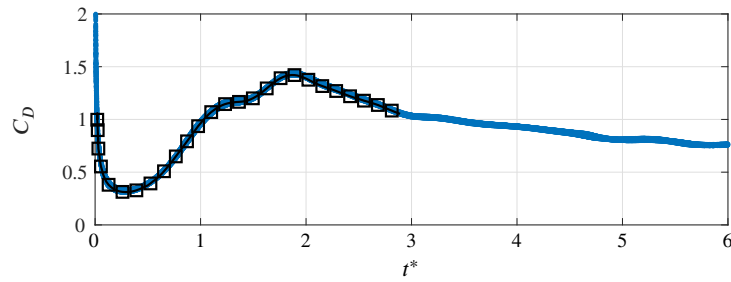


Figure 4.9: **Drag coefficient at $\text{Re}_D = 3000$** - Present approach ($\bullet$) and Koumoutsakos and Leonard [63] ($\square$).

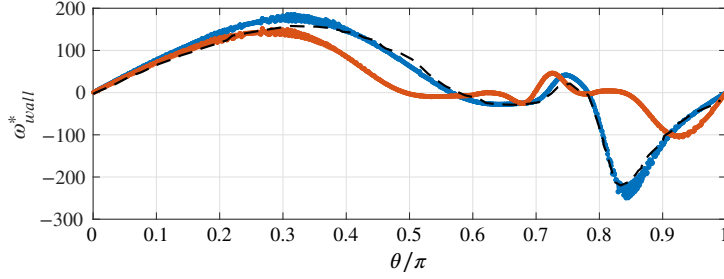


Figure 4.10: **Wall vorticity at $\text{Re}_D = 3000$** - Present approach ($t^* = 1$: —•—, $t^* = 3$: —•—) and Koumoutsakos and Leonard [63] ($t^* = 1$: - -).

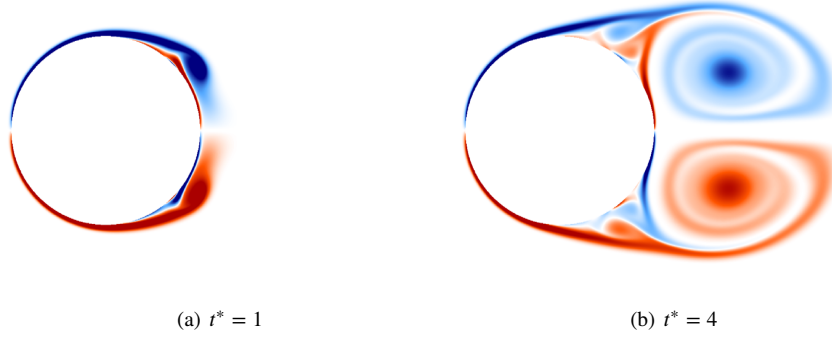


Figure 4.11: **Vorticity contours at $\text{Re}_D = 3000$**

4.4.3 Impulsively started flow past a cylinder at $\text{Re}_D = 9500$

The flow around a cylinder at $\text{Re}_D = 9500$ is presented in Figure 4.14. For this test case, we have used two resolutions to highlight the accuracy of the IIVPM approach and the influence of the accuracy parameter, Q . The coarser computation has been performed with $N_\delta \simeq 8$, while the finer used twice that resolution, $N_\delta \simeq 16$ (i.e. $R/h \simeq 410$ and $R/h \simeq 820$ respectively, see Table 4.4).

The obtained drag coefficient is shown in Figure 4.12, where we observe excellent agreement between the two resolutions. It proves that, for a value of $N_h = 8.4$ (and $Q = 1.38$), the drag coefficient is already quite converged, even when considering high Reynolds number flows. This coefficient can also be seen to globally agree with the results of Rossinelli et al. [114], obtained using an effective number of points per radius 8 times higher ($R/h \simeq 3280$ and $N_\delta \simeq 64$ in their case). The more noticeable differences lie in the drag oscillations around $t^* \simeq 2$, which appear to be sharply cap-

tured by the present method, even at low resolution. This observation, together with the convergence test of the previous sections, lead us to the claim that the differences with Rossinelli et al. [114], in particular for the dynamics of the shear layer, is due to the dissipation/damping behavior of the penalization method¹. This in turn influences the generation of the structures shed in the wake and has a noticeable impact on the drag coefficient.

Finally, Figure 4.13 shows the wall vorticity for both resolutions. Again we observe higher values around the peak compared to Koumoutsakos and Leonard [63] at $t^* = 1$, which is also observable in the drag coefficient behavior. Furthermore, the influence of the wall accuracy parameter, Q , is noticeable as increasing the accuracy at the wall leads to significantly smoother wall vorticity values, as expected.

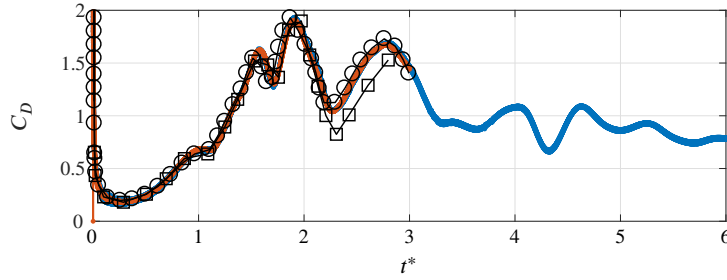


Figure 4.12: **Drag coefficient at $\text{Re}_D = 9500$** - Present approach ($N_\delta = 8.4$: —●—, $N_\delta = 16.8$: —●—), Koumoutsakos and Leonard [63] (—■—) and Rossinelli et al. [114] (—●—).

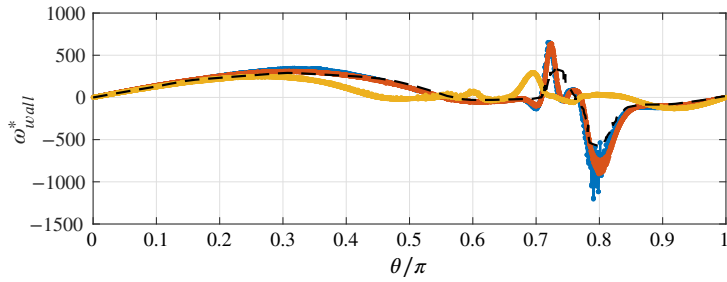
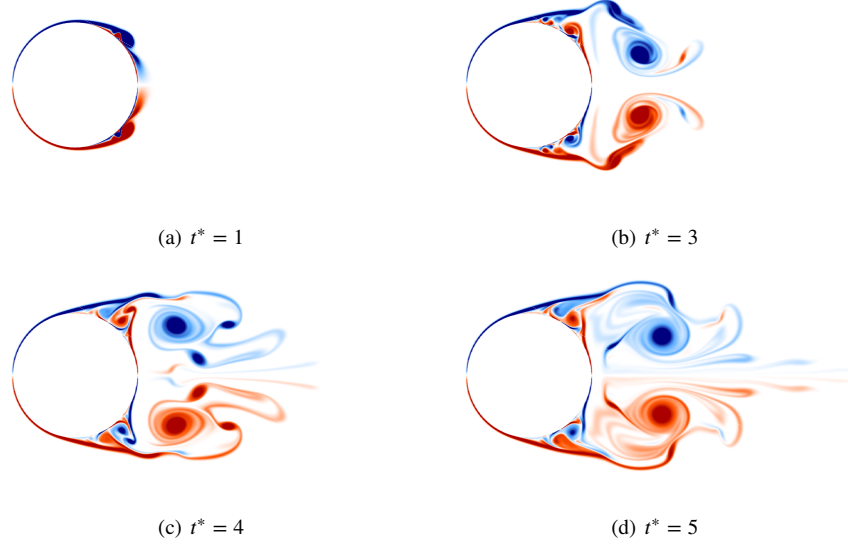


Figure 4.13: **Wall vorticity at $\text{Re}_D = 9500$** - Present approach at $t^* = 1$ ($N_\delta = 8.4$: —●—, $N_\delta = 16.8$: —●—); at $t^* = 3$ ($N_\delta = 16.81$: —●—) and Koumoutsakos and Leonard [63] at $t^* = 1$ (— - —).

¹There is a factor 2 in the dimensionless time between [114] and the present work since we have used the diameter and they have used the radius in the time definition.

Figure 4.14: Vorticity contours at $\text{Re}_D = 9500$

4.4.4 Impulsively started flow past a cylinder at $\text{Re}_D = 40000$

In order to stress our approach, the flow around a cylinder at $\text{Re}_D = 40000$ is presented in Figure 4.17. We decided to keep the scale N_δ constant to a value comparable to the one used in the previous runs, therefore degrading even more the wall accuracy ($Q \simeq 3$). The other simulation parameters are presented in Table 4.4.

The obtained drag coefficient is shown in Figure 4.15 and compared to that of Rossinelli et al. [114]. We note a qualitative agreement with this reference up to $t^* \simeq 1.2$, which corresponds to the detachment of the vorticity patch located at $\theta \simeq \pm 135^\circ$ in Figure 4.4.4. A zoom of the vorticity field, shown in Figure 4.18, focuses on the region of successive separations; the already detached shear layer exhibits azimuthal oscillations which lead to vortex formations, more intense pressure gradients and eventually more separations. Its behavior is extremely sensitive to resolution effects (N_δ is 16 times higher in their results than in ours), to the accuracy treatment at the wall (Q) and to the performance of the numerical scheme, which explains the observed difference. Finally, to stress the importance of the wall accuracy criterion, Q , we present the wall vorticity in Figure 4.16. It can be observed to present oscillations especially in regions of vortex formation.

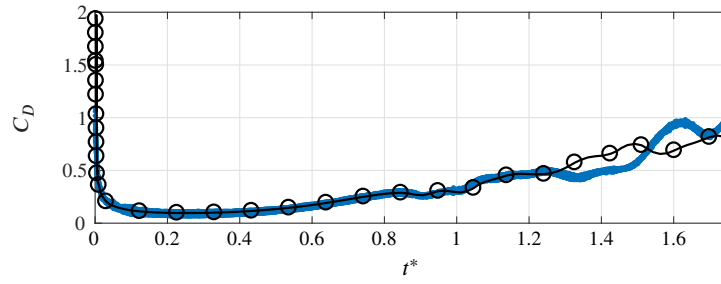


Figure 4.15: **Drag coefficient at $\text{Re}_D = 40000$** - The present approach (—●—) and Rossinelli et al. [114] (—○—).

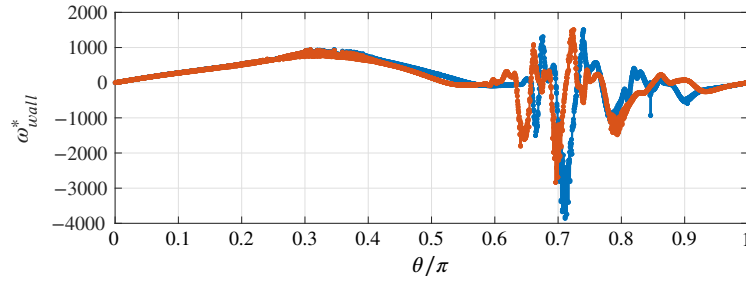


Figure 4.16: **Wall vorticity at $\text{Re}_D = 40000$** - The present approach ($t^* = 1.0$: —●—, $t^* = 1.5$: —●—).

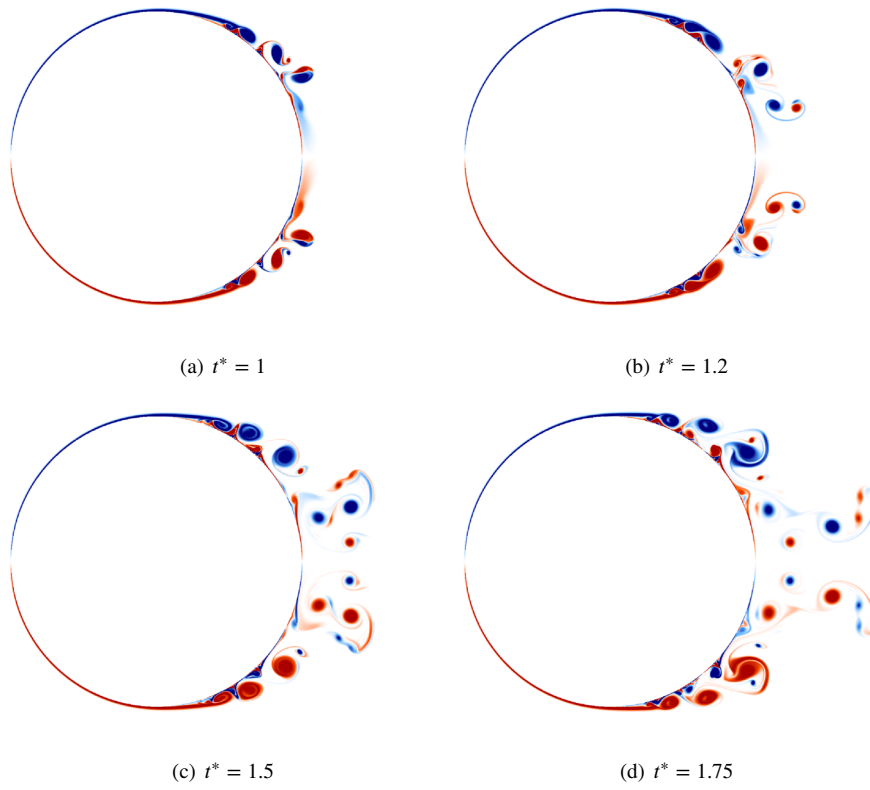
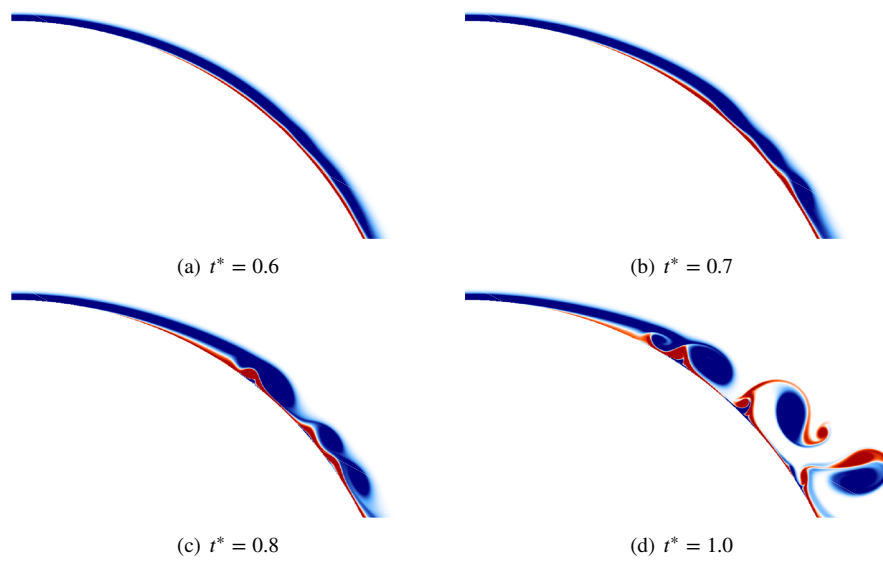


Figure 4.17: Vorticity contours at $Re_D = 40000$

Figure 4.18: Vorticity contours $Re_D = 40000$

4.5 Conclusions

In this chapter, we presented a sharp and accurate approach to take an inner obstacle into account in the Vortex Particle-Mesh (VPM) framework. The proposed method relies on the Immersed Interface tool to handle interface discontinuities on the grid and uses ghost particles to impose the body interface from the particle point of view. Every step of the standard VPM methodology is modified using the Immersed Interface approach: the Poisson equation is solved using the decomposition into a global and a local problem, as proposed in Chapter 3, while the diffusion is computed using IIM-modified stencils along with a Dirichlet boundary condition, computed from the no-slip condition. The diffusion and vorticity fields are then extrapolated inside the body, using an IIM-modified Hamilton-Jacobi extrapolation, in order to ensure the accuracy of the mesh-particle interpolation. After advection and diffusion on the particles (including the ghost particles in the body), we interpolate them back onto the mesh. A remeshing technique is also implemented and applied whenever required.

We prove the second order convergence of the proposed methodology, up to the wall, and the conservation of the temporal convergence order, unlike any already proposed Immersed Interface approach in the VPM framework. Moreover, the present approach only entails one Poisson equation per sub-step, unlike a flux-based (or Lighthill) approach, which further reduces the computational cost. We assess the computational efficiency on the impulsively started cylinder benchmark for various Reynolds numbers (from $Re = 550$ to $Re = 40000$) and compare favourably with other references. We also introduce a “figure of merit” of immersed boundary methods, Q , that quantifies the accuracy of treatment of an interface, and in particular the computation of the wall vorticity. We illustrate that scaling the grid resolution with $Re^{1/2}$ is not sufficient to insure an equally accurate treatment of the boundary, thus a scaling with $Re^{3/4}$ would be more appropriate, keeping a constant Q (typically $Q \leq 1$). We finally show that the present IIVPM method is able to reproduce results from Rossinelli et al. [114], at high Reynolds number ($Re = 9500$), using an eight times coarser grid resolution at the boundary, hence highlighting the gain of accuracy brought by the sharp interface treatment.

This work, so far, has been focused on a second-order accurate method, in the bulk and at the wall. However, it could readily be extended to higher order by enlarging the width of the primary and secondary interpolation layers around the interface. Moreover, the present approach is extendable to 3D without significant difficulties; this has been already achieved for the Poisson solver in Chapter 3.

A 3D Immersed Interface Vortex Particle-Mesh method

5.1 Introduction

In the previous chapters, we detailed every technical aspect required for the development of the 3D Immersed Interface Vortex Particle-Mesh method. This method is built on several recent works that paved the way in 2D [76, 81], and it includes many of the tools presented here: the efficient Poisson solver from Chapter 3, the recycling solver from Chapter 2 and the 2D Immersed Interface Vortex Particle-Mesh (IIVPM) method presented in Chapter 4.

In this chapter, we first present the extension of the IIVPM method from 2D to 3D, including the specific aspect of the treatment of the divergence of the vorticity field, see §5.2. Second, in §5.3, we validate the IIVPM3D method, and especially the evaluation of wall quantities and analyse their behaviors on the benchmark of an impulsively started flow past a sphere at moderate Reynolds number ($Re = 300$). Then, we assess the method using a longer time simulation on the same benchmark and we compare our results and their computational cost with those of the iterative penalization method, as presented in Chapter 2. In §5.5, we present the impulsively started flow past an ellipsoid with an angle of attack. Finally, a conclusion of our approach is presented in §5.6, together with future perspectives.

5.2 Methodology

In this section, we shortly describe the extension to 3D of the IIVPM method. The method is mostly identical to what was proposed in 2D, except for the Poisson solver and the reprojection operation (which was not needed in 2D). Throughout this chapter, we consider the computational domain, Ω , enclosing an immersed body, Ω_b and its surface, $\partial\Omega_b$. In this chapter, we will consider a fixed, non-rotating and simply connected immersed body.

5.2.1 Poisson solver

As proposed in Chapter 3 and by Poncet [109], we use the Helmholtz decomposition of the velocity to compute it from Ψ and Φ :

$$\mathbf{u} = (\mathbf{U}_\infty + \nabla \wedge \Psi) + \nabla \Phi, \quad (5.1)$$

where $\mathbf{U}_\infty + \nabla \wedge \Psi \triangleq \mathbf{u}_b$ represents the bulk contribution, which is solved first, using a bulk Poisson solver. The body contribution is then obtained from the potential, Φ , with a no-through flow boundary condition,

$$\nabla^2 \Phi = 0 \quad \text{such that} \quad \frac{\partial \Phi}{\partial n} = -\mathbf{u}_b \cdot \mathbf{n} \quad \text{on } \partial\Omega_b. \quad (5.2)$$

Implementation

From an implementation point of view, we sequentially solve the following equations,

$$\nabla^2 \Psi = -\omega \quad (5.3a)$$

$$[\nabla^2 + J^\pm] \Phi = 0 \quad (5.3b)$$

where Eq. (5.3b) is solved using the Sherman-Morrison-Woodbury (SMW) approach [45, 104, 105] and the Lattice Green's Function (LGF) convolution kernel [80], as proposed in Chapter 3,

$$\phi = \left[\nabla_h^{-2} - \nabla_h^{-2} E_a C [I_i + E_i^T \nabla_h^{-2} E_a C]^{-1} E_i^T \nabla_h^{-2} \right] f, \quad (5.4)$$

where E_a is the restriction operator to the biased points, E_i is the restriction operator to the interpolation points and C is the Immersed Interface jumps computation. Let us note that, in the present context, the source term is null ($f = 0$) and that the IIM application is an affine linear application, due to the non-zero boundary condition,

$$[E_a C E_i^T] x = [E_a \tilde{C} E_i^T x + E_a \tilde{C}_{bc}] . \quad (5.5)$$

The potential function, Φ , is then computed as

$$[I_i + E_i^T \nabla_h^{-2} E_a \tilde{C}] x = -E_i^T \nabla_h^{-2} E_a \tilde{C}_{bc} \quad (5.6a)$$

$$\Phi = -[\nabla_h^{-2} E_a \tilde{C} x + E_a \tilde{C}_{bc}] \quad (5.6b)$$

The Poisson solver for Eq. (5.3a) is an inflow/outflow solver [16], in order to handle the vorticity leaving the domain. Such a solver can be used in two outflow configurations: the “no-lift” and the “no-drag” configurations. The no-drag condition, typical of lifting bodies, allows normal vorticity at the outflow ($\partial_n \omega_n = 0$ and $\omega_t = 0$). The no-lift condition, typical for dragging bodies, imposes a zero normal vorticity at the outflow ($\omega_n = 0$ and $\partial_n \omega_t = 0$).

When considering the Poisson solver in Eqs. (5.6a) and (5.6b), the source term is compact and located in the vicinity of the body surface, and therefore a fully unbounded Poisson solver is used. The system in Eq. (5.6a) is solved using a recycling GMRes (see Fischer [115] and Chapter 2) and a subdomain in order to reduce the computational cost associated to the iterations. Finally, Eq. (5.6b) is solved in the whole computational domain, using an unbounded Poisson solver.

Let us note that the mismatch between the boundary conditions of the different Poisson solvers is the major disadvantage of the subdomain approach. However, it amounts to a very small error, as it is limited to the inflow and outflow regions, usually located far from the body, and it will therefore have a very limited impact on the solution. The computational cost associated with this Poisson step is

$$4 \mathcal{O}(n_{xyz} \log [n_{xyz}]) + N_{iter} \mathcal{O}(n_{xyz}^{sub} \log [n_{xyz}^{sub}]) \quad (5.7)$$

where N_{iter} is the number of iteration required to solve Eq. (5.6a), n_{xyz} and n_{xyz}^{sub} is the number of unknowns associated to the full domain and the subdomain, respectively.

5.2.2 Wall vorticity

As in 2D, the no-through flow condition will induce a spurious slip velocity, which will in turn generate a vortex sheet at the wall. The wall vorticity is then computed as the sum of the normal vorticity and the planar one, respectively given by

$$\omega_{wall}^\perp = 2\mathbf{W} \cdot \mathbf{n} \quad (5.8)$$

$$\omega_{wall}^\pi = \mathbf{n} \wedge \left(\frac{\partial \mathbf{u}}{\partial n} \right) \quad (5.9)$$

where $\mathbf{W}$ is the angular velocity of the body. In this work, we consider non-moving and non-rotating bodies ($\mathbf{W} = 0$). The wall vorticity is then obtained as

$$\omega_{wall} = \omega_{wall}^\pi = \mathbf{n} \wedge \left(\frac{\partial \mathbf{u}}{\partial n} \right) \quad (5.10)$$

From an implementation point of view, by using three bulk values of the velocity and the boundary condition, we are able to recover the wall vorticity at $\mathcal{O}(h^3)$. However, in practice, when the wall is poorly resolved, an evaluation of the wall vorticity at $\mathcal{O}(h^2)$, hence using two bulk values and the boundary condition, is more robust.

5.2.3 Diffusion and stretching

The wall vorticity is used to compute the diffusion term, $\nabla^2 \boldsymbol{\omega}$, and the stretching term, $(\boldsymbol{\omega} \cdot \nabla) \mathbf{u}$. The diffusion term is evaluated using standard second order finite differences and the Immersed Interface jumps based on the boundary condition, $\boldsymbol{\omega}_{wall}$, compatible with the second order accuracy of the finite difference stencil. As already mentioned, a second order evaluation of the wall vorticity is more robust than a third order evaluation, and it has been observed to be sufficient to ensure a second order convergence in the bulk, see Chapter 4.

The stretching term is here considered in its conservative form,

$$\nabla \cdot (\boldsymbol{\omega} \mathbf{u}) = \frac{\partial}{\partial x_j} (u_i \omega_j) \quad , \quad (5.11)$$

which is evaluated using a standard second order finite differences and the Immersed Interface jumps, with a zero Dirichlet boundary condition on $(\boldsymbol{\omega} \mathbf{u})$, compatible with the global accuracy of the scheme.

5.2.4 Particle-Mesh Interpolation

As in Chapter 4, the field is extended using a second order Hamilton-Jacobi extension in the body. Following §4.2.5, the extension of a field f is computed as the stationary solution, in pseudo-time τ , of the following system,

$$\frac{\partial}{\partial \tau} \begin{bmatrix} f \\ f' \end{bmatrix} + C \begin{bmatrix} \mathbf{n}_{in} \cdot \nabla & -1 \\ 0 & \mathbf{n}_{in} \cdot \nabla \end{bmatrix} \begin{bmatrix} f \\ f' \end{bmatrix} = 0 \quad , \quad (5.12)$$

where C is the smoothing function (see Chapter 4, in order to take into account only a thin layer around the interface, $\mathbf{n}_{in}$ the ingoing normal to the body defined as

$$\mathbf{n}_{in} = -\frac{\nabla \phi}{|\nabla \phi|} \quad , \quad (5.13)$$

with ϕ the levelset representation of the body and f' the normal derivative of the field,

$$f' = \mathbf{n}_{in} \cdot \nabla f \quad . \quad (5.14)$$

For the same reason as in 2D, we first compute the field f and f' in the bulk, using the IIM with an extrapolated boundary condition, and then extend them inside the body.

Ghost particles are computed at the beginning of each (sub) time step, using the extended field computed by solving Eq. (5.12). Those particles will have the same evolution as the bulk particles, i.e. they evolve according to the velocity, to the diffusion and to the stretching term.

5.2.5 Reprojection

Due to the advection and Lagrangian deformation, particles will eventually generate a non divergence-free field, $\tilde{\omega}$. To counteract this phenomenon, we use a Helmholtz-Dodge decomposition that decomposes the vorticity on the mesh into a divergence-free and a curl-free component. This approach is usually also referred to as the solenoidal reprojection method. The divergence-free vorticity field, ω , is then computed as

$$\omega = \tilde{\omega} + \nabla A \quad , \quad (5.15)$$

where A is a scalar field that satisfies the Poisson equation,

$$\nabla^2 A = -\chi \nabla \cdot \tilde{\omega} \quad \text{such that} \quad \left. \frac{\partial A}{\partial n} \right|_{\partial\Omega_b} = 0 \quad , \quad (5.16)$$

with χ , a “relaxation mask”, defined as

$$\chi = \begin{cases} 0 & \text{if } \phi(\mathbf{x}) < h_c \\ \sin\left(\frac{\pi}{2} \frac{(\phi(\mathbf{x}) - h_c)}{h_c}\right) & \text{if } h_c \leq \phi(\mathbf{x}) \leq 2h_c \\ 1 & \text{if } 2h_c < \phi(\mathbf{x}) \end{cases} \quad . \quad (5.17)$$

This masking function is typically added to neglect the divergence source term in the regions close to where the vorticity is generated.

Implementation

From an implementation point of view, we use finite differences for the computation of the divergence in Eq. (5.16) and the gradient in Eq. (5.15). They are then corrected by Immersed Interface jumps if no masking function is used, which costs an extra Poisson solver to compute an up-to-date wall vorticity. The Poisson problem, Eq. (5.16), is solved using a FFT-based solver, similar to the one used to compute the velocities, see §5.2.1. The FFT type is chosen to impose an anti-symmetry condition at the outflow for

a no-lift boundary condition ($\partial_n A = 0$) and a symmetry condition in the case of a no-drag outflow condition ($A = 0$). In the other directions, we impose an anti-symmetry boundary condition. In order to ensure a perfect compatibility with the Immersed Interface jumps used to take the inner boundary into account, the convolution kernel used is, expressed in the Fourier space,

$$\hat{G}_h = \frac{1}{4 \left[\sin^2 \left(\frac{k_x h}{2} \right) + \sin^2 \left(\frac{k_y h}{2} \right) + \sin^2 \left(\frac{k_z h}{2} \right) \right]} . \quad (5.18)$$

Since the problem is no longer unbounded, no subdomain can be used. Finally, let us note that, due to the non-staggered grid employed here, the divergence will decrease, yet it will not be canceled exactly. Using this approach, we then impose an “almost divergence-free” vorticity every $N_{solenoidal} \simeq 20$ time steps.

For the IIVPM application, based on our experience, the generation of the divergence is very strong at the wall and especially when considering a body surface poorly resolved. Moreover, the jumps used in the divergence evaluation will also increase the mismatch due to the non-staggered approach. Therefore, when the wall is poorly resolved, we recommend to discard the divergence and the vorticity correction in the boundary layer in order to not corrupt the near-wall vorticity (using typically $h_c \simeq (4 - 8) h$).

5.2.6 Force computation

First, let us recall some basics from continuum mechanics (we here use the notation and definitions from Wu et al. [116]). The strain rate tensor, the rotation rate tensor (also called the vorticity tensor) and the surface deformation rate tensor are defined respectively as

$$\mathbf{D} = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T) \quad (5.19a)$$

$$\mathbf{\Omega} = \frac{1}{2} (\nabla \mathbf{u} - \nabla \mathbf{u}^T) \quad (5.19b)$$

$$\mathbf{B} = \theta \mathbf{I} - \nabla \mathbf{u}^T . \quad (5.19c)$$

where $\theta = \text{tr}(\mathbf{D}) = \nabla \cdot \mathbf{u}$ denotes the dilatation. They are linked by the following relations,

$$\mathbf{D} = \theta \mathbf{I} + \mathbf{\Omega} - \mathbf{B} \quad (5.20a)$$

$$\nabla \mathbf{u} = \theta \mathbf{I} + 2\mathbf{\Omega} - \mathbf{B} , \quad (5.20b)$$

which come from the triple decomposition of $\mathbf{D}$ and $\nabla \mathbf{u}$ [116, p.50]. The stress tensor, $\mathbf{T}$, is then given as

$$\mathbf{T} = (-p + \lambda\theta)\mathbf{I} + 2\mu\mathbf{D} , \quad (5.21)$$

with p the pressure. A first decomposition is obtained using the deviator of the strain rate tensor, $\tilde{\mathbf{D}} = \mathbf{D} - \frac{1}{3} \text{tr}(\mathbf{D})\mathbf{I} = \mathbf{D} - \frac{1}{3}\theta\mathbf{I}$,

$$\mathbf{T} = \left[-p + \left(\lambda + \frac{2\mu}{3} \right) \theta \right] \mathbf{I} + 2\mu\tilde{\mathbf{D}} = [-p + \zeta\theta]\mathbf{I} + 2\mu\tilde{\mathbf{D}} , \quad (5.22)$$

where $\zeta = \left(\lambda + \frac{2\mu}{3} \right)$ denotes the bulk viscosity. Another decomposition, proposed by Wu et al. [116, p.52], uses Eq. (5.20a), to obtain a triple decomposition of the stress tensor,

$$\mathbf{T} = -\Pi\mathbf{I} + 2\mu\mathbf{\Omega} - 2\mu\mathbf{B} , \quad (5.23)$$

where $\Pi = p - (\lambda + 2\mu)\theta$ consist of the pressure and a viscous contribution of dilatation. Let us note that since $\mathbf{B}$ is trace-free, the surface deformation process does not affect the surface force integrated over a closed boundary [116, p.52]. Therefore, we introduce the reduced stress tensor,

$$\hat{\mathbf{T}} = -\Pi\mathbf{I} + 2\mu\mathbf{\Omega} , \quad (5.24)$$

which can be used in the force integrations over a closed surface.

Still following Wu et al. [116] notation, we consider a surface and its outgoing normal, $\mathbf{n}$, and we obtain that

$$\mathbf{n} \cdot \mathbf{B} = r_s \mathbf{n} + \mathbf{W} \wedge \mathbf{n} , \quad (5.25)$$

where $r_s = \nabla_\pi \cdot \mathbf{u} = \theta - \frac{\partial u_n}{\partial n}$ is the rate of change of this surface area and with $\mathbf{W}$ the angular velocity. The surface stress is defined as $\mathbf{t} = \mathbf{n} \cdot \mathbf{T}$. Similarly to the stress tensor, since $\mathbf{B}$ is trace-free, we introduce the reduced surface stress,

$$\hat{\mathbf{t}} = \mathbf{n} \cdot \hat{\mathbf{T}} = -\Pi\mathbf{n} + \mu\boldsymbol{\omega} \wedge \mathbf{n} . \quad (5.26)$$

From a more general point of view, the surface stress is also given by $\mathbf{t} = \hat{\mathbf{t}} + \mathbf{t}_s$ where $\mathbf{t}_s = -2\mu\mathbf{n} \cdot \mathbf{B} = -2\mu(r_s \mathbf{n} + \mathbf{W} \wedge \mathbf{n})$. Combining those equations, we finally obtain that the expression for the surface stress,

$$\mathbf{t} = \mathbf{n} \cdot \mathbf{T} = -\tilde{\Pi}\mathbf{I} + \boldsymbol{\tau} , \quad (5.27)$$

with $\tilde{\Pi} = \left(p - \lambda\theta - 2\mu \frac{\partial u_n}{\partial n} \right)$ and $\boldsymbol{\tau} = \mu\boldsymbol{\omega}_r \wedge \mathbf{n} = \mu(\boldsymbol{\omega} - 2\mathbf{W}) \wedge \mathbf{n}$, the skin friction stress.

In this work, we use two ways to compute the force acting on the immersed body. First, we apply the control volume approach proposed by Noca [113, Eq. (81)] which relies exclusively on bulk quantities, here in 3D,

$$\mathbf{F} = -\frac{1}{2} \frac{d}{dt} \int_{\Omega+\Omega_b} \mathbf{x} \wedge \boldsymbol{\omega} \, dV + \int_{\Omega+\Omega_b} \mathbf{u} \wedge \boldsymbol{\omega} \, dV \quad (5.28a)$$

$$+ \frac{d}{dt} \int_{\Omega_b} \mathbf{u} \, dV + \frac{1}{2} \oint_{S^*} \mathbf{n}_s \cdot \boldsymbol{\omega} (\mathbf{x} \wedge \mathbf{u}) \, dS \quad (5.28b)$$

$$- \frac{1}{2} \oint_{S^*} \mathbf{n}_s \cdot (\mathbf{u} - \mathbf{u}_s) (\mathbf{x} \wedge \boldsymbol{\omega}) \, dS \quad (5.28c)$$

$$+ \frac{1}{2} \oint_{S^*} \mathbf{x} \wedge [\mathbf{n}_s \wedge (\nabla \cdot \mathbf{T})] \, dS + \oint_{S^*} \mathbf{n}_s \cdot \mathbf{T} \, dS \quad (5.28d)$$

Second, we can also recover the force using wall quantities, where it is decomposed into the pressure and the viscous contribution [116, Eq. (2.71)],

$$\mathbf{F} = \oint_{\partial\Omega_b} \mathbf{n} \cdot \mathbf{T} \, dS = \oint_{\partial\Omega_b} \mathbf{n} \cdot \hat{\mathbf{T}} \, dS = \mathbf{F}_p + \mathbf{F}_v \quad (5.29)$$

with

$$\mathbf{F}_p = \oint_{\partial\Omega_b} -p \mathbf{n} \, dS \quad (5.30a)$$

$$\mathbf{F}_v = \oint_{\partial\Omega_b} \boldsymbol{\tau} \, dS = \mu \oint_{\partial\Omega_b} \boldsymbol{\omega} \wedge \mathbf{n} \, dS \quad (5.30b)$$

Pressure Force

The pressure integral is computed using the Burgatti formula [113, p8],

$$\oint_{\partial\Omega_b} \mathbf{x} \wedge (\mathbf{n} \wedge \nabla p) \, dS = -(N-1) \int_{\Omega} \nabla p \, dV + \int_{\Omega} \mathbf{x} \wedge (\nabla \wedge \nabla p) \, dV \quad (5.31)$$

The last term vanishes since $\nabla \wedge \nabla (\cdot) = 0$ and, using the Green's theorem, this equality becomes

$$\oint_{\partial\Omega_b} \mathbf{x} \wedge (\mathbf{n} \wedge \nabla p) \, dS = -(N-1) \int_{\Omega_b} \nabla p \, dV = -(N-1) \oint_{\partial\Omega_b} p \mathbf{n} \, dS \quad (5.32)$$

The pressure term of the force can then be computed as

$$\mathbf{F}_p = \oint_{\partial\Omega_b} -p \mathbf{n} \, dS = \frac{1}{(N-1)} \oint_{\partial\Omega_b} \mathbf{x} \wedge (\mathbf{n} \wedge \nabla p) \, dS \quad (5.33)$$

To evaluate the term $\mathbf{n} \wedge \nabla p$, we use the momentum equation

$$\rho \mathbf{n} \wedge \left[\frac{1}{\rho} \nabla p = -\frac{D\mathbf{u}}{Dt} - \nu \nabla \wedge \boldsymbol{\omega} \right] \quad (5.34)$$

$$(\mathbf{n} \wedge \nabla p) = -\rho \mathbf{n} \wedge \frac{D\mathbf{u}}{Dt} - \mu \mathbf{n} \wedge (\nabla \wedge \boldsymbol{\omega}) \quad (5.35)$$

Here, we assume that the velocity at the body surface is known, $\mathbf{u} = \mathbf{u}_b$. We finally obtain the following relation for the pressure gradient, at the wall,

$$(\mathbf{n} \wedge \nabla p) = -\rho \mathbf{n} \wedge \frac{D\mathbf{u}_b}{Dt} - \mu \mathbf{n} \wedge (\nabla \wedge \boldsymbol{\omega})|_{\partial\Omega_b}. \quad (5.36)$$

Several techniques are available to evaluate the term $\mathbf{n} \wedge (\nabla \wedge \boldsymbol{\omega})|_{\partial\Omega_b}$. Hereunder, we present two different approaches.

The standard decomposition

We use the following property

$$\nabla (\mathbf{n} \cdot \boldsymbol{\omega}) = (\mathbf{n} \cdot \nabla) \boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla) \mathbf{n} + \mathbf{n} \wedge (\nabla \wedge \boldsymbol{\omega}) + \boldsymbol{\omega} \wedge (\nabla \wedge \mathbf{n}), \quad (5.37)$$

to obtain

$$\mathbf{n} \wedge (\nabla \wedge \boldsymbol{\omega}) = [\nabla (\mathbf{n} \cdot \boldsymbol{\omega}) - (\mathbf{n} \cdot \nabla) \boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \nabla) \mathbf{n} - \boldsymbol{\omega} \wedge (\nabla \wedge \mathbf{n})]. \quad (5.38)$$

Since $\mathbf{n} = \nabla \phi$, where ϕ is the body levelset function, we have that $(\nabla \wedge \mathbf{n}) = 0$ and $\mathbf{n} \cdot \boldsymbol{\omega} = 2 \mathbf{W} \cdot \mathbf{n}$ is known at the boundary. Finally, we obtain the following relation, linking the pressure and the vorticity field, at the wall:

$$(\mathbf{n} \wedge \nabla p) = -\rho \mathbf{n} \wedge \frac{D\mathbf{u}_b}{Dt} + \mu [(\mathbf{n} \cdot \nabla) \boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla) \mathbf{n} - \nabla (\mathbf{n} \cdot \boldsymbol{\omega})]|_{\partial\Omega_b}. \quad (5.39)$$

For a fixed and non-rotating 3D body ($\frac{D\mathbf{u}_b}{Dt} = 0$ and $\mathbf{W} = 0$), the pressure component of the force is then given by

$$\mathbf{F}_p = \frac{\mu}{2} \oint_{\partial\Omega_b} \mathbf{x} \wedge [(\mathbf{n} \cdot \nabla) \boldsymbol{\omega} + (\boldsymbol{\omega} \cdot \nabla) \mathbf{n}] dS. \quad (5.40)$$

The divergence decomposition

Another approach, proposed by Wu et al. [116, p140], highlights the contribution of the divergence of the vorticity. Similarly to the first approach, one obtains that

$$-\mathbf{n} \wedge (\nabla \wedge \boldsymbol{\omega}) = (\mathbf{n} \cdot \nabla) \boldsymbol{\omega} - (\mathbf{n} \wedge \nabla) \wedge \boldsymbol{\omega} \quad (5.41)$$

$$= (\mathbf{n} \cdot \nabla) \boldsymbol{\omega} - [(\nabla \boldsymbol{\omega}) \cdot \mathbf{n} - \mathbf{n} (\nabla \cdot \boldsymbol{\omega})] \quad (5.42)$$

$$= (\mathbf{n} \cdot \nabla) \boldsymbol{\omega} - (\nabla \boldsymbol{\omega}) \cdot \mathbf{n} + \mathbf{n} (\nabla \cdot \boldsymbol{\omega}) \quad (5.43)$$

So, the force component due to the pressure is also given by

$$\mathbf{F}_p = \frac{\mu}{2} \oint_{\partial\Omega_b} \mathbf{x} \wedge [(\mathbf{n} \cdot \nabla) \boldsymbol{\omega} - (\nabla \boldsymbol{\omega}) \cdot \mathbf{n} + \mathbf{n} (\nabla \cdot \boldsymbol{\omega})] dS . \quad (5.44)$$

Let us note that in our case, the divergence of the vorticity should be zero, while in practice, this is not exactly the case, see §5.2.5.

These two formulations are very close one to the other; yet they give different results in practice, especially for a poorly resolved body interface.

5.2.7 Numerical evaluation of surface integrals

In this section, we present a technique to evaluate surface integrals on the immersed surface. This approach, which uses only the level set function, is based on the work proposed by Tornberg et al. [117] and generalized to the Immersed Interface framework by Smereka [118]. It relies on the computation of an equivalent surface, ΔS_α , for each intersection point, such that

$$\oint_{\partial\Omega_b} f(\mathbf{x}) dS \simeq \sum_k \Delta S_{\alpha_k} f(\mathbf{x}_{\alpha_k}) , \quad (5.45)$$

hence the body surface is obtained as

$$S_{\partial\Omega} \simeq \sum_k \Delta S_{\alpha_k} . \quad (5.46)$$

As proposed by Smereka [118], this equivalent surface is defined as

$$\Delta S_{\alpha_k} \triangleq \left[\tilde{\delta}_{\alpha_k}^- + \tilde{\delta}_{\alpha_k}^+ \right] h^3 , \quad (5.47)$$

where $\tilde{\delta}_{\alpha_k}^{-/+}$ represents the density contribution of the left and right biased point to the Dirac function associated with the intersection point,

$$\frac{\left[\tilde{\delta}_{\alpha_k}^- + \tilde{\delta}_{\alpha_k}^+ \right]}{h^2} \triangleq \delta_{\alpha_k} . \quad (5.48)$$

In other words, $\tilde{\delta}_{\alpha_k}^{-/+}$ is the representation on the grid of δ_{α_k} . A given biased point will have a contribution in each intersection direction, $\tilde{\delta}_{ijk} \triangleq \tilde{\delta}_\alpha^x + \tilde{\delta}_\alpha^y + \tilde{\delta}_\alpha^z$. This Dirac function satisfies the discretized equation of the associated Green's function, G_α ,

$$\nabla_h^2 G_\alpha = \delta_\alpha , \quad (5.49)$$

where ∇_h^2 stands for the standard second order finite difference Laplacian. It is well known that this problem is equivalent to a Laplace equation, in the bulk domain, i.e. for $x \notin \Omega_b$,

$$\nabla_h^2 G_\alpha = 0 \quad , \quad (5.50)$$

such that $[G_\alpha] = 0$ and $\left[\frac{\partial G_\alpha}{\partial n}\right] = 1$ on the interface, $\partial\Omega_b$. Using the Immersed Interface, this Laplace equation then becomes

$$\nabla_h^2 G_\alpha - \frac{1}{h^2} \left[J_x^\pm(G_\alpha) + J_y^\pm(G_\alpha) + J_z^\pm(G_\alpha) \right] = 0 \quad , \quad (5.51)$$

with

$$J_x^\pm = \tilde{\delta}_\alpha^x = h^\pm \left[\frac{\partial G_\alpha}{\partial x} \right]_\alpha + \frac{(h^\pm)^2}{2} \left[\frac{\partial^2 G_\alpha}{\partial x^2} \right]_\alpha + \mathcal{O}(h^3) \quad (5.52a)$$

$$J_y^\pm = \tilde{\delta}_\alpha^y = h^\pm \left[\frac{\partial G_\alpha}{\partial y} \right]_\alpha + \frac{(h^\pm)^2}{2} \left[\frac{\partial^2 G_\alpha}{\partial y^2} \right]_\alpha + \mathcal{O}(h^3) \quad (5.52b)$$

$$J_z^\pm = \tilde{\delta}_\alpha^z = h^\pm \left[\frac{\partial G_\alpha}{\partial z} \right]_\alpha + \frac{(h^\pm)^2}{2} \left[\frac{\partial^2 G_\alpha}{\partial z^2} \right]_\alpha + \mathcal{O}(h^3) \quad . \quad (5.52c)$$

where we have used the boundary condition $G_{h,\alpha} = 0$ and $[\cdot]_\alpha$ is the jump across the interface $\mathbf{x}_\alpha$. In the Immersed Interface framework, the contribution of a biased point to the body surface is then given by

$$\tilde{\delta}_{ijk} = \left[J_x^\pm(G_\alpha) + J_y^\pm(G_\alpha) + J_z^\pm(G_\alpha) \right] \quad . \quad (5.53)$$

Without loss of generality, we now consider the x direction only, since the extension to y and z directions is straightforward. Using the boundary condition $\frac{\partial G_\alpha}{\partial n} = 1$, we obtain that

$$\left[\frac{\partial G_\alpha}{\partial x} \right]_\alpha = n_x = \left[\frac{\partial \phi}{\partial x} \right]_\alpha \quad , \quad (5.54)$$

where ϕ is the levelset function (such that $|\nabla \phi| = 1$). Therefore, the jump expression becomes

$$\tilde{\delta}_\alpha^x = h^\pm \left[\frac{\partial \phi}{\partial x} \right]_\alpha + \frac{(h^\pm)^2}{2} \left[\frac{\partial^2 G_\alpha}{\partial x^2} \right]_\alpha + \mathcal{O}(h^3) \quad . \quad (5.55)$$

Second derivative in two dimensions

In the following, we use the simplified notation: $\partial_x(\cdot) = \frac{\partial}{\partial x}(\cdot)$ and $\nabla_t(\cdot) = (\mathbf{t} \cdot \nabla)(\cdot)$, where $\mathbf{t}$ is a tangent vector. For the evaluation of the second derivative, we use the

Laplace equation ($\nabla_h^2 G_\alpha = 0$), together with the two other relations ($\partial_{tn}^2 G_\alpha = \partial_{tt}^2 G_\alpha = 0$), derived from the boundary conditions. In two dimensions, the system then becomes

$$\begin{bmatrix} t_x n_x & t_x n_y + t_y n_x & t_y n_y \\ t_x^2 & 2t_y t_x & t_y^2 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \partial_{xx}^2 G_\alpha \\ \partial_{xy}^2 G_\alpha \\ \partial_{yy}^2 G_\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ -\mathbf{n} \cdot (\nabla_t \mathbf{t}) \\ 0 \end{bmatrix}, \quad (5.56)$$

and its solution is given by,

$$[\partial_{xx}^2 G_\alpha]_\alpha = \frac{\text{sign}(n_x)}{D} [\mathbf{n} \cdot (\nabla_t \mathbf{t})] \quad (5.57a)$$

$$[\partial_{yy}^2 G_\alpha]_\alpha = -\frac{\text{sign}(n_y)}{D} [\mathbf{n} \cdot (\nabla_t \mathbf{t})] \quad (5.57b)$$

with

$$D = (t_x^2 + t_y^2) \frac{(n_x t_y - n_y t_x)}{(n_x t_y + n_y t_x)}. \quad (5.58)$$

Using the fact that $|\mathbf{t}|^2 = (t_x^2 + t_y^2) = 1$ and $\mathbf{n} \wedge \mathbf{t} = -\mathbf{e}_z$, we finally obtain

$$[\partial_{xx}^2 G_\alpha]_\alpha = -\text{sign}(n_x) (n_x t_y + n_y t_x) [\mathbf{n} \cdot (\nabla_t \mathbf{t})] \quad (5.59a)$$

$$[\partial_{yy}^2 G_\alpha]_\alpha = \text{sign}(n_y) (n_x t_y + n_y t_x) [\mathbf{n} \cdot (\nabla_t \mathbf{t})] \quad (5.59b)$$

Finally, we can link the development done by Marichal [14] and the method proposed by Smereka [118]. Indeed, using $\mathbf{t} = \mathbf{e}_z \wedge \mathbf{n}$, we obtain that

$$\begin{aligned} \mathbf{n} \cdot (\partial_t \mathbf{t}) &= \mathbf{n} \cdot [\partial_t (\mathbf{e}_z \wedge \mathbf{n})] = \mathbf{n} \cdot [\mathbf{e}_z \wedge (\partial_t \mathbf{n})] \\ &= (\partial_t \mathbf{n}) \cdot (\mathbf{n} \wedge \mathbf{e}_z) = -\mathbf{t} \cdot (\partial_t \mathbf{n}) \end{aligned} \quad (5.60)$$

While we also know that, since $|\mathbf{t}| = 1$,

$$\nabla^2 \phi = \nabla \cdot \mathbf{n} = \mathbf{e}_n (\partial_n \mathbf{n}) + \mathbf{e}_t (\partial_t \mathbf{n}) = -[\mathbf{n} \cdot (\partial_t \mathbf{t})]. \quad (5.61)$$

In two dimensions, the approach of Marichal [14] and Smereka [118] are then equivalent.

Second derivative in three dimensions

In three dimensions, we use the Laplacian ($\nabla_h^2 G_\alpha = 0$) and the derivatives ($\partial_{sn}^2 G_\alpha = \partial_{tn}^2 G_\alpha = \partial_{st}^2 G_\alpha = \partial_{ss}^2 G_\alpha = \partial_{tt}^2 G_\alpha = 0$), where $\mathbf{s} = (-n_y; n_x + n_z; -n_y)^T$ and $\mathbf{t}$ are the two tangent vectors, such that $\mathbf{n} \wedge \mathbf{s} = \mathbf{t}$. The system is a straightforward extension of

the two case, so is the solution. In Smereka [118], they propose to solve the system using a gaussian elimination but a symbolic expression could be faster to evaluate and easily generated. The system to solve is given by

$$A \begin{bmatrix} \partial_{xx}^2 G_\alpha \\ \partial_{xy}^2 G_\alpha \\ \partial_{xz}^2 G_\alpha \\ \partial_{yy}^2 G_\alpha \\ \partial_{yz}^2 G_\alpha \\ \partial_{zz}^2 G_\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\mathbf{n} \cdot (\nabla_s \mathbf{t}) \\ -\mathbf{n} \cdot (\nabla_s \mathbf{s}) \\ -\mathbf{n} \cdot (\nabla_t \mathbf{t}) \\ 0 \end{bmatrix}, \quad (5.62)$$

with

$$A = \begin{bmatrix} s_x n_x & s_x n_y + s_y n_x & s_x n_z + s_z n_x & s_y n_y & s_y n_z + s_z n_y & s_z n_z \\ t_x n_x & t_x n_y + t_y n_x & t_x n_z + t_z n_x & t_y n_y & t_y n_z + t_z n_y & t_z n_z \\ s_x t_x & s_x t_y + s_y t_x & s_x t_z + s_z t_x & s_y t_y & s_y t_z + s_z t_y & s_z t_z \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \quad (5.63)$$

Similarly to the two dimensional approach, the solution is a linear combination of three terms ($\mathbf{n} \cdot (\partial_s \mathbf{t})$, $\mathbf{n} \cdot (\partial_s \mathbf{s})$ and $\mathbf{n} \cdot (\partial_t \mathbf{t})$), which can be rewritten, using $\mathbf{n} \wedge \mathbf{s} = \mathbf{t}$ and $\mathbf{n} \wedge \mathbf{t} = -\mathbf{s}$,

$$\mathbf{n} \cdot (\partial_t \mathbf{t}) = -\mathbf{t} \cdot (\partial_t \mathbf{n}) \quad (5.64a)$$

$$\mathbf{n} \cdot (\partial_s \mathbf{t}) = -\mathbf{t} \cdot (\partial_s \mathbf{n}) \quad (5.64b)$$

$$\mathbf{n} \cdot (\partial_s \mathbf{s}) = -\mathbf{s} \cdot (\partial_s \mathbf{n}) \quad (5.64c)$$

Considering a sphere body, let us note that $\mathbf{n} \cdot (\partial_s \mathbf{t}) = 0$. However, it is interesting to note that the Laplacian of the level set can be rewritten as

$$\nabla^2 \phi = \nabla \cdot \mathbf{n} = \mathbf{e}_n \partial_n \mathbf{n} + \mathbf{e}_t \partial_t \mathbf{n} + \mathbf{e}_s \partial_s \mathbf{n} = -[\mathbf{n} \cdot (\partial_t \mathbf{t}) + \mathbf{n} \cdot (\partial_s \mathbf{s})], \quad (5.65)$$

For bodies without warping ($\mathbf{n} \cdot (\partial_s \mathbf{t}) = 0$), a sphere or an ellipsoid for example, the Laplacian contains all the required information. Therefore, we could envision an approach similar to Marichal [14]. However, when the body is warped, the Laplacian of the levelset does not contain enough information and will then fail to retrieve the correct surface. In order to stay most general, we choose to apply the approach proposed by Smereka [118].

5.3 Validation of the wall quantities

In this section, we validate the approach proposed above. We consider a sphere of radius R ($D = 2R$), with a stream velocity $\mathbf{u}_\infty = (0 ; 0 ; U_\infty)$. The notation conven-

tion of the spherical coordinates used is detailed in Figure 5.1, where $\theta \in [-\pi ; \pi]$ and $\phi \in \left[-\frac{\pi}{2} ; \frac{\pi}{2}\right]$.

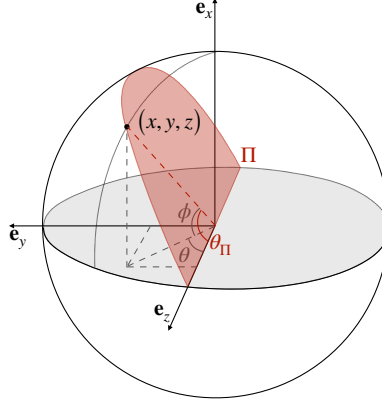


Figure 5.1: Spherical coordinates notation with $\mathbf{e}_z$ as stream direction.

5.3.1 Stokes flow past a sphere

First, we consider the Stokes flow past a sphere, which has an analytical solution for the velocity field, the vorticity field and the drag coefficient. The solution is axisymmetric, therefore, for any plane Π going through the center of the sphere, the solution, in the plane's coordinates (see Figure 5.1), is given as

$$u_r = U_\infty \left[1 - \frac{3}{2} \left(\frac{R}{r} \right) + \frac{1}{2} \left(\frac{R}{r} \right)^3 \right] \cos(\theta_\Pi) \quad (5.66a)$$

$$u_\theta = -U_\infty \left[1 - \frac{3}{4} \left(\frac{R}{r} \right) - \frac{1}{4} \left(\frac{R}{r} \right)^3 \right] \sin(\theta_\Pi) \quad (5.66b)$$

$$\omega = \frac{U_\infty}{R} \frac{3}{2} \left(\frac{R}{r} \right)^2 \sin(\theta_\Pi) \quad (5.66c)$$

$$\frac{\partial \omega}{\partial n} = \frac{\partial \omega}{\partial r} = -\frac{3 U_\infty}{R^2} \left(\frac{R}{r} \right)^3 \sin(\theta_\Pi) \quad (5.66d)$$

The drag coefficient can be decomposed into the pressure and the viscous contribution. They are related to the Reynolds number ($\text{Re}_D \ll 1$) as follow,

$$C_{D,p} = \frac{(2\pi\mu U_\infty R)}{\left(\frac{1}{2}\rho U_\infty \pi R^2\right)} = \frac{8}{\text{Re}_D} \quad (5.67a)$$

$$C_{D,v} = \frac{(4\pi\mu U_\infty R)}{\left(\frac{1}{2}\rho U_\infty \pi R^2\right)} = \frac{16}{\text{Re}_D} . \quad (5.67b)$$

Hereunder, we use this analytical solution with $\text{Re} = 10^{-3}$, in order to validate our computation of the wall quantities (wall vorticity, pressure component of the drag and viscous component of the drag).

Surface of the sphere

The surface of the sphere can be approximated using Eq. (5.46). The error of this approximation is defined as

$$\text{error}(S) \triangleq \frac{S - (4\pi R^2)}{(4\pi R^2)} . \quad (5.68)$$

It is seen in Figure 5.2 to reach a second order convergence. Let us note that this result, already obtained by Smereka [118] and in 2D by Marichal [14], is a key advantage of the IIVPM method, which proposes a way to compute an equivalent surface associated with each biased point, independently of the other intersections and regardless of any solution field.

Wall vorticity, wall vorticity flux and drag coefficient

The wall vorticity is computed using Eq. (5.10). We define the 2-norm and ∞ -norm of the error on the wall vorticity as

$$\text{error}_2(\omega_{wall}) \triangleq \frac{1}{U_\infty} \sqrt{\sum (\omega_{wall} - \omega_{wall}^{exact})^2 h^2} \quad (5.69a)$$

$$\text{error}_\infty(\omega_{wall}) \triangleq \frac{\max(|\omega_{wall} - \omega_{wall}^{exact}|)}{U_\infty/R} . \quad (5.69b)$$

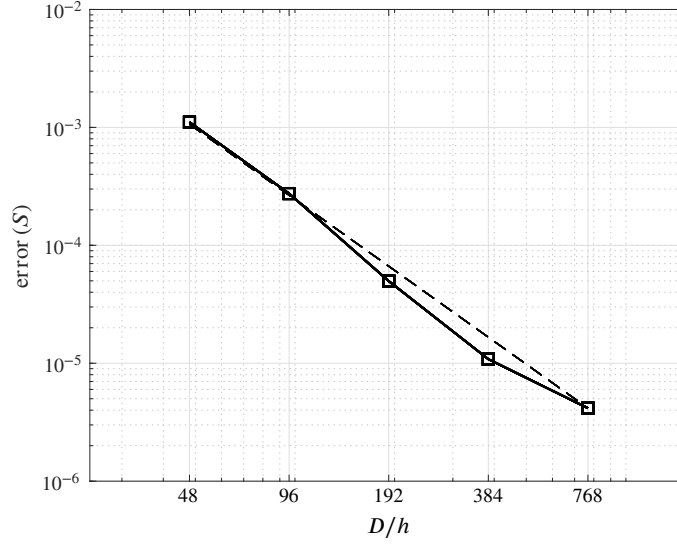


Figure 5.2: **Surface convergence** - Error on the surface evaluation using Eq. (5.46) ($\square$); $\mathcal{O}(h^2)$ convergence (- -).

The vorticity fluxes, $\frac{\partial \omega}{\partial n} = \partial_n \omega$, are also computed using the Immersed Interface tool; their error norms are defined as

$$\text{error}_2(\partial_n \omega_{wall}) \triangleq \frac{1}{U_\infty/R} \sqrt{\sum (\partial_n \omega_{wall} - \partial_n \omega_{wall}^{exact})^2 h^2} \quad (5.70a)$$

$$\text{error}_\infty(\partial_n \omega_{wall}) \triangleq \frac{\max(|\partial_n \omega_{wall} - \partial_n \omega_{wall}^{exact}|)}{U_\infty/R^2} . \quad (5.70b)$$

In Figure 5.3, we show the convergence of the wall vorticity and the normal wall derivative using these definitions. We note that, unlike the evaluation of the diffusion in the bulk or the stretching term, the order of convergence of the wall vorticity flux, $\partial_n \omega$, is heavily impacted by the order of accuracy of the wall vorticity evaluation. The wall vorticity needs to be evaluated at third order to obtain a second order convergence in the wall vorticity flux. As highlighted later, this observation is valid when the vorticity field is smooth enough on the grid (i.e. is well resolved) and thus the third order does not exhibit a very stiff behavior.

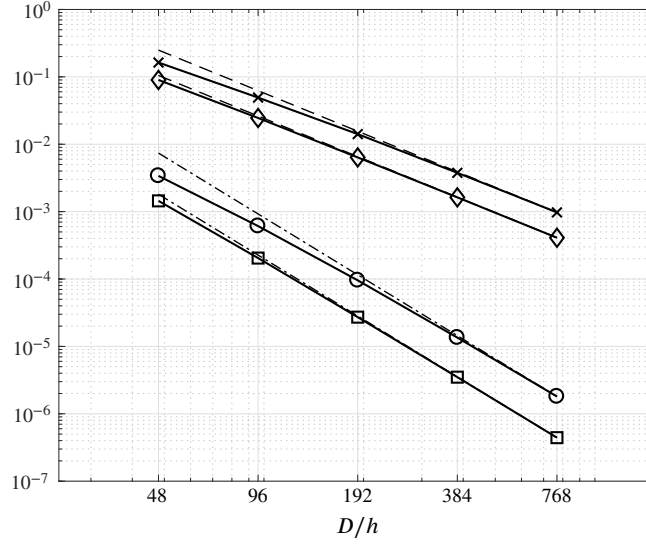


Figure 5.3: **Wall vorticity convergence** - error (ω_{wall}): 2-norm ($\blacksquare$) and ∞ -norm ($\bigcirc$); error ($\frac{\partial \omega}{\partial n}$): 2-norm ($\blacklozenge$) and ∞ -norm ($\times$); $\mathcal{O}(h^2)$ (- -); $\mathcal{O}(h^3)$ (- · -).

The same analysis can be performed for the drag coefficient (pressure and viscous components). The errors are defined as

$$\text{error}(C_{D,p}) = \frac{\|C_{D,p} - 8/\text{Re}_D\|}{8/\text{Re}_D} \quad (5.71a)$$

$$\text{error}(C_{D,v}) = \frac{\|C_{D,v} - 16/\text{Re}_D\|}{16/\text{Re}_D}, \quad (5.71b)$$

where the coefficients are computed using Eq. (5.30b) for the viscous component and Eqs. (5.40) and (5.44) for the pressure component. As expected, the pressure and viscous component of the force are recovered with a second order accuracy, as shown in Figure 5.4. We also note that both approaches, the standard decomposition and the divergence decomposition of the wall pressure, produce the same results. Again, this conclusion is only valid when the vorticity field is smooth enough. In practice, the standard decomposition gives slightly better results, when the vorticity field is stiff. Finally we observe that the error in the viscous drag coefficient is mainly driven by the error on the surface computation, while the error of the pressure drag coefficient is driven by the error on the evaluation of the vorticity flux.

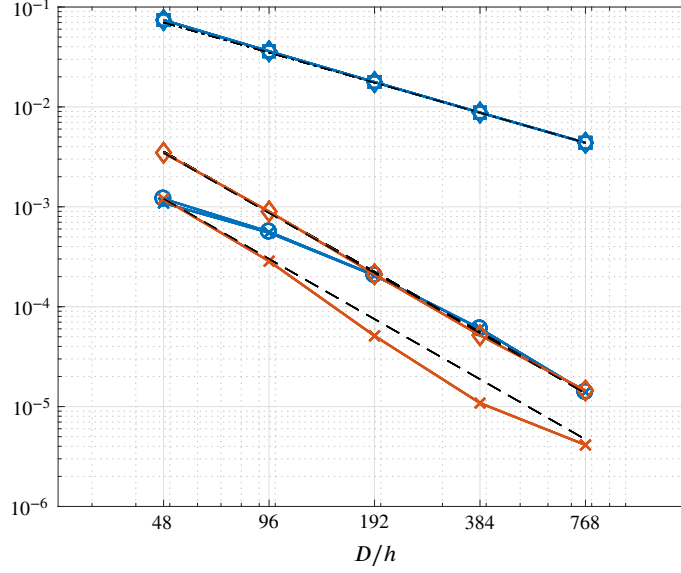


Figure 5.4: **Drag coefficient** - $C_{D,p}$ with Eq. (5.40) ($\omega_{wall} \propto \mathcal{O}(h^2)$): $\square$ and $\omega_{wall} \propto \mathcal{O}(h^3)$: $\circ$); $C_{D,p}$ with Eq. (5.44) ($\omega_{wall} \propto \mathcal{O}(h^2)$): $\diamond$ and $\omega_{wall} \propto \mathcal{O}(h^3)$: $\times$); $C_{D,v}$ with Eq. (5.30b) ($\omega_{wall} \propto \mathcal{O}(h^2)$: $\diamond$ and $\omega_{wall} \propto \mathcal{O}(h^3)$: $\times$); $\mathcal{O}(h)$ (---); $\mathcal{O}(h^2)$ (- -).

5.3.2 Impulsively started flow past a sphere at $Re_D = 300$

In this section, we investigate the behavior of the IIVPM method on a widely studied benchmark, the impulsively flow past a sphere at $Re_D = 300$. We examine the impact of the resolution, D/h , on the quality of various diagnostics, including the drag coefficient and the evaluation of wall quantities. We consider a fixed sphere of diameter $D = 2R$ and the domain dimensions are $2D \times 2D \times 8D$, where the z direction is the stream direction. The center of the sphere is located at $(0, 0, 0)$.

The position of the inflow plane relative to the sphere is determined from the potential flow theory. Without loss of generality, we consider the plane $y = 0$, where the transverse component of the velocity, u , is

$$\frac{u}{U_\infty} = \frac{R^3}{2} \left(\frac{-3xz}{r^5} \right), \quad (5.72)$$

with $r = \sqrt{x^2 + y^2 + z^2}$. This allows to estimate the blocking effect due to the inflow condition. We put the inflow plane at $z = -2.5D$, in order to maintain the perpendicu-

lar velocity under 0.2% of U_∞ . The simulations are performed until $t^* = t U_\infty / D = 5$, and we use a relative tolerance of 10^{-6} in the GMRes iterative solver for Eq. (5.6a).

We remesh every $N_{remesh} = 5$ time steps and the reprojection step is not performed in this section, as it will not influence much the flow since we have a fairly well-resolved wake. A comparative study between the non-reprojected and reprojected fields is presented in §5.4.

Similarly to Chapter 4, we use an adaptive time-step, chosen such that the accuracy criteria, Eq. (4.4), are satisfied

$$\Delta t \|\mathbf{D}\|_1 \leq \text{LCFL}_D \quad \text{and} \quad \Delta t \|\boldsymbol{\omega}\|_1 \leq \text{LCFL}_\omega, \quad (5.73)$$

with $\text{LCFL}_D = 0.3^1$, $\text{LCFL}_\omega = 0.4$, $r = 0.4$. The drag and lift coefficients are taken as

$$C_{D,L} = \frac{F_{D,L}}{\frac{1}{2} \rho U_\infty^2 \pi R^2}. \quad (5.74)$$

while their top-hat filtering, with a length of 30 time steps, is noted as $\overline{C}_D$ and $\overline{C}_L$, respectively.

In Table 5.1, we summarize the different accuracy coefficients, N_δ and Q , for the three considered resolution. We also give the value of $\overline{C}_D$ computed using Eq. (5.28).

D/h	N_δ	Q	$\overline{C}_D$ Noca	$\overline{C}_v$ Eq. (5.30b)	$\overline{C}_p$ Eq. (5.44)	$\overline{C}_p$ Eq. (5.40)
$t^* = 0.25$						
64	3.70	1.27	0.938	0.645	1.593	1.877
96	5.54	0.56	0.890	0.604	0.810	1.075
128	7.39	0.32	0.916	0.597	0.662	0.924
$t^* = 0.50$						
64	3.70	1.27	0.813	0.485	1.166	1.373
96	5.54	0.56	0.782	0.458	0.690	0.889
128	7.39	0.32	0.796	0.454	0.595	0.793
$t^* = 0.75$						
64	3.70	1.27	0.796	0.427	1.101	1.281
96	5.54	0.56	0.770	0.404	0.699	0.874
128	7.39	0.32	0.780	0.401	0.611	0.786

Table 5.1: **Impulsively started flow past a sphere at $\text{Re} = 300$** - Influence of the resolution on the drag and lift coefficient computed using Noca's formula and using the integration of wall quantities (Eq. (5.30b), Eq. (5.40) and Eq. (5.44)).

¹The $\mathbf{D}$ tensor has been computed using Immersed Interface modified stencils

This is compared to the pressure coefficient, $\overline{C_p}$ obtained through the standard decomposition, Eq. (5.40) and the divergence decomposition, Eq. (5.44) and the viscous coefficient, $\overline{C_v}$, obtained through Eq. (5.30b). In Figure 5.5, we present the different terms and the various ways to compute the drag coefficient. From now on, we only show the drag computed using Noca's formula for the highest resolution of $D/h = 128$.

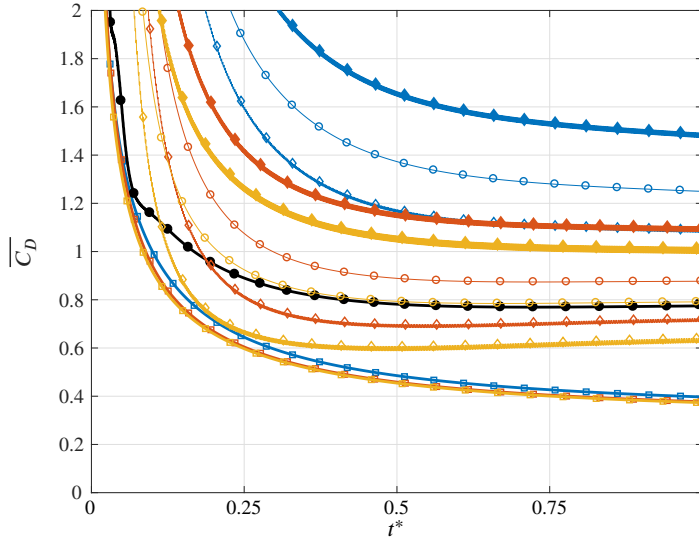


Figure 5.5: **Flow past a sphere at $Re = 300$** - $\overline{C_D}$ using Noca's formula ($\bullet$); $\overline{C_p}$ using the standard decomposition, Eq. (5.40), $D/h = [64 ; 96 ; 128]$ ($\circ$, $\square$, $\diamond$); $\overline{C_p}$ using the divergence decomposition, Eq. (5.44), $D/h = [64 ; 96 ; 128]$ ($\diamond$, $\square$, $\diamond$); $\overline{C_v}$ using Eq. (5.30b) $D/h = [64 ; 96 ; 128]$ ($\square$, $\square$, $\square$); $\overline{C_D} = \overline{C_p} + \overline{C_v}$ using the divergence decomposition Eq. (5.44) $D/h = [64 ; 128]$ ($\diamond$, $\diamond$, $\diamond$).

We note, in Figure 5.5, that the viscous drag coefficient values no longer change when going from $D/h = 96$ to $D/h = 128$. This is expected because it is computed from the wall vorticity and the quality of the evaluation of this wall quantity is the key ingredient of our approach. However, the pressure coefficient is still quite different, hence a very large number of points is required to obtain a drag coefficient close to that computed using Noca's formula.

This observation highlights the main difficulty of any approach which relies on evaluating with high accuracy the wall vorticity derivatives. Here we so far rely on the evaluation of the directional derivatives, i.e. using $\mathbf{n} \cdot \nabla \boldsymbol{\omega}$, while the vorticity flux

is driven by the normal derivative, not necessarily aligned with one mesh direction. Improving the method will likely rely on a better way, still in the IIM framework, of evaluating the wall normal vorticity derivative. Finally, we also note that removing explicitly the divergence part of the wall integral, as done in the divergence decomposition, Eq. (5.44), improves the integral evaluation as it produces a lower pressure drag coefficient.

5.4 Long time simulation of a flow past a sphere at $\text{Re}_D = 300$

In this section, we consider the flow past a sphere at $\text{Re} = 300$ for longer times. The domain considered is a mixed inflow/outflow and unbounded domain, with a size of $4D \times 4D \times 16D$. The resolution of the mesh is chosen to be similar to that used by Ploumhans et al. [30], $D/h = 64$. In order to trigger the expected oscillations, we perturb the transverse velocity at the Strouhal frequency,

$$\frac{u(t^*)}{U_\infty} = 0.1 \sin \left[2\pi \frac{(t^* - t_s^*)}{T_{St}} \right], \quad (5.75)$$

where t_s^* is the start-time, T_{St} the Strouhal characteristic time and $t_s^* + T_{St}$ the end time. In practice, we used $t_s^* = 1$ and an a-priori Strouhal value of $S_t = f D / U_\infty = 0.135$ [32].

In this section, we consider two approaches. The first simulation does not use the reprojection. Indeed the Reynolds number considered here is relatively limited and the grid resolution is sufficient to have a limited divergence in the wake vorticity field. The second simulation performs the reprojection, using a masking function with $h_c = 4$. The masking function is required as the relatively poor resolution near the wall generates strong divergence, which leads to an erroneous reprojected vorticity field, with a spurious radial mode. We also attribute this spurious divergence produced near the interface to the field extension inside the body, through the Hamilton-Jacobi procedure.

The relative GMRes tolerance for the Poisson solver is taken as 10^{-6} and the coefficient for the adaptive time-step are taken as $\text{LCFL}_D = 0.1^2$, $\text{LCFL}_\omega = 0.2$ and $r = 0.4$. For the reprojection, we impose a stronger tolerance, as the GMRes is solved with a relative tolerance of 10^{-10} .

²The $\mathbf{D}$ tensor has not been computed using Immersed Interface tool

5.4.1 Drag and lift coefficients

For the non-reprojected approach, the filtered drag coefficient, $\overline{C_D}$ is shown in Figure 5.6, while the filtered lift coefficient, $\overline{C_L}$ is shown in Figure 5.7. A shorter time comparison between the non-reprojected and the reprojected approaches is shown in Figure 5.8. Both approaches produce essentially the same evolution of the drag and the lift coefficients, and the same final Strouhal number, $St = 0.134$, which are in good agreement with the results obtained by Ploumhans et al. [30] and Mimeau et al. [33].

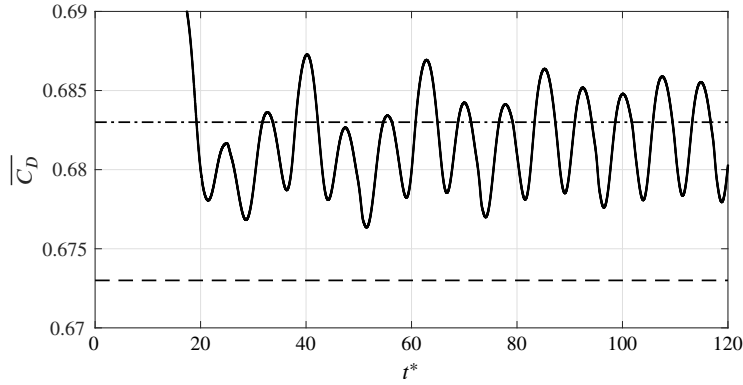


Figure 5.6: **Flow past a sphere at $Re = 300$** - Drag coefficient $\overline{C_D}$: non-reprojected method (—); $C_{D,mean}$ from Ploumhans et al. [30] (---) and $C_{D,mean}$ from Mimeau et al. [33] (- · -).

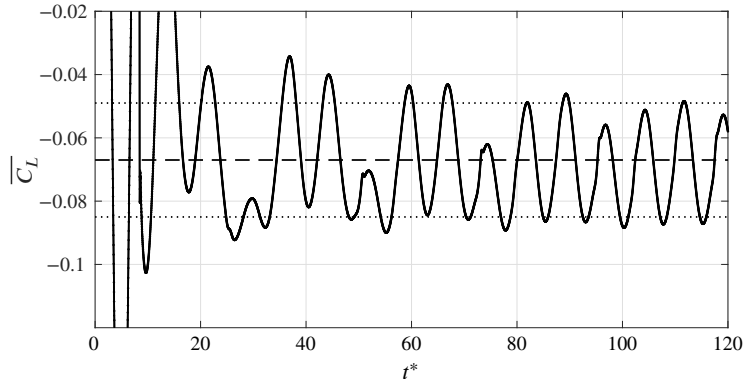


Figure 5.7: **Flow past a sphere at $Re = 300$** - Lift coefficient $\overline{C_L}$: non-reprojected method (—); $C_{L,mean}$ (- · -) and $C_{L,mean} \pm \Delta C_L$ (····) from Ploumhans et al. [30].

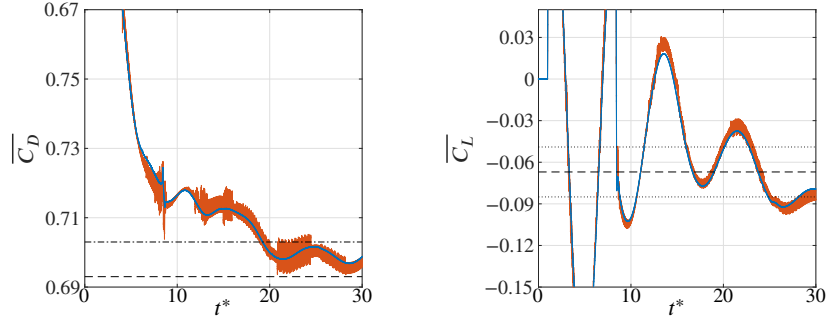


Figure 5.8: **Flow past a sphere at $\text{Re} = 300$** - Drag coefficient (left) and lift coefficient (right): non-reprojected method (—) and reprojected method (—).

The high frequency oscillations of the C_L and C_D curves is due to the numerical evaluation of the first term in Eq. (5.28a),

$$\frac{d}{dt} \int_{\Omega + \Omega_b} \mathbf{x} \wedge \boldsymbol{\omega} dV, \quad (5.76)$$

which oscillates with a period of $N_{\text{remesh}} \Delta t$, due to the periodic reset of the vortex particles (here $N_{\text{remesh}} = 5$). We also observe that a smaller time-step reduces these oscillations. Moreover, the highest vorticity values are located at the body interface, where the resolution is relatively poor, as compared to the quality factors defined in Chapter 4 ($N = 3.7$ and $Q = 1.27$). This also affects the quality of the evaluation of the integral. Finally, this evaluation could likely be improved in the vicinity of the interface by considering the boundary points as truncated integration volumes.

5.4.2 Computational time and GMRes iterations

The computational time elapsed in each major step of the simulation is shown in Figure 5.9. We notice that the time spent in the Poisson solver (the bulk solver, the GMRes iterations and the final Poisson solver) represents around 70% of the computational time. The iterated Poisson solver (i.e. the GMRes linear solver) represents only 10% of the total computational time, thanks to the use of the subdomain. We also note that most of the time is spent in the bulk solver, i.e. solving Eq. (5.3a) and in the final Poisson solver, i.e. solving Eq. (5.6b). These timings highlight the importance of having a very efficient and fast bulk Poisson solver. Finally, the effect of the recycling, which reduces the computational cost, is especially noticeable at the early times of the simulation.

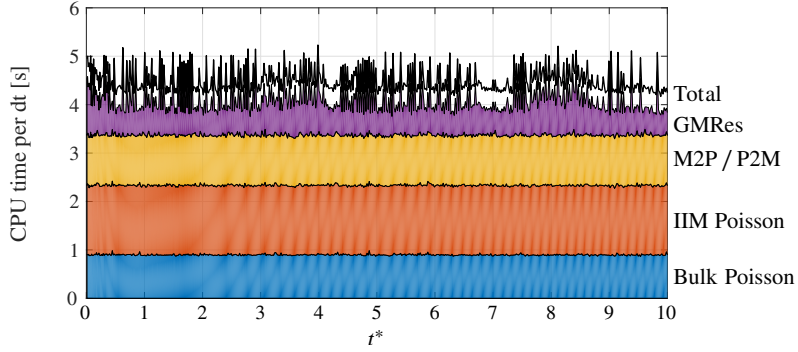


Figure 5.9: **Flow past a sphere at $Re = 300$** - Total run time per time step for each operation of the approach with reprojection.

The number of iterations is shown in Figure 5.10 for both the reprojected and non-reprojected methods. As expected, we reduce the number of iterations through the

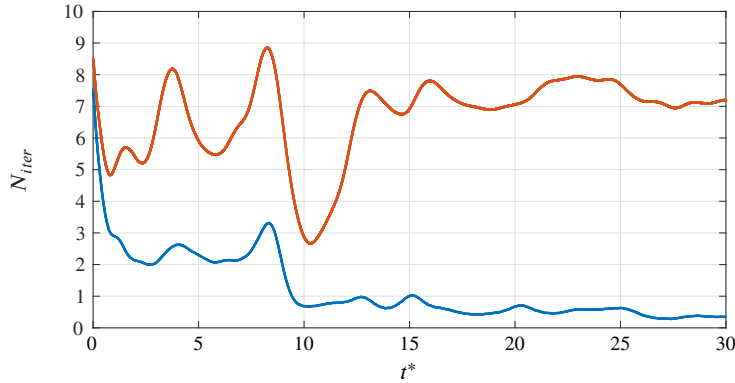


Figure 5.10: **Flow past a sphere at $Re = 300$** - Filtered number of iterations required by the Immersed Interface Poisson solver: non-reprojected approach (—) and reprojected approach (—).

use of recycling [61, 115], which provides a mean number of iterations less than 1 for the non-reprojected approach. We also note that the reprojected approach requires more iterations due to the periodic imposition of the divergence-free vorticity (here $N_{remesh} = 4$, $N_{reproj} = 20$). Let us stress that Figure 5.10 shows the number of iterations for the whole time-step, i.e. the sum of the iterations required for all sub-steps of the third order Runge-Kutta scheme used here.

5.4.3 Comparison with the Iterative Penalization VPM method

Extension of the Iterative Penalization VPM method to higher dimensions

We here briefly describe the generalization of the Iterative Penalization Vortex Particle-Mesh method (IPenVPM) to three dimensions. There is no major change to the methodology already presented in two dimensions and we here summarize its extension, for the sake of clarity of the manuscript.

Given the vorticity distribution on the particles, $\boldsymbol{\omega}_p^n$, we first compute the particle to mesh interpolation to get the non-corrected vorticity field, $\tilde{\boldsymbol{\omega}}^n$, and its corresponding velocity $\tilde{\mathbf{u}}^n$. This vorticity field is then corrected by a vorticity increment, $\delta\boldsymbol{\omega}$, which is computed as the solution of the following linear system,

$$A(\delta\boldsymbol{\omega}) = \nabla \wedge (\chi \delta\mathbf{u}) = \nabla \wedge [\chi (\mathbf{u}_b^n - \tilde{\mathbf{u}}^n)] , \quad (5.77)$$

where $\delta\mathbf{u}$ is the correction in velocity corresponding to the vorticity correction $\delta\boldsymbol{\omega}$. The Eq. (5.77) is solved using a recycling GMRes method [115], as for the Immersed Interface Poisson problem. Let us note, that, at convergence of the GMRes procedure, the corrected vorticity,

$$\boldsymbol{\omega}^n = \tilde{\boldsymbol{\omega}}^n + \delta\boldsymbol{\omega} , \quad (5.78)$$

is such that the induced velocity is given by $\mathbf{u}^n = \tilde{\mathbf{u}}^n + \delta\mathbf{u}$ and

$$\nabla \wedge [\chi (\mathbf{u}_b - \tilde{\mathbf{u}}^n - \delta\mathbf{u})] = 0 \Rightarrow \mathbf{u}^n = \tilde{\mathbf{u}}^n - \delta\mathbf{u} = \mathbf{u}_b^n . \quad (5.79)$$

Here we have a fixed body, hence $\mathbf{u}_b = 0$.

We then correct the vorticity on the mesh and on the particles. We also correct the velocity on the mesh. Then, the right hand side of Eq. (6), i.e. the diffusion and stretching terms, is evaluated and interpolated back on the particles in a classical way, considering these corrected fields.

Drag coefficient comparison

We here compare the drag coefficient and the computational cost of both approaches. We used the same setup, the same relative tolerance for the GMRes, 10^{-6} , and the same subdomain. In other words, we here use the same tools and the same computational resources and compare the accuracy and the cost per time-step of both approaches.

The different drag coefficients obtained using both approaches are shown in Figure 5.11. We notice a small mismatch between the mean values. The drag coefficient obtained using the iterative penalization approach is closer to the values obtained by Mimeau et al. [33], while that obtained using the Immersed Interface approach is closer

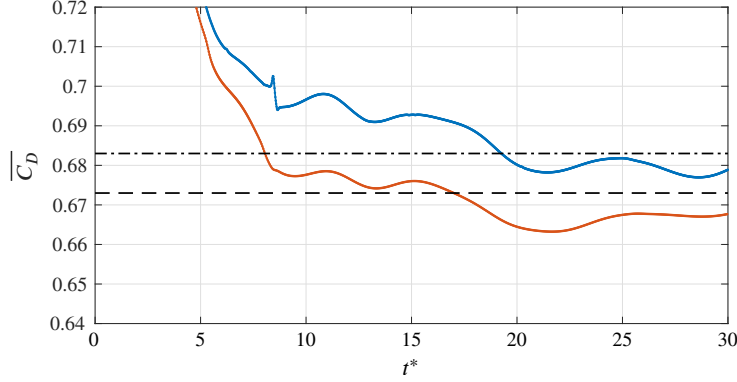


Figure 5.11: **Flow past a sphere at $Re = 300$** - Comparison of the filtered drag coefficient obtained using the IIVPM method (—) and using the Iterative Penalization VPM method (—), $C_{D,mean}$ from Ploumhans et al. [30] (---) and $C_{D,mean}$ from Mimeau et al. [33] (---).

to that obtained by Ploumhans et al. [30]. This is expected since the method used by Ploumhans et al. [30] and the Immersed Interface approach are sharp at the wall, while both penalization methods correspond to a smearing of the interface, and are thus less accurate (yet also more robust/forgiving on the resolution required).

Cost comparison

The computational cost per time-step required by the two methodologies applied is compared in Figure 5.12. The Immersed Interface approach is around 1.5 times cheaper, while involving more operations for the computation of the diffusion and stretching terms. The iterative penalization approach indeed requires more iterations to converge toward the solution as it needs to produce a flip-flopping solution inside the body, see Figure 5.13. This highly oscillating mode is very slow to converge, yet mandatory to reach the desired condition, that $\nabla \wedge [\chi (\mathbf{u}_b - \tilde{\mathbf{u}}^n - \delta \mathbf{u})] = 0$. The unknowns inside the body limit the convergence of the iterative penalization VPM method, which is not the case for the Immersed Interface VPM method as it iterates on the interface unknowns only.

Field comparison

In Figure 5.13, we here propose a comparison of the vorticity, $(\omega_y D/U_\infty)$, around the sphere in the plane $y = 0$, at $t^* = 25$. The Immersed Interface VPM method produces a sharp vorticity at the boundary, while the penalization approach, as already shown

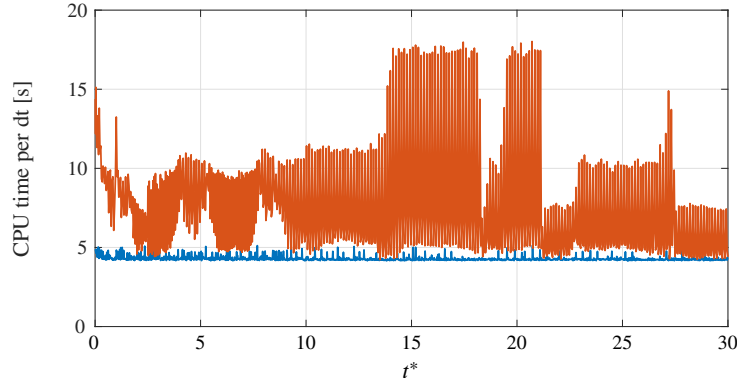


Figure 5.12: **Flow past a sphere at $Re = 300$** - Comparison of the computational cost obtained using the Immersed Interface Vortex Particle-Mesh method (—) and using the Iterative Penalization Vortex Particle-Mesh method (—).

in Chapter 2, produces a vorticity smeared around the interface and with a strong flip-flopping vorticity inside the sphere, and actually stronger near the body surface.

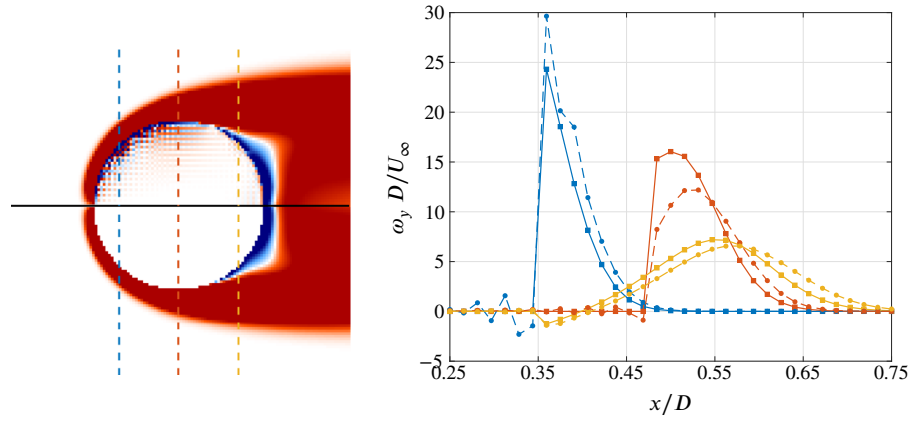


Figure 5.13: **Flow past a sphere at $Re = 300$** - Comparison, at $t^* = 25$, of the vorticity field $\omega_y D/U_\infty$: (left) slice in the field $\omega_y D/U_\infty$ at $y/D = 0$ for the iterative penalization VPM method (up) and the IIVPM method (down); (right) cut in that slice at various $z/D = [-0.35; 0; 0.35]$ for the iterative penalization VPM method (—) and the IIVPM method (—).

Flow visualization

We here propose a few flow visualizations for the non-reprojected Immersed Interface approach. Among others, there are typically two criteria to define and visualize a complex vortex structures: the λ_2 and the Q criteria [119, 120]. The Q -criterion is defined as

$$Q = \frac{1}{2} [\mathbf{\Omega}_{ij}\mathbf{\Omega}_{ij} - \mathbf{D}_{ij}\mathbf{D}_{ij}] > 0 . \quad (5.80)$$

The Q -criterion defines a vortex as the region where the rotation rate tensor, $\mathbf{\Omega}$ is greater than the strain rate tensor, $\mathbf{D}$, i.e. $Q > 0$. The second criterion, the λ_2 -criterion, is defined as the second eigenvalue, $\lambda_1 \geq \lambda_2 \geq \lambda_3$ of the tensor

$$(\mathbf{D}^2 + \mathbf{\Omega}^2) = (D_{ik}D_{kj} + \Omega_{ik}\Omega_{kj}) . \quad (5.81)$$

As proposed by Jeong and Hussain [119], a vortex structure is the region with two negatives eigenvalues, thus defined as $\lambda_2 < 0$. Finally, let us note that both λ_2 and Q criteria have the dimension of $1/s^2$.

In Figs. 5.15 and 5.16, we show the dimensionless λ_2 -criterion ($\lambda_2 D^2 / U_\infty^2$) for different times over one period. The asymmetry of the λ_2 field is the signature of a non-zero mean lift coefficient, as shown in Figure 5.7. The same oscillation can be retrieved in the wall vorticity, $(|\boldsymbol{\omega}_{wall}| D / U_\infty)$, shown in Figure 5.14. As expected, the minimum wall vorticity is found in $\theta > 0$, which is also a consequence of the non-zero lift coefficient. Finally, in Figure 5.17, we present some visualization of the magnitude of the vorticity field for various times.

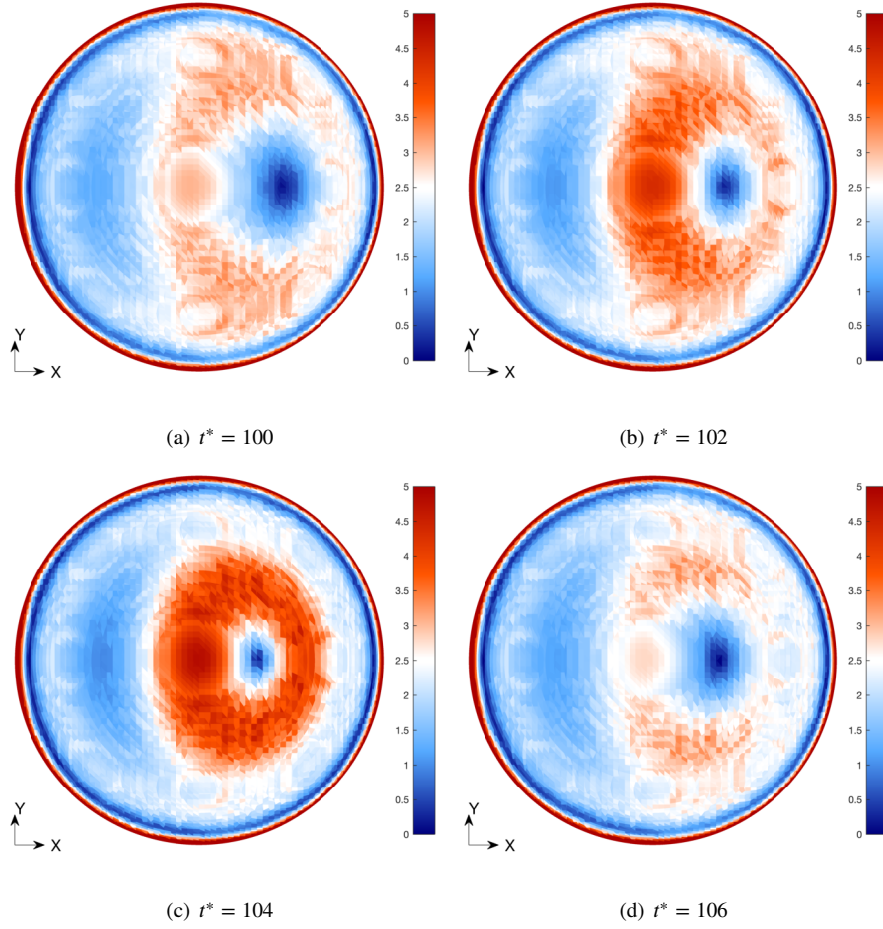
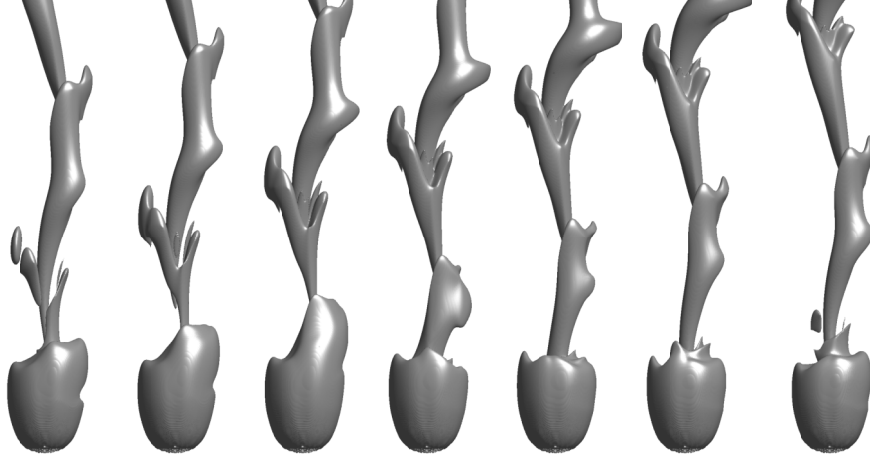
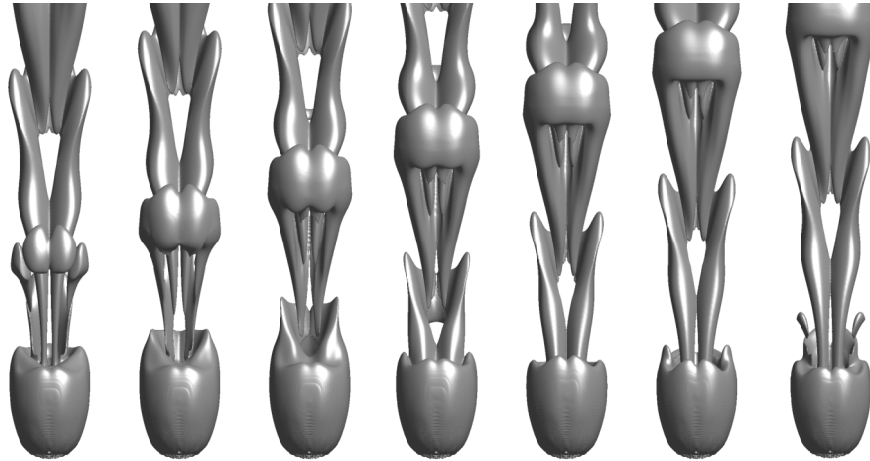


Figure 5.14: **Flow past a sphere at $\text{Re} = 300$** - wall vorticity magnitude ($(|\omega_{wall}|D)/U_\infty$), when looking at the back of the sphere (i.e. in the $-z$ direction), at various times, for the non-reprojected IIVPM method.



(a) $t^* = 100$ (b) $t^* = 101$ (c) $t^* = 102$ (d) $t^* = 103$ (e) $t^* = 104$ (f) $t^* = 105$ (g) $t^* = 106$

Figure 5.15: **Flow past a sphere at $Re = 300$** - Side view of $\lambda_2 = -0.001$ criterion at various times over a period, for the non-reprojected IIVPM method.



(a) $t^* = 100$ (b) $t^* = 101$ (c) $t^* = 102$ (d) $t^* = 103$ (e) $t^* = 104$ (f) $t^* = 105$ (g) $t^* = 106$

Figure 5.16: **Flow past a sphere at $Re = 300$** - Bottom view of $\lambda_2 = -0.001$ criterion at various times over a period, for the non-reprojected IIVPM method.

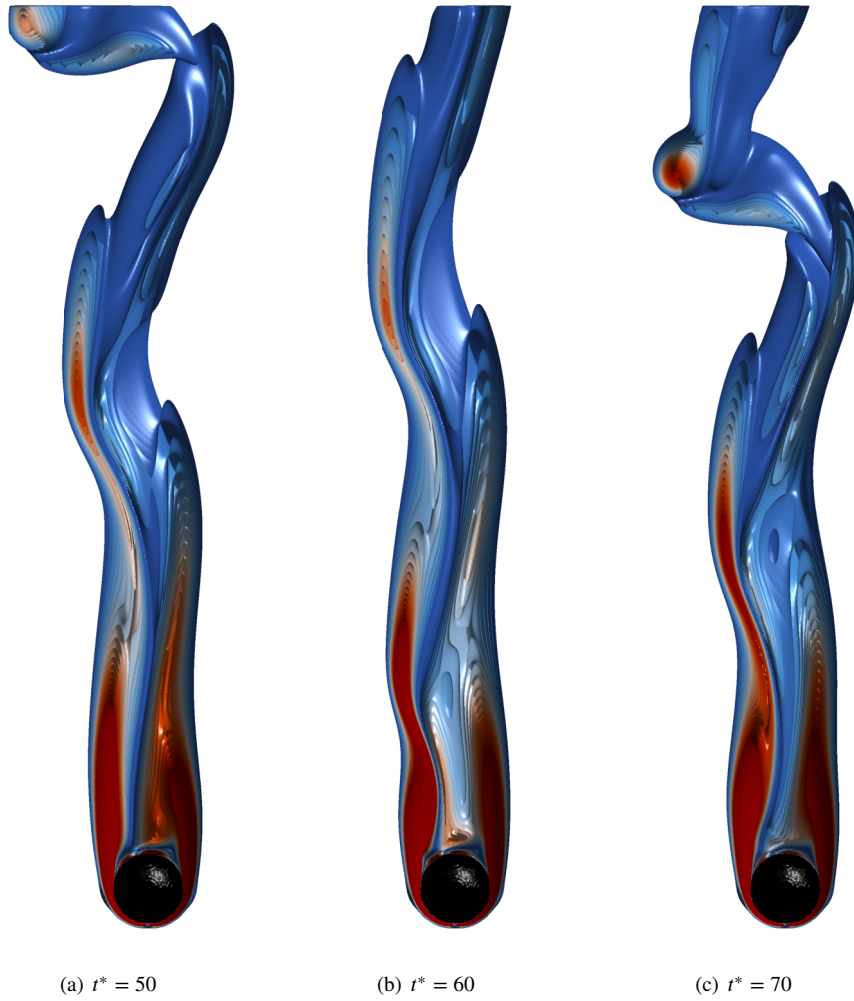


Figure 5.17: **Flow past a sphere at $\text{Re} = 300$** - iso-surface of vorticity magnitudes, $(|\omega|D/U_\infty)$, at various times for the non-reprojected approach.

5.5 Impulsively started flow past an ellipsoid at $Re_c = 300$

Finally, we consider the application of the proposed approach to the impulsively started flow past an ellipsoid with an angle of attack of 10 deg. The ellipsoid is taken with a span b , a chord $c = b/2$ and a thickness $t = b/8$. The Reynolds number is

$$Re_c = \frac{cU_\infty}{\nu} = 300 \quad (5.82)$$

and the computational domain is $3b \times 2b \times 12b$, while the resolution is $b/h = 192$, thus $t/h = 24$. This is thus “quite challenging” simulation as we only have 24 grid points where the thickness is minimum. The radius of curvature of the leading edge at the ellipsoid center is $R_{le} = \frac{b^2}{c}$. Using that radius, we obtain the following quality criteria: $N = 0.346$ and $Q = 144.34$.

The drag and lift coefficients are taken as

$$C_{D,L} = \frac{F_{D,L}}{\frac{1}{2}\rho U_\infty^2 \pi bc} \quad (5.83)$$

while their filtered values are noted as $\overline{C_D}$ and $\overline{C_L}$, respectively. We run the simulation until convergence of the drag and lift coefficients: at $t^* = tU_\infty/b = 4$, we obtain $\overline{C_D} = 0.316$ and $\overline{C_L} = 0.223$, see Figure 5.18. As the resolution at the body surface is quite poor, the early time evolution of $\overline{C_D}$ and $\overline{C_L}$ is likely unphysical. We also present some flow visualizations at various times in Figure 5.19.

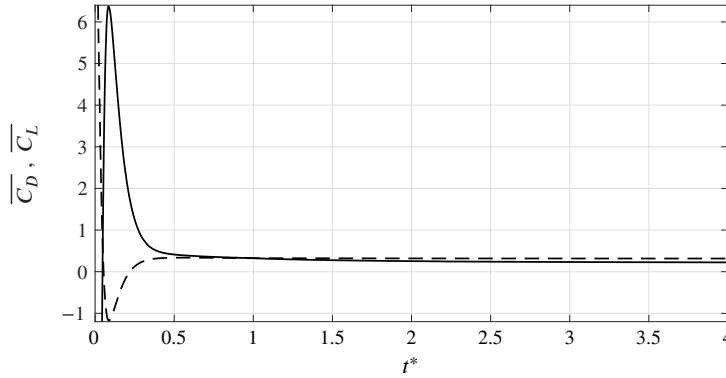


Figure 5.18: **Flow past an ellipsoid at $Re_c = 300$** - filtered lift coefficient $\overline{C_L}$ (—) and filtered drag coefficient $\overline{C_D}$ (- -).

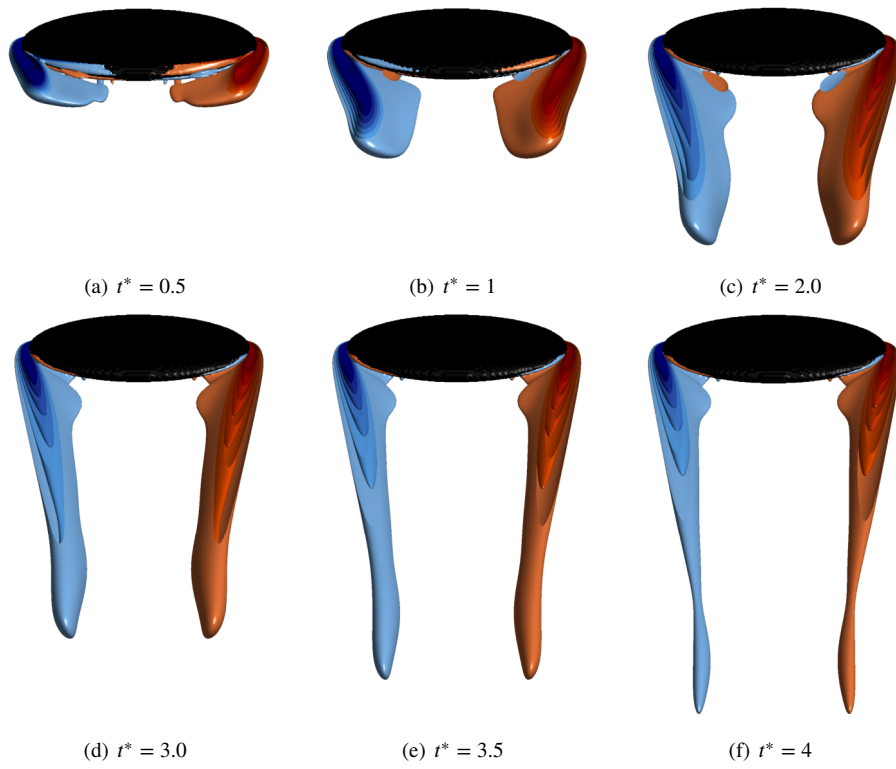


Figure 5.19: **Flow past an ellipsoid at $\text{Re}_b = 600$** - iso-surface of vorticity magnitude, $|\omega|b/U_\infty$, at various times using the non-reprojected IIVPM method.

5.6 Conclusions

In this chapter, we present an extension to 3D of the method introduced in Chapter 4. This Immersed Interface Vortex Particle-Mesh approach relies on the evaluation of the wall vorticity, using the Immersed Interface jumps. This wall quantity is used afterwards as a boundary condition in the evaluation of the diffusion and stretching term during the time step. Using the Immersed Interface tool, we are also able to compute wall quantities and therefore the pressure and viscous drag coefficients. The surface integrals are computed as a sum, over the biased points, of a small equivalent surface, as proposed by Smereka [118]. This approach is very convenient as it does not depend on the neighbouring intersections neither on any solution field.

We validate this approach using a Stokes flow past a sphere and we prove the convergence at the expected order to the desired quantities: body surface, wall vorticity and wall vorticity fluxes, pressure and viscous drag coefficient. We also validate the whole Immersed Interface VPM method on the benchmark of the impulsively started flow past a sphere at $Re = 300$ for long times. We retrieve the lift and drag coefficient already obtained by Ploumhans et al. [30] and show a small offset with the results obtained using a penalization approach [33].

Using the same numerical setup and the same computational resources, we also compare the computational cost per time-step of the proposed method to the cost required by the iterative penalization technique. We highlight the computational gain obtained (1.5 times cheaper) and the smoothness of the vorticity field at the boundary compared to the latest. Finally, we were also able to perform some more challenging simulation as the impulsively started flow past an ellipsoid at $Re_c = 300$.

This work also highlights two inherent problems of our approach. First, it shows a lack of accuracy while computing wall derivatives and especially while evaluating the force exchange at the boundary, using wall quantities. The viscous contribution to the drag coefficient is easily captured as it relies on the wall vorticity. An accurate pressure drag coefficient, however, comes at a much higher resolution. This is also a perfect example of the limitation of the uniform grid approach. Indeed the resolution required to reach an accurate wall quantities computation is high and difficult to achieve in a uniform grid framework. Second, a non-physical and very strong divergence is produced at the wall. We attribute this high divergence value to the extrapolation of the vorticity inside the body. Indeed, the Hamilton-Jacobi approach we use does not offer any warranty about the divergence. We ambition to solve these two drawbacks in a future work, by using a multiresolution framework and by adding physics and body dynamics information to the Immersed Interface jump computation.

A Multiresolution Poisson Solver for Immersed Interface Applications

This chapter is based on the results obtained during a research stay at
ETH Zürich - CSE Lab, under the supervision of Pr. Siddhartha Verma
and Pr. Petros Koumoutsakos.

6.1 Introduction

The Poisson equation is an ubiquitous and deeply investigated topic in computational physics. Still, as highlighted in Chapters 4 and 5, taking an immersed obstacle into account requires a very fine grid at the interface compared to the bulk requirements, hence overwhelming the computational cost associated to the Poisson solver. In Chapter 3, we presented an efficient formulation of the Poisson problem around arbitrary geometries using a uniform grid. This approach relies on the Sherman-Morrison-Woodbury formula to decompose the Poisson problem into a global Poisson equation to be solved in the entire domain and a local Poisson problem to be solved in the vicinity of the interface only. In this chapter, we propose a “body-free” unbounded multiresolution Poisson solver to be used as a global Poisson solver within this SMW approach.

There are multiple ways to solve an unbounded problem, typically divided into two families of methods: the FFT-based and the FMM-based approaches. Among them, we mention the Hockney and Eastwood [15] FFT-based approach, which doubles the computational domain and its recent applications, see Chapter 1 for more details. The complexity of this family of methods is $C_{FFT} \mathcal{O}(N) \log(N)$, where C_{FFT} is a constant

driven by the implementation. The FFT-based algorithms are well-suited for uniform and dense domains, i.e. the source distribution fulfils the whole computational domain. On the other hand, the Fast Multipole Method-based (FMM) family, compresses the source terms of far away targets points. We here allude to the standard FMM [121–123] as well to the kernel independent version [124] that recycles the method originally proposed by Anderson [125]. This family of methods achieves a complexity of $C_{FMM}\mathcal{O}(N)$, where C_{FMM} is an implementation-driven constant. This type of method is well-suited for non-uniform and sparse domains, i.e. when the sources do not fill the whole computational domain. Moreover, let us note that, in practice, we have $C_{FFT} \ll C_{FMM}$. We refer to the work of Gholami et al. [126] for a detailed comparison between FMM and FFT-based methods.

A work by Hejlesen [127] presents a multiresolution Poisson solver using FFT-based approach, hence considering only one rectangular patch per level. This approach, well suited for a continuous forcing approach, e.g. the iterative penalization approach [60, 61], has not been tailor-made for discrete forcing approaches, such as the Immersed Interface method. Indeed, the local problem solved during the Immersed Interface Poisson equation contains unknowns along the immersed surface only and does not treat the interior of the body.

In this work, we propose a 2D multiresolution Poisson solver which allows multiple patches at a given level. This solver takes inspiration from the wavelet framework and the fast multipole approach. In §6.2 we present the proposed method and validate its convergence in §6.3. We then close this chapter by a conclusion on the proposed approach.

6.2 Methodology

In this section we describe the proposed Poisson solver to solve an unbounded problem

$$\nabla_h^2 u = f, \quad (6.1)$$

using a wavelet-based multiresolution structured grid. We refer the reader to Appendix E for more details on the wavelet approach. Since the goal of this chapter is to propose an Immersed Interface compatible Poisson solver, we consider the 2D standard 3-point stencil of Laplacian, ∇_h^2 and the associated LGF, as presented in Chapter 1. Let us stress that the presented approach is directly extendable to 3D and to any other Green's function.

We consider a block-structured grid, composed of blocks, B_i^l , at different levels $l \in [l_{min}; l_{max}]$ (see Figure 6.1). We separate the blocks per level and write the part

of the domain spanned by the blocks of a same level l by $\mathcal{B}^l \triangleq \cup_i \mathcal{B}_i^l$. Let us note that the computational domain is then defined by $\cup_l \mathcal{B}^l$ and $\mathcal{B}^l \cap \mathcal{B}^k = \emptyset$ for $l \neq k$. Additionally, we use the Deslauriers and Dubuc [128] wavelet interpolation of order p , noted by the operator $\mathcal{I}_{l \rightarrow l \pm 1}^p(\cdot)$, to interpolate a grid's level, l , to a finer/coarser level. The interpolation operator through several levels is obtained by a successive application of the operator on itself. For example, the interpolation operator from l to $l - 3$ is given by

$$\mathcal{I}_{l \rightarrow l-3}^p = \mathcal{I}_{l-2 \rightarrow l-3}^p (\mathcal{I}_{l-1 \rightarrow l-2}^p (\mathcal{I}_{l \rightarrow l-1}^p(\cdot))) \quad . \quad (6.2)$$

For the sake of compactness, we slightly abuse the notation and denote $\mathcal{I}_{l \rightarrow l \pm k}^p(\cdot)$ by $[\cdot]_{l \pm k}$, where we neglect the p index and the level of origin as they can be easily deduced from the context of application. Finally, considering a level l , we note all the finer levels as l^+ and all the coarser levels will be noted l^- .

The algorithm we propose is composed of two steps: the downward pass, that goes from l_{max} to l_{min} (from the finest to the coarsest level) and the upward pass that goes the opposite direction. For a given level (and therefore a given cell size h), we have to take into account three different source contributions: the sources that are on the coarser levels, f_{l^-} , the sources at the same level, f_l , and the sources that are represented at the finer levels, f_{l^+} .

1. **The downward pass** works in a recursive way starting from the finest level, l_{max} , as presented in Figure 6.2. For each level, l , it will compute the interpolation of the source term, $\tilde{f}_l$, to the next coarser level: $[\tilde{f}_l]_{l-1}$ where $\tilde{f}_l = f_l + [\tilde{f}_{l+1}]_l$. The interpolated source term is decomposed into two parts: the sources that are interpolated from a finer level, $[\tilde{f}_{l+1}]_l$ and the sources which belong to the current level, f_l . After the end of the downward pass, every source term is present at the coarsest level, l_{min} , in a compressed version: $\tilde{f}_{l_{min}} = [\tilde{f}_{l^+}]_{l_{min}} + f_{l_{min}}$, see Figure 6.3.
2. **The upward pass** effectively solves the problem, starting at the coarsest level, see Figure 6.4. For a given level, it will first interpolate the solution from its previous level, $[u_{l-1}]_l$. This interpolated solution contains the contribution of the sources at the coarser level $\tilde{f}_{l-1} = [\tilde{f}_l]_{l-1} + f_{l-1}$, where the term $[\tilde{f}_l]_{l-1}$, already taken into account in the solution, is a compressed version of the original $\tilde{f}_l$. Therefore, we need to correct it by adding its details, $\delta \tilde{f}_l$, not already taken into account by the coarser level. We identified two different ways to do it. First, we could compare the sources terms and define the increment as

$$\delta \tilde{f}_l = \tilde{f}_l - [\tilde{f}_{l-1}]_l = \tilde{f} - [[\tilde{f}_l]_{l-1}]_l \quad , \quad (6.3)$$

where $[f_{l-1}]_l$ is not taken into account since $f_{l-1} \notin \mathcal{B}^l$. Second, we could use the representation at the current level of the sources effectively solved by the coarser level, i.e. $\nabla_{h_l}^2 [u_{l-1}]_l$:

$$\delta \tilde{f}_l = \tilde{f}_l - \nabla_{h_l}^2 [u_{l-1}]_l, \quad (6.4)$$

where h_l is the resolution associated to the considered level.

We prefer the second, more general, approach as it takes into account the fact that the kernel may depend on the resolution. Let us for example consider the application of this method to the lattice Green's function, G_h , that depends on the resolution of the considered level. We associate the cell size h to the level l and $2h$ to $l-1$. The coarser level solution, interpolated on l , has been resolved using G_{2h} . However, the solution will be corrected with another LGF, G_h . If we consider the first approach, we get the following relation,

$$u_l = G_h \star \delta \tilde{f}_l + [G_{2h} \star \tilde{f}_{l-1}]_l \quad (6.5)$$

$$= G_h \star (\tilde{f}_l - [\tilde{f}_{l-1}]_l) + [G_{2h} \star \tilde{f}_{l-1}]_l \quad (6.6)$$

$$= G_h \star \tilde{f}_l + [G_{2h} \star \tilde{f}_{l-1}]_l - G_h \star [\tilde{f}_{l-1}]_l, \quad (6.7)$$

which is equal to $G_h \star \tilde{f}_l$ if $[G_{2h} \star \tilde{f}_{l-1}]_l = G_h \star [\tilde{f}_{l-1}]_l$, which is not the case for LGF kernels. Indeed, let us consider the case where $\tilde{f}_{l-1} = \delta$: the equality is not verified since $[G_{2h}]_l(\mathbf{m}) = G_h(\mathbf{m}/2) \neq G_h(\mathbf{m})$ for small $|\mathbf{m}|$.

The last missing piece of the method is the computation, at each level, of the unbounded convolution between the source term and the chosen kernel. The two usually considered algorithms are the FFT-based convolution and the FMM methods. In this work, we recommend a combination of the two methods: FFT-based accelerated convolution for dense levels (the root level, is dense, by definition) and FMM-based compression for sparser levels, e.g. the local solver for Immersed Interface applications. Indeed the FFT-based approach will be more efficient if the level is dense (highly connected and "rectangular-like" support, as in Hejlesen [127]). On the contrary, when considering very sparse source distributions, it will consider many useless unknowns and null sources. In that configuration we recommend the use of the (Ki)FMM, which will take advantage of the source compression. Nevertheless, the final choice is left to the reader as it is driven by the domain configuration considered in the envisioned application.

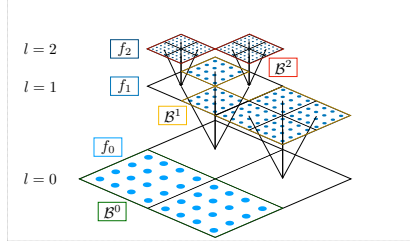


Figure 6.1: Example: initial state of the sources on the grid, every source term $f_i \in B^i$.

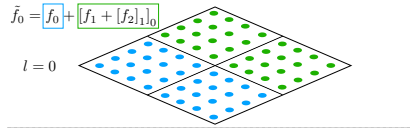


Figure 6.3: Example: upward pass, the lowest level contains a compression of the sources from every level.

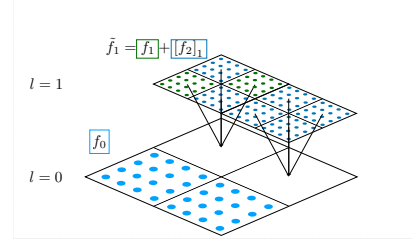


Figure 6.2: Example: downward pass through the grid, the source terms are interpolated from a level to a coarser one.

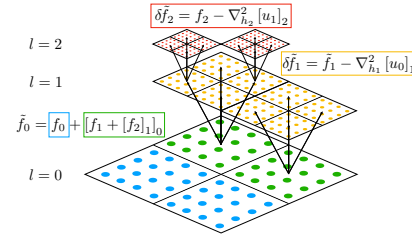


Figure 6.4: Example: upward pass through the grid, we compute the increment of sources from one level to another and solve the unbounded Poisson problem associated.

6.3 Validation: 2D unbounded Gaussian source-term

In this section, we present convergence results on an analytical benchmark of multiple Gaussian source. We consider a computational domain $[0 ; L] \times [0 ; L]$ with the source term defined as $\omega = \sum_{i=0}^4 \omega_i$ and

$$\omega_i = \frac{\Gamma_i}{2\pi} \exp\left(-\frac{r_{c_i}^2}{2\sigma^2}\right), \quad (6.8)$$

with $\sigma = 0.05L$, $r_{c_i} = \sqrt{(x - x_{c_i})^2 + (y - y_{c_i})^2}$ and $[x_{c_1} ; y_{c_1}] = [0.3 L ; 0.5 L]$, $[x_{c_2} ; y_{c_2}] = [0.7 L ; 0.5 L]$, $[x_{c_3} ; y_{c_3}] = [0.4375 L ; 0.5625 L]$, $[x_{c_4} ; y_{c_4}] = [0.5625 L ; 0.4375 L]$. To ensure a zero-circulation we consider $\Gamma = -\Gamma_1 = \Gamma_2 = -\Gamma_3 = \Gamma_4$ (see figure 6.5).

We solve the unbounded Poisson equation,

$$\nabla_h^2 \psi = -\omega, \quad (6.9)$$

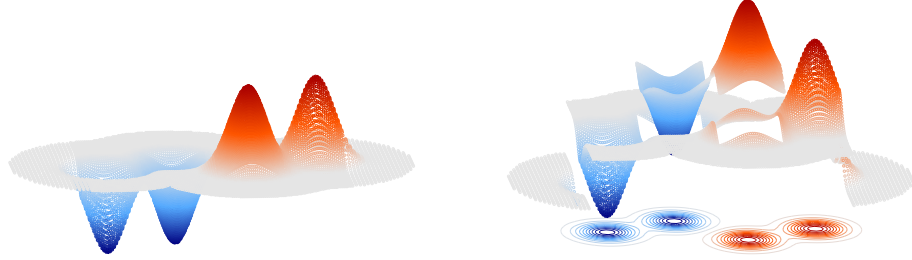


Figure 6.5: Validation - source: (left) considered benchmark source term; (right) contour and representation of the source at different levels ($l = 0, l = 1, l = 2$) shifted upward for better visualisation.

where the analytical solution is known

$$\psi_{exact} = \sum_{i=0}^4 \frac{-\Gamma_i}{4\pi} [\log(\rho_i^2) + E_1(\rho_i^2)] \quad , \quad (6.10)$$

with $\rho_i = r_{c_i}/\sigma$.

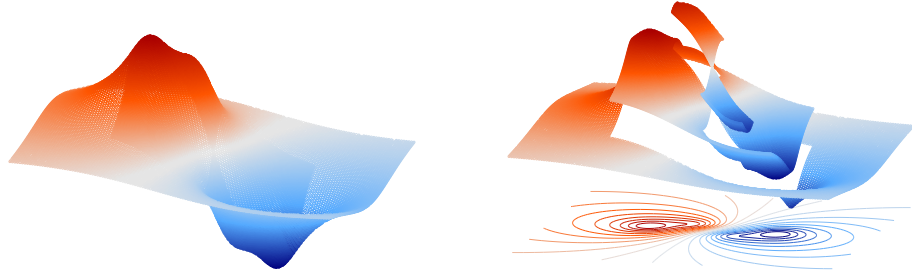


Figure 6.6: Validation - solution: (left) considered benchmark solution; (right) contour and representation of the solution at different levels ($l = 0, l = 1, l = 2$) shifted upward for better visualisation.

To solve this equation, we use a three level grid represented in Figure 6.5 and Figure 6.6. Furthermore, we define the local error as $e(\mathbf{x}) = \psi(\mathbf{x}) - \psi_{exact}(\mathbf{x})$, in the bulk, and the associated dimensionless 2-norm and infinite norm, respectively as,

$$E_2 = \frac{1}{\Gamma L^3} \sqrt{\sum_{\mathbf{x} \in B^l} e^2(\mathbf{x}) h^2} \quad \text{and} \quad E_\infty = \frac{1}{\Gamma L^2} \max_{\mathbf{x} \in B^l} |e(\mathbf{x})| \quad . \quad (6.11)$$

In addition, we define N , the global resolution of a mesh configuration as $N = \max_{B^l} \frac{1}{h_l}$ and finally, define the local convergence orders, r_∞ and r_2 as

$$[r_q]_i = \log([E_q]_i / [E_q]_{i-1}) / \log(N_{i-1}/N_i) \quad , \quad (6.12)$$

with $q = \infty$ and $q = 2$, respectively. The convergence results for the proposed method using the FFT and KiFMM solvers (at each level, including the coarsest level) are shown in Figs. 6.7 and 6.8, respectively.

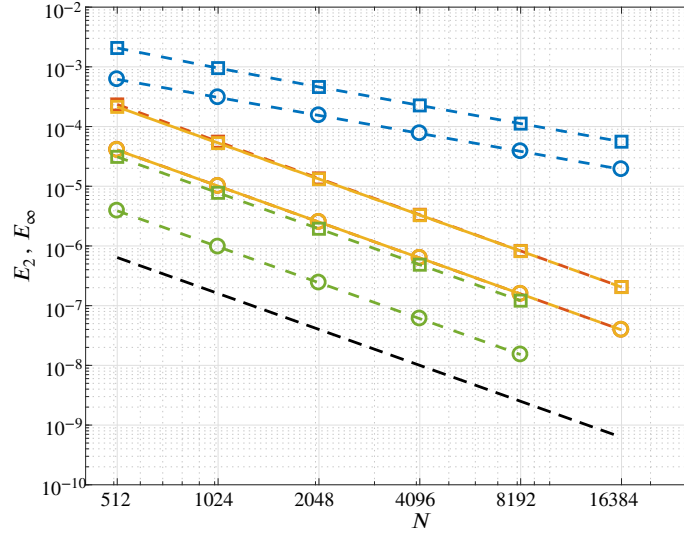


Figure 6.7: Convergence study using FFT unbounded solver at each level for E_2 ($\square$) and E_∞ ($\circ$): interpolation order 2 (—), 4 (—), 6 (—); Uniform resolution (FFT) (—) and convergence $\mathcal{O}(h^2)$ (—).

We first notice that the use of a second order wavelet interpolation leads to a first order convergence rate. We explain this behavior by an inconsistency between the wavelet interpolation and the LGF used to solve the Poisson equation at each level. Using a fourth order or sixth order interpolation wavelet does not influence the convergence anymore and provides the same level of error. More interestingly, we observe a clear difference, between the behavior of the FFT-based method and the KiFMM convolution. In the case of the FFT-based solver, the convergence is maintained for a large number of unknowns. In the case of the KiFMM-based method, the convergence reaches a plateau. This saturation is due to the error inherent to the KiFMM algorithm, which truncates the singular value decomposition while solving the inverse problem (referenced as L2M in [124]). In this method, we reinforce this truncation error while computing the increment of source $\delta \tilde{f}$ using the Laplacian expression as in Eq. (6.4). The whole convergence procedure may also be compared to the same benchmark run with a uniform resolution. The latter is of course more accurate but uses much more computational resources.

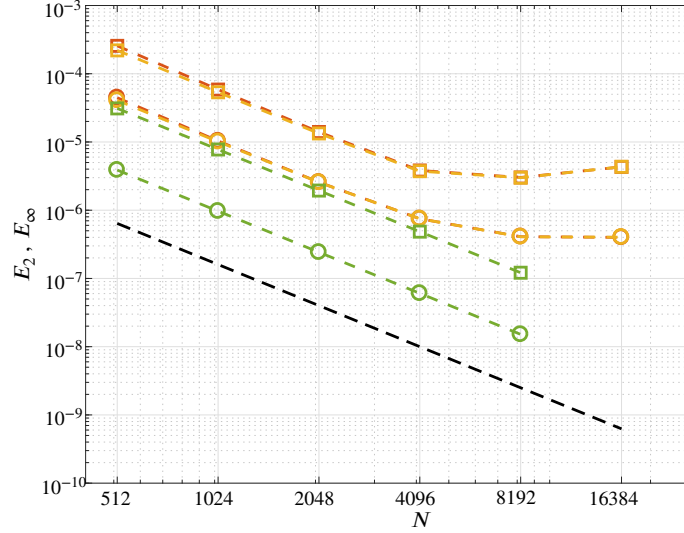


Figure 6.8: Convergence study using KiFMM solver at each level for E_2 (□) and E_∞ (○): interpolation order 2 (—), 4 (—), 6 (—); Uniform resolution (FFT) (—) and convergence $\mathcal{O}(h^2)$ (—).

6.4 Conclusions

In this chapter, we introduced a way to compute the unbounded convolution in a multi-resolution framework, which is a generalization of the work proposed by [102] to arbitrary convolution kernels (LGF in our application) and to FMM-based solvers. It uses first generation interpolating wavelets approach to interpolate the Poisson solution from one level to another and computes the final solution on each level as the sum of the solution on the underlying levels. We proved the expected second order convergence of the method on an analytical benchmark, using a lattice Green's function approach.

This work, so far, has been focused on the targeted application: an Immersed Interface body, hence using the LGF methodology. However, it could easily be extended to other convolution kernels: the regularized kernels or the Helmholtz Green's function. The extension of the proposed methodology to other applications is straightforward as it only involves FMM and FFT's, two well-known 3D algorithms. Finally, this approach could be applied to a broader range of elliptic problems, as the Helmholtz problems for example.

Acknowledgments

We would like to thank Pr. Siddhartha Verma and Pr. Petros Koumoutsakos for the helpful discussion and for providing an access to MRAG-I2D [114].

Conclusions

The Vortex Particle-Mesh (VPM) method has been developed for advection dominated flow simulations, such as wake analysis, relying on the particle discretization of the Navier-Stokes equations. The way to handle inner boundaries is crucial for such studies as the wake is the signature of the force exchange that happens at the fluid-structure interface through very fine flow phenomena. Yet, taking an immersed object into account in the VPM framework still remains an open challenge due to non-body-conforming mesh and the vorticity formulation, both inherent to the method. The first proposed approach, the coupling of the BEM and the VPM framework [30, 63], relies on a splitting in the diffusion operation, bounding the time convergence of this technique. Another widely used method is the penalization approach [32, 34, 55], which relies on an inaccurate representation of the body, leading to bounded spatial convergence behavior in the vicinity of the wall. Therefore there is a need for a flexible, accurate and efficient treatment of inner boundaries.

In this work we investigate such a treatment based on the Immersed Interface tool applied to the VPM framework. A first work in that direction has been proposed by Marichal [14], while three still open questions have drawn our attention: the computational cost of the Poisson solver, the splitting approach in the diffusion operation bounding the time convergence and the implementation of the proposed approach in 3D. In addition to these challenges we also address three other questions: the use of recycling techniques to decrease the computational cost associated to linear systems, the computation of wall quantities for fluid-structure interaction problems and the early development of a multiresolution approach to dedicate computational resources where it is most needed.

Contributions

In Chapter 3 we address the problem of the computational cost associated to the Poisson solver. We decompose the problem into two Poisson subproblems using the Sherman-Morrison-Woodbury (SMW) formula: the first one is associated with the bulk sources inside the whole computational domain and the second one is associated with the unknown Immersed Interface source term, i.e. in a smaller domain. The unknown Immersed Interface source term is first computed through a linear system to solve, typically using a GMRes iterative solver where every iteration comes at the cost of one Poisson equation to solve in the vicinity of the body surface. Finally, the contribution of this additional source term is evaluated in the whole computational domain. This approach is very convenient as it decomposes the Immersed Interface problem, initially embedded into the bulk problem, into two different source contributions, both solved using already existing fast Poisson solvers (FMM or FFT based approaches).

Since the Poisson problem has to be solved several times per time step and for thousands of time steps, we further also use a recycling technique in order to reduce the number of matrix-vector multiplications required and hence the associated computational cost. Introduced in the context of the iterative penalization technique in Chapter 2, the recycling approach uses previously computed solutions and residuals to deduce the best initial guess. This recycling approach heavily reduces the number of required iterations and hence it enables the resolution of such linear systems in realistic applications without overwhelming the computational cost. Let us note that it can easily be extended to any linear system to solved a large number of times, in a wide range of computational problems. Together with the ability to use highly efficient and well studied linear algebra tools, recycling is one of the main advantage to formulate any iterative process as a linear system to solve.

The next challenge we address is the diffusion splitting problem that bounds the time convergence. In Chapter 4, we propose to compute the diffusion term alike any other operation on the grid. Taking advantage of the Immersed Interface tool, we first compute the wall vorticity relying on the no-slip boundary condition and then we use it as a boundary condition when evolving the vorticity field. Hence, we avoid splitting the diffusion term computation and we reach the expected order for time convergence, unlike the BEM treatment proposed by Marichal [14]. The Immersed Interface tool enables the accurate and sharp imposition of the body boundary directly into the diffusion computation, hence dispensing with the BEM approach to impose the no-slip boundary condition. We notice that our approach also matches well the idea behind hybrid Eulerian-Lagrangian methods: the use of the structured grid to improve the

computational efficiency when computing the velocity, the diffusion and the stretching term.

Regarding the particles point of view, the interpolation operators rely on an extension of the field inside the body to avoid the use of a decentered interpolation scheme. This extrapolation is performed using a PDE-based approach, the Hamilton-Jacobi extension, well known in the field of levelset methods. Still in Chapter 4 we extend this technique to the Immersed Interface framework and we use it to build ghost particles and ghost points. This extension does not have any physical meaning, yet it enables the use of centered and well-studied interpolation kernels (M'_4 or M^*_6 [11, 12]). Moreover, unlike the fast marching method [129], this type of extensions is easily implemented, especially on highly distributed architectures.

In Chapter 5, we put all these tools together and we present the 3D Immersed Interface Vortex Particle-Mesh (IIVPM) method. This method is validated on the benchmark of the impulsively started flow past a sphere for short and long times and is shown to be as fast as the widely used iterative penalization approach, yet much sharper and more accurate. We also detail the way to evaluate wall quantities and surface integrals using local information only and show how to retrieve the viscous and the pressure drag coefficients. These informations are especially required for fluid-structure interaction problems like deforming obstacles and moving objects. In this chapter, we also highlight the drawbacks of the methodology we propose, as the need for a multiresolution approach in order to provide higher accuracy in the vicinity of the body surface. Finally, let us note that, despite the heavy implementation overhead required by the Immersed Interface implementation, we consider that this method is particularly well suited for future developments and especially for moving and deforming objects. Nevertheless, handling the zero divergence of the vorticity field in the vicinity of the interface remains one of the open challenges. Indeed, the Immersed Interface treatment appears to create divergence in the computed vorticity field and the influence of the Hamilton-Jacobi on that is still not clear.

Finally, based on the observation that a multiresolution framework would be more appropriate to the Immersed Interface approach, a preliminary work on a multiresolution Poisson solver is presented in Chapter 6. We use an interpolating wavelet-based structured grid and we decompose the problem into subproblems using a forward and a backward pass. Finally, we sum the contribution of every level and compute the final solution in the whole computational domain. This exploratory work, exploits the wavelet approach to gain from a formalised multiresolution analysis and propose an associated Poisson solver. Nevertheless this work still needs improvement on the implementation efficiency and the characterisation of the error induced by this approach.

Perspectives

The Immersed Interface Vortex Particle-Mesh method still suffers from some open questions and drawbacks.

First, as already mentioned in this work, the Immersed Interface is a methodology well-suited for sparse Poisson solvers. In this work, we rely on a FFT-based Poisson solver inside a subdomain using the lattice Green function (LGF) convolution kernel. This FFT-based method is faster than a FMM technique, yet in some cases the latter would be better as it exploits the sparsity of the source term and of the target locations. For such an approach, the standard FMM has to be replaced by a kernel independent FMM approach in order to properly treat the LGF kernel. The KiFMM method introduces additional errors though, especially through the use of the singular value decomposition in order to compress the kernel for far field interactions. Both the FFT-based and the KiFMM-based approaches come at the price of the evaluation of the LGF, which is computed as a highly oscillating integral and stored in memory to decrease the computational cost associated with the evaluation of the integral. Another option could be to use a multi-grid approach, as some progress have been reported by Hosseinverdi and Fasel [69] in this direction.

Second, for the moment, the IIVPM method has been developed in a uniform resolution framework. However, as noticed in Chapter 4 and Chapter 5, the grid resolution needs to be important near the boundary and especially to evaluate the pressure and the viscous drag coefficient. This resolution is not required in the bulk, where the support of the vorticity is broader and the gradients are smoother. This suggests the use of a multiresolution framework in order to dedicate computational effort where it is most needed, in the vicinity of the interface where the Immersed Interface tool modifies the bulk operations. However, this approach comes with two major challenges. The first one is the Poisson solver, since a FFT-based approach is not valid anymore as such, see Chapter 6. Using a FMM-based approach or a multi-grid approach would constitutes a wiser choice, as already discussed. The second is the load balancing, as each level has to be distributed among every processor in order to ensure an optimal load-balancing on very large scale architectures. Nevertheless, the multiresolution approach can also be seen as a potential solution to the load balancing issue caused by the local stencil modification required by the Immersed Interface approach. Such considerations are crucial in the development of an Immersed Interface solver capable of tackling ambitious challenges on very large scale architectures.

Third, the way to extend the field inside the body, using the Hamilton-Jacobi extension, produces spurious divergence of the vorticity field near the body surface. We also

note that, when the curvature is high, the spurious divergence is more important. We think that a modification of the Hamilton-Jacobi extension by adding some diffusion may help to reduce its generation.

Additionally to the three mentioned improvements, we envision two directions for future research using the Immersed Interface Method.

The IIVPM approach has been developed for fixed bodies. In order to tackle more challenging problems, we need to extend it to moving and deforming bodies. We do not foresee significant differences between both frameworks, as soon as the level-set is available for the Immersed Interface stencil computation. The actual challenge is located in the levelset advection and computation rather than in the extension of the IIVPM method.

Finally, in order to reach significant Reynolds numbers, we will need to use a Large Eddy Simulation (LES) approach to model the influence of the smallest structures, not captured by the grid, on the resolved larger structures. This kind of approach will be crucial and we will have to couple the Immersed Interface method to a wall-modelling approach in order to reduce the number of unknowns required to capture sufficiently well the turbulent boundary layer effects without resolving it. These wall-model developments will thus constitute a key step before tackling high Reynolds number simulation. The Immersed Interface should be particularly well-suited for such an approach, as it directly uses the velocity at the wall, which could be obtained through these wall models.

Bibliography

- [1] O. Thiry, Investigation of wall shear stress models for large eddy simulation, Ph.D. thesis, UCLouvain, 2017.
- [2] G.-H. Cottet, P. Koumoutsakos, Vortex Methods, Theory and Practice, Cambridge University Press, 2000.
- [3] G. S. Winckelmans, Encyclopedia of Computational Mechanics, chap. Vortex Methods, John Wiley & Sons, Ltd, ISBN 9780470091357, 129–153, 2004.
- [4] J. Monaghan, Smoothed particle hydrodynamics, Annu. Rev. Astron. Astrophys. 30 (1992) 534–574.
- [5] J. T. Beale, A. Majda, Vortex Methods I: convergence in 3 dimensions, Mathematics of Computation 39 (159) (1982) 1–27.
- [6] J. T. Beale, On the accuracy of vortex methods at large times, in: B. E. et al. (Ed.), Proc. Workshop on Comput. Fluid Dyn. and React. Gas Flows, IMA, Univ. of Minnesota, 1986, Springer-Verlag, New York, 19, 1988.
- [7] J. Monaghan, SPH without a Tensile Instability, Journal of Computational Physics 159 (2000) 290–311.
- [8] G.-H. Cottet, Artificial viscosity models for vortex and particle methods, J. Comput. Phys. 127 (2) (1996) 299–308.
- [9] P. Koumoutsakos, Inviscid axisymmetrization of an elliptical vortex, Journal of Computational Physics 138 (2) (1997) 821–857.

- [10] P. Ploumhans, G. S. Winckelmans, Vortex Methods for High-Resolution Simulations of Viscous Flow Past Bluff Bodies of General Geometry, *Journal of Computational Physics* 165 (2) (2000) 354–406.
- [11] J. Monaghan, Extrapolating B splines for interpolation, *Journal of Computational Physics* 60 (2) (1985) 253 – 262.
- [12] J. J. Monaghan, Smoothed particle hydrodynamics, *Reports on Progress in Physics* 68 (8) (2005) 1703–1759.
- [13] W. M. van Rees, A. Leonard, D. I. Pullin, P. Koumoutsakos, A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers, *Journal of Computational Physics* 230 (8) (2011) 2794–2805.
- [14] Y. Marichal, An immersed interface vortex particle-mesh method, Ph.D. thesis, Université catholique de Louvain, 2014.
- [15] R. Hockney, J. Eastwood, *Computer Simulation using Particles*, Taylor & Francis, Inc. Bristol, PA, USA, 1988.
- [16] P. Chatelain, P. Koumoutsakos, A Fourier-based elliptic solver for vortical flows with periodic and unbounded directions, *Journal of Computational Physics* 229 (7) (2010) 2425–2431.
- [17] M. Hejlesen, J. Rasmussen, P. Chatelain, J. Walther, A high order solver for the unbounded Poisson equation, *Journal of Computational Physics* 252 (2013) 458–467.
- [18] L. Greengard, *A rapid evaluation of potential fields in particle systems*, MIT Press, 1988.
- [19] J. K. Salmon, M. S. Warren, Fast Parallel Tree Codes for Gravitational and Fluid Dynamical N-Body Problems, *The International Journal of Supercomputer Applications and High Performance Computing* 8 (2) (1994) 129–142.
- [20] R. Beatson, L. Greengard, A short course on fast multipole methods, in: *Wavelets, Multilevel Methods and Elliptic PDEs*, Oxford University Press, 1–37, 1997.
- [21] A. Gillman, P. Martinsson, Fast and accurate numerical methods for solving elliptic difference equations defined on lattices, *Journal of Computational Physics* 229 (24) (2010) 9026 – 9041.

- [22] A. Gillman, P. Martinsson, A fast solver for Poisson problems on infinite regular lattices, *Journal of Computational and Applied Mathematics* 258 (2014) 42 – 56.
- [23] F. Vico, L. Greengard, M. Ferrando, Fast convolution with free-space Green's functions, *Journal of Computational Physics* 323 (2016) 191 – 203.
- [24] R. Cocle, G. Winckelmans, G. Daeninck, Combining the vortex-in-cell and parallel fast multipole methods for efficient domain decomposition simulations, *Journal of Computational Physics* 227 (21) (2008) 9091–9120.
- [25] S. Backaert, P. Chatelain, L. Bricteux, G. Winckelmans, P. Koumoutsakos, Vortex particle-mesh methods with immersed lifting lines applied to the Large Eddy Simulation of wind turbine wakes, *Journal of Computational Physics* .
- [26] G.-H. Cottet, P. Poncet, Advances in direct numerical simulations of 3D wall-bounded flows by Vortex-in-Cell methods, *J. Comput. Phys.* 193 (1) (2004) 136–158.
- [27] G. Daeninck, Developments in hybrid approaches. Vortex method with known separation location. Vortex method with near-wall Eulerian solver. RANS-LES coupling, Ph.D. thesis, Université catholique de Louvain, 2006.
- [28] G. Papadakis, Development of a hybrid compressible vortex particle method and application to external problems including helicopter flows, Ph.D. thesis, National Technical University of Athens, 2014.
- [29] P. Parmentier, Coupling a vortex particle-mesh method for compressible flows with a discontinuous Galerkin solver, Ph.D. thesis, UCLouvain, 2018.
- [30] P. Ploumhans, G. Winckelmans, J. Salmon, A. Leonard, M. Warren, Vortex Methods for Direct Numerical Simulation of Three-Dimensional Bluff Body Flows: Application to the Sphere at $Re=300$, 500, and 1000, *Journal of Computational Physics* 178 (2) (2002) 427 – 463.
- [31] J. Rasmussen, Particle Methods in Bluff Body Aerodynamics, Ph.D. thesis, Technical University of Denmark, 2011.
- [32] C. Mimeau, F. Gallizio, G.-H. Cottet, I. Mortazavi, Vortex penalization method for bluff body flows, *International Journal for Numerical Methods in Fluids* 79 (2) (2015) 55–83, fld.4038.

- [33] C. Mimeau, G.-H. Cottet, I. Mortazavi, Direct numerical simulations of three-dimensional flows past obstacles with a vortex penalization method, *Computers & Fluids* 136 (2016) 331 – 347.
- [34] M. M. Hejlesen, P. Koumoutsakos, A. Leonard, J. H. Walther, Iterative Brinkman penalization for remeshed vortex methods, *Journal of Computational Physics* 280 (2015) 547–562.
- [35] H. J. Spietz, M. M. Hejlesen, J. H. Walther, Iterative Brinkman penalization for simulation of impulsively started flow past a sphere and a circular disc, *Journal of Computational Physics* 336 (2017) 261 – 274.
- [36] P. Chatelain, A. Curioni, M. Bergdorf, D. Rossinelli, W. Andreoni, P. Koumoutsakos, Billion vortex particle Direct Numerical Simulations of aircraft wakes, *Computer Methods in Applied Mechanics and Engineering* 197 (13) (2008) 1296–1304.
- [37] G. Winckelmans, R. Cocle, L. Dufresne, R. Capart, Vortex methods and their application to trailing wake vortex simulations, *C. R. Phys.* 6 (4-5) (2005) 467–486.
- [38] P. Chatelain, M. Duponcheel, Y. Marichal, G. Winckelmans, Vortex Particle-Mesh methods for large scale LES of aircraft wakes, in: *APS-DFD*, Boston Nov 21–23, American Physical Society - Division of Fluid Dynamics, American Physical Society, 2015.
- [39] D.-G. Caprace, S. Buffin, M. Duponcheel, P. Chatelain, G. Winckelmans, Large Eddy Simulation of Advancing Rotor for Near to Far Wake Assessment, in: *43rd European Rotorcraft Forum*, Milan, Italy, 570, 2017.
- [40] E. Branlard, *Wind Turbine Aerodynamics and Vorticity-Based Methods*, Springer, 2017.
- [41] P. Chatelain, M. Duponcheel, D.-G. Caprace, Y. Marichal, G. Winckelmans, Vortex particle-mesh simulations of vertical axis wind turbine flows: from the airfoil performance to the very far wake, *Wind Energy Science* 2 (1) (2017) 317.
- [42] M. Gazzola, P. Chatelain, W. M. van Rees, P. Koumoutsakos, Simulations of single and multiple swimmers with non-divergence free deforming geometries, *Journal of Computational Physics* 230 (19) (2011) 7093–7114.

- [43] W. M. van Rees, M. Gazzola, P. Koumoutsakos, Optimal morphokinematics for undulatory swimmers at intermediate Reynolds numbers, *Journal of Fluid Mechanics* 775 (2015) 178–188.
- [44] S. Verma, G. Novati, P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning, *Proceedings of the National Academy of Sciences* 115 (23) (2018) 5849–5854.
- [45] R. Chatelin, P. Poncet, Hybrid grid-particle methods and Penalization: A Sherman-Morrison-Woodbury approach to compute 3D viscous flows using FFT, *J. Comput. Phys.* 269 (0) (2014) 314 – 328.
- [46] A. Thom, T. G. Ingram, The flow past circular cylinders at low speeds, *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 141 (845) (1933) 651–669.
- [47] W. R. Briley, A numerical study of laminar separation bubbles using the Navier-Stokes equations 47 (4) (1971) 713–736.
- [48] L. C. Woods, A Note on the Numerical Solution of Fourth Order Differential Equations 5 (4) (1954) 176–184.
- [49] E. Weinan, J.-G. Liu, Vorticity boundary condition and related issues for finite difference schemes, *Journal of computational physics* 124 (2) (1996) 368–382.
- [50] M. Napolitano, G. Pascazio, L. Quartapelle, A review of vorticity conditions in the numerical solution of the ζ - Ψ equations, *Computers & Fluids* 28 (2) (1999) 139–185.
- [51] R. Mittal, G. Iaccarino, Immersed boundary methods, *Annual Review of Fluid Mechanics* 37 (2005) 239–261.
- [52] C. S. Peskin, Numerical analysis of blood flow in the heart, *Journal of Computational Physics* 25 (3) (1977) 220–252.
- [53] K. Taira, T. Colonius, The immersed boundary method: A projection approach, *Journal of Computational Physics* 225 (2) (2007) 2118–2137.
- [54] S. Liska, T. Colonius, A fast immersed boundary method for external incompressible viscous flows using lattice Green’s functions, *Journal of Computational Physics* 331 (2017) 257 – 279.

- [55] P. Angot, C. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, *Numerische Mathematik* 81 (1999) 497–520.
- [56] N. K.-R. Kevlahan, J.-M. Ghidaglia, Computation of turbulent flow past an array of cylinders using a spectral method with Brinkman penalization, *European Journal of Mechanics - B/Fluids* 20 (3) (2001) 333–350.
- [57] M. Coquerelle, G. H. Cottet, A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies, *Journal of Computational Physics* 227 (21) (2008) 9121–9137.
- [58] M. Gazzola, P. Chatelain, P. Koumoutsakos, Vortex methods for fluid-structure interaction problems with deforming geometries and their application to swimming, in: *Division of Fluid Dynamics*, American Physical Society, 2010.
- [59] W. van Rees, 3D simulations of vortex dynamics and biolocomotion, Ph.D. thesis, ETH Zurich, 2014.
- [60] M. Hejlesen, J. Rasmussen, A. Larsen, J. H. Walther, On estimating the aerodynamic admittance of bridge sections by a mesh-free vortex method, *Journal of Wind Engineering and Industrial Aerodynamics* 146 (2015) 117 – 127.
- [61] T. Gillis, G. Winckelmans, P. Chatelain, An efficient iterative penalization method using recycled Krylov subspaces and its application to impulsively started flows, *Journal of Computational Physics* 347 (2017) 490 – 505.
- [62] F. Gibou, D. Hyde, R. Fedkiw, Sharp interface approaches and deep learning techniques for multiphase flows, *Journal of Computational Physics* 380 (2019) 442–463.
- [63] P. Koumoutsakos, A. Leonard, High-resolution simulations of the flow around an impulsively started cylinder using vortex methods, *J. Fluid. Mech.* 296 (1) (1995) 1–38.
- [64] M. J. Lighthill, Introduction of boundary layer theory, in: R. L. (Ed.), *Laminar Boundary Layers*, Oxford University Press, New York, 1963.
- [65] R. J. Leveque, Z. L. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM Journal on Numerical Analysis* 31 (4) (1994) 1019–1044.

- [66] A. Wiegmann, K. Bube, The Explicit-Jump Immersed Interface Method: Finite Difference Methods for PDEs with Piecewise Smooth Solutions, *SIAM Journal on Numerical Analysis* 37 (3) (2000) 827–862.
- [67] M. N. Linnick, H. F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, *Journal of Computational Physics* 204 (2005) 157–192.
- [68] A. Mayo, A. Greenbaum, Fourth order accurate evaluation of integrals in potential theory on exterior 3D regions, *Journal of Computational Physics* 220 (2) (2007) 900 – 914.
- [69] S. Hosseinverdi, H. F. Fasel, An efficient, high-order method for solving Poisson equation for immersed boundaries: Combination of compact difference and multiscale multigrid methods, *Journal of Computational Physics* 374 (2018) 912–940.
- [70] P. Martinsson, Fast multiscale methods for lattice equations, Ph.D. thesis, The University of Texas, 2002.
- [71] Z. Li, I. Kazufumi, The immersed interface method: Numerical solutions of PDEs involving interfaces and irregular domains, SIAM, 2006.
- [72] S. Liska, T. Colonius, A parallel fast multipole method for elliptic difference equations, *Journal of Computational Physics* 278 (2014) 76 – 91.
- [73] R. A. James, Solution of Poisson’s equation for isolated source distributions, *Journal of Computational Physics* 25 (2) (1977) 71–93.
- [74] K. Lackner, Computation of ideal MHD equilibria, *Computer Physics Communications* 12 (1) (1976) 33 – 44.
- [75] G. Miller, An iterative boundary potential method for the infinite domain Poisson problem with interior Dirichlet boundaries, *Journal of Computational Physics* 227 (16) (2008) 7917 – 7928.
- [76] Y. Marichal, P. Chatelain, G. Winckelmans, An immersed interface solver for the 2-D unbounded Poisson equation and its application to potential flow, *Computers & Fluids* 96 (2014) 76–86.
- [77] G. S. Winckelmans, Topics in vortex methods for the computation of three- and two-dimensional incompressible unsteady flows, Ph.D. thesis, California Institute of Technology, 1989.

- [78] R. Cortez, The Method of Regularized Stokeslets, *SIAM Journal on Scientific Computing* 23 (4) (2001) 1204–1225.
- [79] P.-G. Martinsson, G. J. Rodin, Asymptotic expansion of lattice Green’s function, *Proceedings: Mathematical, Physical and Engineering Sciences* 458 (2027) (2002) 2609–2622.
- [80] S. Liska, T. Colonius, A fast lattice Green’s function method for solving viscous incompressible flows on unbounded domains, *Journal of Computational Physics* 316 (2016) 360 – 384.
- [81] Y. Marichal, P. Chatelain, G. Winckelmans, Immersed interface interpolation schemes for particle–mesh methods, *Journal of Computational Physics* 326 (2016) 947 – 972.
- [82] T. Gillis, G. Winckelmans, P. Chatelain, Fast immersed interface Poisson solver for 3D unbounded problems around arbitrary geometries, *Journal of Computational Physics* 354 (2018) 403 – 416.
- [83] G. H. Cottet, B. Michaux, S. Ossia, G. Vanderlinden, A comparison of spectral and vortex methods in three-dimensional incompressible flows., *Journal of Computational Physics* 175 (2) (2002) 702–712.
- [84] M. Bergdorf, P. Koumoutsakos, A. Leonard, Direct numerical simulations of vortex rings at $Re\text{-}\Gamma=7500$, *Journal of Fluid Mechanics* 581 (2007) 495–505.
- [85] J. T. Rasmussen, G.-H. Cottet, J. H. Walther, A multiresolution remeshed Vortex-In-Cell algorithm using patches, *Journal of Computational Physics* 230 (2011) 6742–6755.
- [86] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2003.
- [87] Y. Saad, M. H. Schultz, GMRes: A Generalized Minimal Residual Algorithm For Solving NonSymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.* 7 (3).
- [88] P. Sonneveld, CGS, a Fast Lanczos-type Solver for Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.* 10 (1) (1989) 36–52.
- [89] H. A. van der Vorst, BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.* 13 (2) (1992) 631–644.

- [90] E. de Sturler, Nested Krylov methods based on GCR, *Journal of Computational and Applied Mathematics* 67 (1996) 15–41.
- [91] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, S. maiti, Recycling Krylov subspaces for sequences of linear systems, *SIAM Journal on Scientific Computing* 28 (5) (2006) 1651–1674.
- [92] R. B. Morgan, GMRes with deflated restarting, *SIAM Journal on Scientific Computing* 24 (1) (2002) 20–37.
- [93] M. H. Carpenter, A General Algorithm for Reusing Krylov Subspace Information., Technical Memorandum NASA/TM–2010–216190, Langley Research Center, 2010.
- [94] K. Ahuja, P. Benner, E. de Sturler, L. Feng, Recycling BiCGStab with an application to parametric model order reduction, *SIAM Journal on Scientific Computing* 37 (5) (2015) S429–S446.
- [95] E. de Sturler, Truncation Strategies for optimal Krylov subspace methods, *SIAM Journal on Numerical Analysis* 36 (3) (1999) 864–889.
- [96] A. Amritkar, E. Sturler, K. Swirydowicz, D. Tafti, K. Ahuja, Recycling Krylov subspaces for CFD applications and a new hybrid recycling solver, *Journal of Computational Physics* 303 (2015) 222–237.
- [97] D. Rossinelli, M. Bergdorf, G.-H. Cottet, P. Koumoutsakos, GPU Accelerated Simulations of Bluff Body Flows Using Vortex Particle Methods, *J. Comput. Phys.* 229 (9) (2010) 3316–3333.
- [98] C. Lanczos, Solution of systems of linear equations by minimized iterations, *Journal of Research of the National Bureau of Standards* 49 (1952) 33–53.
- [99] F. Noca, D. Shiels, D. Jeon, A comparison of methods for evaluating time-dependent fluid dynamic forces on bodies, using only velocity fields and their derivatives, *Journal of Fluids and Structures* 13 (5) (1999) 551–578.
- [100] M. Bar-Lev, H. T. Yang, Initial flow field over an impulsively started circular cylinder, *Journal of Fluid Mechanics* 72 (1975) 625–647.
- [101] P. McCorquodale, P. Colella, G. T. Balls, S. B. Baden, A Local Corrections Algorithm for Solving Poisson's Equation in Three Dimensions, *Communications in Applied Mathematics and Computational Science* 2 (1) (2007) 57–81.

- [102] M. M. Hejlesen, J. H. Walther, A multiresolution method for solving the Poisson equation using high order regularization, *Journal of Computational Physics* 326 (2016) 188 – 196.
- [103] T. Colonius, K. Taira, A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions, *Computer Methods in Applied Mechanics and Engineering* 197 (25-28) (2008) 2131–2146.
- [104] J. Sherman, W. J. Morrison, Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix, *Ann. Math. Statist.* 21 (1) (1950) 124–127.
- [105] W. W. Hager, Updating the Inverse of a Matrix, *SIAM Review* 31 (2) (1989) 221–239.
- [106] G. H. Golub, C. F. Van Loan, *Matrix Computation*, The Johns Hopkins University Press, 1996.
- [107] Y. Marichal, P. Chatelain, G. Winckelmans, An immersed interface vortex particle-mesh solver, in: *Bulletin of the American Physical Society*, vol. 59, American Physical Society, 2014.
- [108] I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, P. Koumoutsakos, PPM A highly efficient parallel particle mesh library for the simulation of continuum systems, *J. Comput. Phys.* 215 (2006) 566–588.
- [109] P. Poncet, Analysis of an immersed boundary method for three-dimensional flows in vorticity formulation, *J. Comput. Phys.* 228 (2009) 7268–7288.
- [110] P. Chatelain, A. Leonard, Isotropic compact interpolation schemes for particle methods, *Journal of Computational Physics* 227 (6) (2008) 3244 – 3259.
- [111] J. H. Williamson, Low-storage Runge-Kutta schemes, *Journal of Computational Physics* 35 (1) (1980) 48–56.
- [112] T. D. Aslam, A partial differential equation approach to multidimensional extrapolation, *Journal of Computational Physics* 193 (1) (2004) 349 – 355.
- [113] F. Noca, On the evaluation of instantaneous fluid-dynamic forces on a bluff body, Report FM96-5, GALCIT, 1996.

- [114] D. Rossinelli, B. Hejzialhosseini, W. van Rees, M. Gazzola, M. Bergdorf, P. Koumoutsakos, MRAG-I2D: Multi-resolution adapted grids for remeshed vortex methods on multicore architectures, *Journal of Computational Physics* 288 (2015) 1–18.
- [115] P. F. Fischer, Projection techniques for iterative solution of $Ax = b$ with successive right-hand sides, *Computer Methods in Applied Mechanics and Engineering* 163 (1) (1998) 193–204.
- [116] J.-Z. Wu, H.-Y. Ma, M.-D. Zhou, *Vorticity and Vortex Dynamics*, Springer, 2006.
- [117] A.-K. Tornberg, B. Engquist, R. Tsai, Discretization of Dirac delta functions in level set methods, *Journal of Computational Physics* 207 (2005) 28–51.
- [118] P. Smereka, The numerical approximation of a delta function with application to level set methods, *Journal of Computational Physics* 211 (1) (2006) 77–90.
- [119] J. Jeong, F. Hussain, On the identification of a vortex 285 (1995) 69–94.
- [120] Y. Zhang, K. Liu, H. Xian, X. Du, A review of methods for vortex identification in hydroturbines, *Renewable and Sustainable Energy Reviews* 81 (2018) 1269–1285.
- [121] M. S. Warren, J. K. Salmon, A parallel hashed Oct-Tree N-body algorithm, in: *Supercomputing*, 12–21, 1993.
- [122] L. Greengard, V. Rokhlin, A Fast Algorithm for Particle Simulations, *Journal of Computational Physics* 73 (1987) 325–348.
- [123] G. Winckelmans, J. Salmon, M. Warren, A. Leonard, B. Jodoin, Application of fast parallel and sequential tree codes to computing three-dimensional flows with the vortex element and boundary element methods, in: *ESAIM: Proceedings*, vol. 1, EDP Sciences, 225–240, 1996.
- [124] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, *Journal of Computational Physics* 196 (2) (2004) 591 – 626.
- [125] C. R. Anderson, A method of local corrections for computing the velocity field due to a distribution of vortex blobs, *Journal of Computational Physics* 62 (1) (1986) 111–123.

- [126] A. Gholami, D. Malhotra, H. Sundar, G. Biros, FFT, FMM, or Multigrid? A comparative Study of State-Of-the-Art Poisson Solvers for Uniform and Nonuniform Grids in the Unit Cube, *SIAM Journal on Scientific Computing* 38 (3) (2016) C280–C306.
- [127] M. Hejlesen, A high order regularisation method for solving the Poisson equation and selected applications using vortex methods, Ph.D. thesis, DTU Mechanical Engineering, 2016.
- [128] G. Deslauriers, S. Dubuc, Symmetric iterative interpolation processes, *Constructive Approximation* 5 (1) (1989) 49–68.
- [129] J. Sethian, Fast Marching Methods, *SIAM Review* 41 (2) (1999) 199–235.
- [130] S. H. Lamb, *Hydrodynamics*, Dover, New York, 1993.
- [131] DLMF, NIST Digital Library of Mathematical Functions, <http://dlmf.nist.gov/>, Release 1.0.14 of 2016-12-21, f.~W.~J. Olver, A.~B. Olde Daalhuis, D.~W. Lozier, B.~I. Schneider, R.~F. Boisvert, C.~W. Clark, B.~R. Miller and B.~V. Saunders, eds., 2016.
- [132] T. Funada, Funada's file of potential functions for ellipsoids, 2006.
- [133] B. C. Carlson, E. M. Notis, Algorithm 577: Algorithms for Incomplete Elliptic Integrals [S21], *ACM Trans. Math. Softw.* 7 (3) (1981) 398–403.
- [134] S. Haykin, B. V. Veen, *Signals and Systems*, JOHN WILEY & SONS LTD, second edition edn., 2005.
- [135] G. F. Koster, J. C. Slater, Simplified Impurity Calculation, *Physical Review* 96 (5) (1954) 1208–1223.
- [136] T. Morita, Use of a recurrence formula in computing the lattice Green function 8 (4) (1975) 478–489.
- [137] T. Horiguchi, T. Morita, Note on the lattice Green's function for the simple cubic lattice 8 (11) (1975) L232–L235.
- [138] J. Cserti, Application of the lattice Green's function for calculating the resistance of an infinite network of resistors, *American Journal of Physics* 68 (10) (2000) 896–906.

- [139] S. G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (7) (1989) 674–693.
- [140] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, 1992.
- [141] W. Sweldens, The lifting Scheme: a custom-design construction of biorthogonal wavelets, *Applied and Computational Harmonic Analysis* 3 (1996) 186–200.
- [142] A. Cohen, I. Daubechies, J.-C. Feauveau, Biorthogonal bases of compactly supported wavelets, *Communications on Pure and Applied Mathematics* 45 (5) (1992) 485–560.
- [143] D. Donoho, T. Yu, Deslauriers Dubuc Ten Years After, Tech. Rep., Stanford University, 1996.
- [144] D. Donoho, *Interpolation Wavelet Transforms*, Tech. Rep., Stanford University, 1992.
- [145] W. Sweldens, The Lifting Scheme: A Construction of Second Generation Wavelets, *SIAM Journal on Mathematical Analysis* 29 (2) (1998) 511–546.

Analytical formulations

A.1 Signed Distance Function of a sphere

The levelset function of a sphere is given by

$$\phi(x, y, z) = \sqrt{x^2 + y^2 + z^2} - R^2. \quad (\text{A.1})$$

The normal is given by $\nabla\phi/|\nabla\phi|$, and in this case,

$$\nabla\phi = \mathbf{n} = \frac{1}{\sqrt{x^2 + y^2 + z^2}} \begin{bmatrix} x & y & z \end{bmatrix}, \quad (\text{A.2})$$

since $|\nabla\phi| = 1$. The second derivative of the levelset is given by

$$\frac{\partial \mathbf{n}_i}{\partial x_j} = \frac{1}{(x^2 + y^2 + z^2)^{3/2}} \begin{bmatrix} (y^2 + z^2) & (-xy) & (-xz) \\ (-xy) & (x^2 + z^2) & (-yz) \\ (-xz) & (-yz) & (x^2 + y^2) \end{bmatrix}. \quad (\text{A.3})$$

A.2 Signed distance function of an ellipsoid

The levelset function of an ellipsoid is given by

$$\phi(x, y, z) = \sqrt{\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2} - R^2. \quad (\text{A.4})$$

The normal, i.e. the derivative of the levelset, of an ellipsoid, is given by

$$\nabla\phi = \frac{1}{\sqrt{\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2}} \begin{bmatrix} \frac{x}{a^2} & \frac{y}{b^2} & \frac{z}{c^2} \end{bmatrix}^T. \quad (\text{A.5})$$

The norm of $\nabla\phi$ is not constant, so the normal is defined as $\mathbf{n} = \nabla\phi/|\nabla\phi|$ and is given by

$$\mathbf{n} = \frac{1}{\left(\frac{x^2}{a^4} + \frac{y^2}{b^4} + \frac{z^2}{c^4}\right)^{1/2}} \begin{bmatrix} \frac{x^2}{a^2} & \frac{y^2}{b^2} & \frac{z^2}{c^2} \end{bmatrix}^T. \quad (\text{A.6})$$

The normal derivative is then given by

$$\frac{\partial \mathbf{n}_i}{\partial x_j} = \frac{1}{D} \begin{bmatrix} \left(\frac{y^2}{a^2 b^4} + \frac{z^2}{a^2 c^4}\right) & \left(-\frac{xy}{a^2 b^4}\right) & \left(-\frac{xz}{a^2 c^4}\right) \\ \left(-\frac{xy}{b^2 a^4}\right) & \left(\frac{x^2}{b^2 a^4} + \frac{z^2}{b^2 c^4}\right) & \left(-\frac{yz}{b^2 c^4}\right) \\ \left(-\frac{xz}{c^2 a^4}\right) & \left(-\frac{yz}{c^2 b^4}\right) & \left(\frac{x^2}{c^2 a^4} + \frac{y^2}{c^2 b^4}\right) \end{bmatrix}, \quad (\text{A.7})$$

with

$$D = \left(\frac{x^2}{a^4} + \frac{y^2}{b^4} + \frac{z^2}{c^4}\right)^{3/2}. \quad (\text{A.8})$$

A.3 Analytical solution of the potential flow past an ellipsoid

The analytical expression of the potential flow past an ellipsoid is given by [130, page 152]. We have that the solution of the potential flow aligned with z , around an ellipsoid,

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1, \quad (\text{A.9})$$

with $a < b < c$, is given by

$$\phi = \frac{abc}{2 - \gamma_0} U_\infty z \int_\lambda^\infty (a^2 + \lambda)^{-1/2} (b^2 + \lambda)^{-1/2} (c^2 + \lambda)^{-3/2} d\lambda, \quad (\text{A.10})$$

with

$$\gamma_0 = abc \int_0^\infty (a^2 + \lambda)^{-1/2} (b^2 + \lambda)^{-1/2} (c^2 + \lambda)^{-3/2} d\lambda = abc \frac{2}{3} R_D(a^2, b^2, c^2), \quad (\text{A.11})$$

and where λ is the higher absolute value root of the third order polynomial

$$\frac{x^2}{a^2 + \theta} + \frac{y^2}{b^2 + \theta} + \frac{z^2}{c^2 + \theta} = 1. \quad (\text{A.12})$$

In other words, for every point in space, one has to find λ by finding the roots of

$$\begin{aligned} & \theta^3 \\ & + \theta^2 (a^2 + b^2 + c^2 - x^2 - y^2 - z^2) \\ & + \theta (a^2 b^2 + b^2 c^2 + c^2 a^2 - x^2 (b^2 + c^2) - y^2 (a^2 + c^2) - z^2 (a^2 + b^2)) \\ & + (a^2 b^2 c^2 - x^2 b^2 c^2 - y^2 a^2 c^2 - z^2 a^2 b^2) = 0 \end{aligned} \quad (\text{A.13})$$

which can be solved using exact formula [131, §1.11(iii)]. The remaining integral, I ,

$$I = \int_{\lambda}^{\infty} (a^2 + \lambda)^{-1/2} (b^2 + \lambda)^{-1/2} (c^2 + \lambda)^{-3/2} d\lambda . \quad (\text{A.14})$$

can also be expressed as (see [132])

$$I = \frac{2}{(c^2 - a^2)^{3/2}} D(\zeta, k) , \quad (\text{A.15})$$

where D is the Legendre integral defined in [131, §19.2(ii)] and

$$\sin \zeta = \eta = \sqrt{\frac{c^2 - a^2}{c^2 + \lambda}} \quad k = \sqrt{\frac{c^2 - b^2}{c^2 - a^2}} . \quad (\text{A.16})$$

Using the definition of R_D , given in [131, §19.25(i)], we finally get

$$I = \frac{2/3}{(c^2 - a^2)^{3/2}} R_D(\epsilon - 1, \epsilon - k^2, \epsilon) , \quad (\text{A.17})$$

with $\epsilon = \frac{1}{\sin^2 \zeta} = \frac{1}{\eta^2}$. Finally, the potential is expressed as

$$\phi = \frac{abc U_{\infty} z}{3 - abc R_D(a^2, b^2, c^2)} \left[\frac{R_D(\epsilon - 1, \epsilon - k^2, \epsilon)}{(c^2 - a^2)^{3/2}} \right] , \quad (\text{A.18})$$

which is evaluated using [133].

The Fourier transform

B.1 Definitions

In this section, we first briefly recall the Fourier transform and some helpful properties. The Fourier transform is different depending on the boundary conditions of the real space and the Fourier space. The possible combinations are driven by two rules:

- If one of the domain is periodic, the other one will be discrete
- If one of the domain is continuous, the other one is unbounded.

There is therefore four different Fourier Transform, presented below. We use the denomination proposed in the book of Haykin and Veen [134].

Periodic and continuous domain - Fourier Series

A periodic function $f(x)$ with $x \in [0 ; L]$ has the following Fourier transforms:

$$\mathbf{F}[f(x)] = \hat{f}[k] = \frac{1}{L} \int_0^L f(x) \mathbf{e}^{-ik\omega_0 x} dx \quad (\text{B.1})$$

$$\tilde{\mathbf{F}}^{-1}[\hat{f}(p)] = f(x) = \sum_{k=-\infty}^{\infty} \hat{f}(p) \mathbf{e}^{ik\omega_0 x} \quad (\text{B.2})$$

where $\omega_0 = \frac{2\pi}{L}$ is called the fundamental frequency and $k \in \mathbb{Z}$.

Unbounded and continuous domain - Fourier Transform

An unbounded function $f(x)$ with $x \in]-\infty ; \infty[$ has the following Fourier transforms:

$$\mathbf{F}[f(x)] = \hat{f}(k) = \int_{-\infty}^{\infty} f(x) \mathbf{e}^{-ikx} dx \quad (\text{B.3})$$

$$\mathbf{F}^{-1}[\hat{f}(k)] = f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(k) \mathbf{e}^{ikx} dk, \quad (\text{B.4})$$

where the frequency $k \in]-\infty ; \infty[$.

Periodic and discrete domain - Discrete Space Fourier Series (DFT)

A periodic and discrete function $f(x)$ with $x_j = j \frac{L}{N}$ with $j \in [0 ; N - 1]$ is defined by

$$\mathbf{F}[f(x_j)] = \hat{f}[k] = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) \mathbf{e}^{-i k \omega_0 x_j} \quad (\text{B.5a})$$

$$\mathbf{F}^{-1}[\hat{f}_k] = f(x_j) = \sum_{k=-N/2}^{N/2} \hat{f}_k \mathbf{e}^{i k \omega_0 x_j} \quad (\text{B.5b})$$

where $\omega_0 = \frac{2\pi}{L}$ is the fundamental frequency.

Unbounded and discrete domain - Discrete Space Fourier Transform

The Fourier transform for an unbounded and discrete function, $f(x)$ with $x = ih$ where h is the grid spacing and $i \in \mathbb{Z}$, is defined by

$$\mathbf{F}[f(x_j)] = \hat{f}(k) = \sum_{j=-\infty}^{\infty} f(x_j) \mathbf{e}^{-ikx_j} \quad (\text{B.6a})$$

$$\mathbf{F}^{-1}[\hat{f}(k)] = f(x_j) = \frac{1}{2\pi} \int_{-\pi/h}^{\pi/h} \hat{f}(k) \mathbf{e}^{ikx_j} dk \quad (\text{B.6b})$$

with $k \in \left[-\frac{\pi}{h} ; \frac{\pi}{h}\right]$.

B.2 Properties

Derivative in the real domain

If the function is continuous, a space derivative is equivalent to multiplying by ik :

$$\frac{d}{dx} f = \frac{1}{2\pi} \frac{d}{dx} \left[\int_{-\infty}^{\infty} \hat{f}(k) e^{ikx} dk \right] \quad (\text{B.7})$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} (ik) \hat{f}(k) e^{ikx} dk \quad (\text{B.8})$$

When a function is discrete, there is no such equivalence. However, one can get an approximation of it using finite differences.

Fourier transform of real data

If the data are real, the negative Fourier numbers k , are conjugate to the positive ones:

$$\hat{f}(k) = \overline{\hat{f}(-k)} . \quad (\text{B.9})$$

Therefore, one can simplify the FT by using DCT/DST. Their periodicity conditions are given by even and odd periodicity. When taking the derivative of a field, an even periodicity becomes odd one and inversely.

Shift in the real domain

A shift in the real domain is equivalent to a factor multiplication in the Fourier domain. Considering the case of an unbounded continuous domain,

$$\mathbf{F} [f(x - x_0)] = e^{-ikx_0} \hat{f}(k) . \quad (\text{B.10})$$

Windowing

Windowing a signal in the real space is equivalent to a multiplication with its window function. In the Fourier space, it will be equivalent to a convolution with the associated Fourier transform. Considering the case of an unbounded continuous function and a sharp window, we then obtain:

$$\mathbf{F} [f(x) W(x)] = \hat{f}(k) \star \hat{W}(k) . \quad (\text{B.11})$$

For a continuous rectangular window, from $-L$ to L ,

$$\hat{W}(k) = \frac{2 \sin(kL)}{k} . \quad (\text{B.12})$$

The zeros of this functions are located in $k_i = i \frac{\pi}{L}$, while the maximum/minimum values are located in $k_j = (i + 1/2) \frac{\pi}{L}$. For a discrete rectangular window, from $-M$ to M :

$$\hat{W}(k) = \frac{\sin \left[\frac{kh}{2} (2M + 1) \right]}{\sin \left(\frac{kh}{2} \right)} . \quad (\text{B.13})$$

Let us note that this function has several zeros, given by $k_p = \frac{2\pi}{h} \frac{p}{(2M + 1)}$

Lattice Green's Function computation

C.1 Two dimensions Lattice Green's function

The 2D Green's function is defined as (see Chapter 1)

$$G_h(\mathbf{m}) = \frac{1}{(2\pi)^2} \iint_{-\pi}^{\pi} \frac{\exp(i \mathbf{m} \cdot \boldsymbol{\xi})}{4 \sin^2(\xi_x/2) + 4 \sin^2(\xi_y/2)} d\boldsymbol{\xi} . \quad (\text{C.1})$$

In practice, to avoid the singularity in $(0,0)$ we shift the Green's function and rather compute $G_h(\mathbf{m}) - G_h(0,0)$. The real part of the integral then becomes

$$G_h(\mathbf{m}) = \frac{1}{(2\pi)^2} \iint_{-\pi}^{\pi} \frac{\cos(m_x \xi_x + m_y \xi_y) - 1}{4 \sin^2(\xi_x/2) + 4 \sin^2(\xi_y/2)} d\boldsymbol{\xi} . \quad (\text{C.2})$$

Another formulation, equivalent to Eq. (C.1) can also been developed by first, taking the real part of this integral,

$$G_h(\mathbf{m}) = \frac{1}{(2\pi)^2} \iint_{-\pi}^{\pi} \frac{\cos(m_x \xi_x + m_y \xi_y)}{4 \sin^2(\xi_x/2) + 4 \sin^2(\xi_y/2)} d\boldsymbol{\xi} . \quad (\text{C.3})$$

Then, using the symmetry of the problem, we obtain the expression, already obtained by many in the literature, see Koster and Slater [135] for example,

$$G_h(\mathbf{m}) = \frac{1}{\pi^2} \iint_0^{\pi} \frac{\cos(m_x \xi_x) \cos(m_y \xi_y)}{4 - 2 \cos(\xi_x) - 2 \cos(\xi_y)} d\boldsymbol{\xi} . \quad (\text{C.4})$$

Following the development proposed by [135] and using the definition of the modified Bessel function [131],

$$I_n(z) = \frac{1}{\pi} \int_0^\pi \exp(z \cos \theta) \cos(n\theta) d\theta, \quad (C.5)$$

we obtain the following expressions,

$$\begin{aligned} G_h &= \frac{1}{\pi^2} \int_0^\pi \int_0^\pi \cos(m_x \xi_x) \cos(m_y \xi_y) \exp(2t \cos(\xi_x) + 2t \cos(\xi_y) - 4t) d\xi dt \\ &= \int_0^\pi \exp(-4t) I_{m_x}(2t) I_{m_y}(2t) dt \end{aligned} \quad (C.6)$$

For small $|\mathbf{m}|$, we evaluate the integral; for $|\mathbf{m}|$ sufficiently large (in practice, $|\mathbf{m}| \geq 8$ or $|\mathbf{m}| \geq 32$), the integral is replaced by the following expansion [79]

$$\begin{aligned} \Theta(\mathbf{m}) &= \frac{1}{2\pi} \left[\log(\mathbf{m}) + \gamma + \frac{\log(8)}{2} \right] - \frac{1}{24\pi |\mathbf{m}|^6} \left[m_x^4 + m_y^4 - 6m_x^2 m_y^2 \right] \\ &\quad - \frac{1}{480\pi |\mathbf{m}|^{12}} \left[43 m_x^8 - 772 m_x^6 m_y^2 + 1570 m_x^4 m_y^4 - 772 m_x^2 m_y^6 + 43 m_y^8 \right] \\ &\quad - \frac{1}{2016\pi |\mathbf{m}|^{18}} \left[609 m_x^{12} - 24234 m_x^{10} m_y^2 + 109935 m_x^8 m_y^4 \right. \\ &\quad \quad \left. - 160524 m_x^6 m_y^6 + 109935 m_x^4 m_y^8 - 24234 m_x^2 m_y^{10} + 609 m_y^{12} \right] \\ &\quad - \frac{1}{2880\pi |\mathbf{m}|^{24}} \left[63139 m_x^{16} - 4467336 m_x^{14} m_y^2 + 38334996 m_x^{12} m_y^4 \right. \\ &\quad \quad - 98512568 m_x^{10} m_y^6 + 122747922 m_x^8 m_y^8 - 98512568 m_x^6 m_y^{10} \\ &\quad \quad \left. + 38334996 m_x^4 m_y^{12} - 4467336 m_x^2 m_y^{14} + 63139 m_y^{16} \right] \\ &\quad + \mathcal{O}(1/|\mathbf{m}|^{10}). \end{aligned} \quad (C.7)$$

C.2 Three dimensions Lattice Green's function

The 3D Green's function is defined as (see Chapter 1)

$$G_h(\mathbf{x}_m) \triangleq \frac{1}{8\pi^3 h} \iiint_{-\pi}^{\pi} \frac{\exp(i \mathbf{m} \cdot \boldsymbol{\xi})}{4 \sin^2(\xi_x/2) + 4 \sin^2(\xi_y/2) + 4 \sin^2(\xi_z/2)} d\boldsymbol{\xi} \quad (C.8a)$$

$$= \frac{1}{\pi^3 h} \iiint_0^\pi \frac{\cos(m_x \xi_x) \cos(m_y \xi_y) \cos(m_z \xi_z)}{6 - 2 \cos(\xi_x) - 2 \cos(\xi_y) - 2 \cos(\xi_z)} d\boldsymbol{\xi} \quad (C.8b)$$

where the second expression is obtained using the same development as in 2D.

In order to evaluate this integral, two main approaches are possible. The first one relies on recurrences, as proposed by Morita [136] and Horiguchi and Morita [137]. The second method is the generalization of the 2D approach. It relies on the evaluation of the equivalent unbounded integral, as proposed by Koster and Slater [135], Cserti [138] and used by Liska and Colonius [72].

$$\Theta(\mathbf{m}) = -\frac{1}{2} \int_0^\infty e^{-3t} I_{m_1}(t) I_{m_2}(t) I_{m_3}(t) dt \quad (\text{C.9a})$$

$$= -\int_0^\infty e^{-6t} I_{m_1}(2t) I_{m_2}(2t) I_{m_3}(2t) dt \quad (\text{C.9b})$$

where $I_n(\mathbf{x})$ is the modified Bessel function of the first kind of order n ¹. For larger $|\mathbf{m}|$, we use an expansion to avoid heavy computations. The way to derive the expansion for large $\mathbf{m}$'s is given by Martinsson and Rodin [79]. We here present the expansion as used in our work:

¹This integral is solved using the `quadgk` Matlab function between 0 and ∞ with 12 significant digits and the scaled Bessel function from Matlab `besseli(v, z, 1)`.

$$\begin{aligned}
\Theta(\mathbf{m}) = & \frac{1}{4\pi|\mathbf{m}|} \\
& + \frac{1}{16\pi|\mathbf{m}|^7} [m_x^4 + m_y^4 + m_z^4 - 3m_x^2m_y^2 - 3m_x^2m_z^2 - 3m_y^2m_z^2] \\
& + \frac{1}{128\pi|\mathbf{m}|^{13}} [23(m_x^8 + m_y^8 + m_z^8) \\
& - 244(m_x^6(m_y^2 + m_z^2) + m_y^6(m_x^2 + m_z^2) + m_z^6(m_x^2 + m_y^2)) \\
& - 228m_x^2m_y^2m_z^2|\mathbf{m}|^2 + 621(m_x^4m_y^4 + m_y^4m_z^4 + m_x^4m_z^4)] \\
& + \frac{1}{2048\pi|\mathbf{m}|^{19}} [2588(m_x^{12} + m_y^{12} + m_z^{12}) \\
& - 65676(m_x^{10}m_y^2 + m_x^2m_y^{10} + m_x^{10}m_z^2 + m_x^2m_z^{10} + m_y^{10}m_z^2 + m_y^2m_z^{10}) \\
& + 426144(m_x^8m_y^4 + m_x^4m_y^8 + m_x^8m_z^4 + m_x^4m_z^8 + m_y^8m_z^4 + m_y^4m_z^8) \\
& - 712884(m_x^6m_y^6 + m_x^6m_z^6 + m_y^6m_z^6) \\
& - 62892(m_x^8m_y^2m_z^2 + m_x^2m_y^8m_z^2 + m_x^2m_y^2m_z^8) \\
& - 297876(m_x^6m_y^4m_z^2 + m_x^6m_y^2m_z^4 + m_x^4m_y^6m_z^2 \\
& + m_x^2m_y^6m_z^4 + m_x^2m_y^4m_z^6 + m_x^4m_y^2m_z^6) \\
& + 2507340(m_x^4m_y^4m_z^4)] + \mathcal{O}\left(\frac{1}{|\mathbf{m}|^9}\right). \tag{C.10}
\end{aligned}$$

As in 2D, we evaluate the integral for small $|\mathbf{m}| \leq 32$ and we evaluate the expansion for $|\mathbf{m}| > 32$.

Immersed Interface stencils

D.1 Polynomial interpolation

The Immersed Interface tool relies on interpolation coefficients to extrapolate/interpolate information from the flow up to the boundary. We here present the polynomial reconstruction of the information, as used in the present work.

Let us consider a general example with a set of 3 points (x_1 , x_2 and x_3) with corresponding value (U_1 , U_2 and U_3) and a boundary condition of degree k , $U_\alpha^{(k)}$, located at x_α . For our application, we look for the linear combination of $U_\alpha^{(k)}$, U_1 , U_2 and U_3 to obtain the value of the d th derivative of the field, at x_α . We then build the following system using Taylor series and $\Delta_i = x_i - x_\alpha$,

$$\sum_{i=1}^3 (c_i U_i) + c_\alpha U_\alpha^{(k)} = u_\alpha [c_1 + c_2 + c_3 + \delta_{0k} c_\alpha] \quad (\text{D.1})$$

$$+ u_\alpha^{(1)} [c_1 (\Delta_1) + c_2 \Delta_1 + c_3 \Delta_1 + \delta_{1k} c_\alpha] \quad (\text{D.2})$$

$$+ u_\alpha^{(2)} \left[c_1 \frac{(\Delta_1)^2}{2} + c_2 \frac{(\Delta_2)^2}{2} + c_3 \frac{(\Delta_3)^2}{2} + \delta_{2k} c_\alpha \right] \quad (\text{D.3})$$

$$+ u_\alpha^{(3)} \left[c_1 \frac{(\Delta_1)^3}{6} + c_2 \frac{(\Delta_2)^3}{6} + c_3 \frac{(\Delta_3)^3}{6} + \delta_{3k} c_\alpha \right] . \quad (\text{D.4})$$

The accuracy of such a scheme is given by $\mathcal{O}(h^{4-q})$. To obtain the interpolation coefficient, one solves the following system

$$\begin{bmatrix} \delta_{0k} & 1 & 1 & 1 \\ \delta_{1k} & (\Delta_1) & (\Delta_2) & (\Delta_3) \\ \delta_{2k} & \frac{(\Delta_1)^2}{2} & \frac{(\Delta_2)^2}{2} & \frac{(\Delta_3)^2}{2} \\ \delta_{3k} & \frac{(\Delta_1)^3}{6} & \frac{(\Delta_2)^3}{6} & \frac{(\Delta_3)^3}{6} \end{bmatrix} \begin{bmatrix} c_\alpha^q \\ c_1^q \\ c_2^q \\ c_3^q \end{bmatrix} = \begin{bmatrix} \delta_{0q} \\ \delta_{1q} \\ \delta_{2q} \\ \delta_{3q} \end{bmatrix}. \quad (\text{D.5})$$

As an example, we detail the computation of the interpolation coefficients, S_i^q , with $k = 0$, as used in the IIVPM method. For each derivative, q , the coefficients are the solution of the following system:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & (\Delta_1) & (\Delta_2) & (\Delta_3) \\ 0 & \frac{(\Delta_1)^2}{2} & \frac{(\Delta_2)^2}{2} & \frac{(\Delta_3)^2}{2} \\ 0 & \frac{(\Delta_1)^3}{6} & \frac{(\Delta_2)^3}{6} & \frac{(\Delta_3)^3}{6} \end{bmatrix} \begin{bmatrix} c_\alpha^q \\ c_1^q \\ c_2^q \\ c_3^q \end{bmatrix} = \begin{bmatrix} \delta_{0q} \\ \delta_{1q} \\ \delta_{2q} \\ \delta_{3q} \end{bmatrix}. \quad (\text{D.6})$$

D.2 Summary of the Immersed Interface stencil used in 2D

In this section, we list all the encountered cases and the interpolation that we use for every sub-step of the Immersed Interface method:

- **First derivative - Dirichlet boundary condition:** wall vorticity

$$\left. \frac{\partial^k u}{\partial \xi^k} \right|_\alpha = S_\alpha^{\xi,k} u_\alpha + \sum_{i=1}^2 S_i^{\xi,k} u(\xi_{i+1}, \eta_0) + \mathcal{O}(h^{3-k}) \quad (\text{D.7})$$

$$\left. \frac{\partial^k u}{\partial \eta^k} \right|_\alpha = S_\alpha^{\eta,k} u_\alpha + \sum_{i=1}^2 S_i^{\eta,k} u(\xi_\alpha, \eta_i) + \mathcal{O}(h^{3-k}) \quad (\text{D.8})$$

$$u(\xi_\alpha, \eta_j) = \sum_{i=1}^3 T_i^{\xi,0} u(\xi_{i-2+s_{\eta_j}}, \eta_j) + \mathcal{O}(h^3) \quad (\text{D.9})$$

- **Second derivative - Dirichlet boundary condition:** Poisson , diffusion

$$\left. \frac{\partial^k u}{\partial \xi^k} \right|_\alpha = S_\alpha^{\xi,k} u_\alpha + \sum_{i=1}^3 S_i^{\xi,k} u(\xi_{i+1}, \eta_0) + \mathcal{O}(h^{4-k}) \quad (\text{D.10})$$

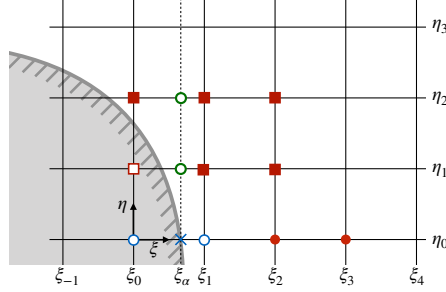


Figure D.1: **IIM stencils** - wall vorticity. The mesh intersects the interface and generates one control point ($\times$), two discontinuity affected points (or biased points) ($\circ$) and primary interpolation points in the current direction, ξ , and in the transverse direction, η (represented as $\bullet$ and $\circ$, respectively). Primary interpolation points in the η direction are computed as a polynomial interpolated values using secondary interpolation points ($\blacksquare$ and $\square$).

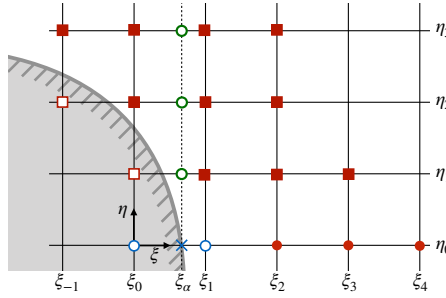


Figure D.2: **IIM stencils** - Poisson and diffusion. The mesh intersects the interface and generates one control point ($\times$), two discontinuity affected points (or biased points) ($\circ$) and primary interpolation points in the current direction, ξ , and in the transverse direction, η (represented as $\bullet$ and $\circ$, respectively). Primary interpolation points in the η direction are computed as a polynomial interpolated values using secondary interpolation points ($\blacksquare$ and $\square$).

- **First derivative - no boundary condition:** Hamilton - Jacobi

$$\left. \frac{\partial^k u}{\partial \xi^k} \right|_\alpha = \sum_{i=1}^3 S_i^{\xi, k} u(\xi_{i+1}, \eta_0) + \mathcal{O}(h^{3-k}) \quad (\text{D.11})$$

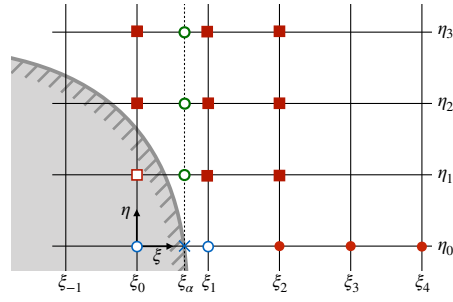


Figure D.3: **IIM stencils** - Hamilton-Jacobi. The mesh intersects the interface and generates one control point ($\times$), two discontinuity affected points (or biased points) ($\circ$) and primary interpolation points in the current direction, ξ , and in the transverse direction, η (represented as $\bullet$ and $\circ$, respectively). Primary interpolation points in the η direction are computed as a polynomial interpolated values using secondary interpolation points ($\blacksquare$ and $\square$).

Wavelet theory and multiresolution analysis

E.1 Multiresolution analysis

We here give a concise description of the multiresolution analysis, as introduced by Mallat [139] and presented in van Rees [59]. We refer the reader to the work of Daubechies [140], Sweldens [141] and references therein for the underlying theory and more details.

The multiresolution theory is based on a decomposition of the space $L_2(\mathbb{R})$ ¹ as a collection of closed spaces, V_j with $j \in \mathbb{Z}$ such that

$$V_j \subset V_{j+1} \quad . \quad (\text{E.1})$$

The vector space V_j can be interpreted as the set of all possible approximations at resolution 2^j of the functions in $L_2(\mathbb{R})$. The causality property, Eq. (E.1), ensures that the resolution 2^{j+1} contains all the necessary information to approximate the same function at a lower resolution, 2^j . The union $\cup_j V_j$ is dense in $L_2(\mathbb{R})$ and their intersection is empty, $\cap_{j \in \mathbb{Z}} V_j = \emptyset$. This collection of subspaces also satisfies the dilatation property,

$$f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1} \quad . \quad (\text{E.2})$$

Finally, there exists a unique function, $\phi(x) \in L_2(\mathbb{R})$, called the scaling function, such that if we define, for each $j \in \mathbb{Z}$, the family of function

$$\phi_{j,k} = \sqrt{2^j} \phi(2^j x - k) \quad , \quad (\text{E.3})$$

¹ $L_2(\mathbb{R})$ denotes the vector space of the measurable, square-integrable one dimensional functions $f(x)$

is an orthonormal basis of V_j [139].

The difference between V_j and V_{j+1} is characterized by a new subspace, W_j such that

$$W_j \perp V_j \quad \text{and} \quad V_{j+1} = V_j \oplus W_j . \quad (\text{E.4})$$

Alike the subspace V_j , the family of function

$$\psi_{j,k} = \sqrt{2^j} \psi(2^j x - k) , \quad (\text{E.5})$$

is an orthonormal basis of W_j and $\psi(x)$ is called the orthogonal wavelet. This basis functions are linked one to each other by the refinement relation [139], which gives to the following identities,

$$\phi(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \phi(2x - k) \quad (\text{E.6a})$$

$$\psi(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} g_k \phi(2x - k) \quad (\text{E.6b})$$

where the filters h and g defined as²

$$h_k = \langle \phi(x) , \sqrt{2} \phi(2x - k) \rangle \quad (\text{E.7a})$$

$$g_k = \langle \psi(x) , \sqrt{2} \phi(2x - k) \rangle . \quad (\text{E.7b})$$

Finally, let us note that since ψ and ϕ are orthogonal, $\langle \psi(x) , \phi(x) \rangle = 0$, we have that $g(n) = (-1)^n h(1 - n)$.

E.2 Wavelet Decomposition

Using the approach mentioned above, we can reproject any function $f(x)$ into V_j using,

$$f_j(x) = \sum_{k=-\infty}^{\infty} \langle f(x) , \phi_{j,k} \rangle \phi_{j,k} . \quad (\text{E.8})$$

Using the properties of V_j , V_{j+1} and W_j , we also obtain

$$f_{j+1}(x) = \sum_{k=-\infty}^{\infty} \langle f(x) , \phi_{j+1,k} \rangle \phi_{j+1,k} \quad (\text{E.9a})$$

$$= \sum_{k=-\infty}^{\infty} \langle f(x) , \phi_{j,k} \rangle \phi_{j,k} + \sum_{k=-\infty}^{\infty} \langle f(x) , \psi_{j,k} \rangle \psi_{j,k} . \quad (\text{E.9b})$$

²We here follow the notation of van Rees [59], which is not the same as the one proposed by Sweldens [141]. The filter coefficients referenced in both works linked by $\tilde{h}_k^{\text{van Rees}} = 2 h_k^{\text{Sweldens}}$.

The wavelet decomposition of the function $f(x)$ is then given with respect to a reference resolution J as,

$$f(x) = \sum_{k=-\infty}^{\infty} c_k^J \phi_{J,k}(x) + \sum_{j=J}^{\infty} \sum_{k=-\infty}^{\infty} d_k^j \psi_{j,k}(x) , \quad (\text{E.10})$$

where the coefficients c_k^J and d_k^j are called the scaling and the detail coefficient, respectively. They are given by

$$c_k^J = \langle f(x), \phi_{J,k} \rangle \quad (\text{E.11a})$$

$$d_k^j = \langle f(x), \psi_{j,k} \rangle \quad (\text{E.11b})$$

The scaling and detail coefficient are obtained using a recursive relation, obtained by combining Eq. (E.11) and Eq. (E.6),

$$c_k^j = \sum_n h_{n-2k} c_n^{j+1} \quad (\text{E.12a})$$

$$d_k^j = \sum_n g_{n-2k} c_n^{j+1} . \quad (\text{E.12b})$$

This operation is referred to as the Fast Wavelet Transform (FWT) and is related to a downsampling. The inverse Fast Wavelet Transform is obtained as

$$c_k^{j+1} = \sum_n (h_{k-2n} c_n^j + g_{k-2n} d_n^j) . \quad (\text{E.13})$$

E.3 Bi-orthogonal wavelets

The restriction on the orthogonal wavelets has been alleviate by the introduction of the bi-orthogonality in the wavelet basis for V_j and W_j [142]. The bi-orthogonality of the basis introduces dual spaces, $\tilde{V}_j$ and their associated basis, $\tilde{\phi}$ so that $\langle \phi, \tilde{\phi}(x-k) \rangle = \delta_k$. As in the orthogonal case, the basis functions are linked by the following scaling relations,

$$\phi(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \phi(2x-k) \quad (\text{E.14a})$$

$$\tilde{\phi}(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} \tilde{h}_k \phi(2x-k) \quad (\text{E.14b})$$

$$\psi(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} g_k \phi(2x-k) \quad (\text{E.14c})$$

$$\tilde{\psi}(x) = \sqrt{2} \sum_{k=-\infty}^{\infty} \tilde{g}_k \tilde{\phi}(2x-k) , \quad (\text{E.14d})$$

The orthogonality conditions require that

$$g(n) = (-1)^n \tilde{h}(1-n) \quad \text{and} \quad \tilde{g}(n) = (-1)^n h(1-n) . \quad (\text{E.15})$$

Using bi-orthogonal functions, the wavelet decomposition is equivalently given by [59],

$$f(x) = \sum_{k=-\infty}^{\infty} c_k^J \tilde{\phi}_{J,k}(x) + \sum_{j=J}^{\infty} \sum_{k=-\infty}^{\infty} d_k^j \tilde{\psi}_{j,k}(x) \quad (\text{E.16a})$$

$$= \sum_{k=-\infty}^{\infty} \tilde{c}_k^J \phi_{J,k}(x) + \sum_{j=J}^{\infty} \sum_{k=-\infty}^{\infty} \tilde{d}_k^j \psi_{j,k}(x) \quad (\text{E.16b})$$

with

$$c_k^J = \langle f(x), \phi_{J,k} \rangle \quad (\text{E.17a})$$

$$\tilde{c}_k^J = \langle f(x), \tilde{\phi}_{J,k} \rangle \quad (\text{E.17b})$$

$$d_k^j = \langle f(x), \psi_{j,k} \rangle \quad (\text{E.17c})$$

$$\tilde{d}_k^j = \langle f(x), \tilde{\psi}_{j,k} \rangle \quad (\text{E.17d})$$

The Fast Wavelet Transform is finally given by

$$c_k^j = \sum_n h_{n-2k} c_n^{j+1} \quad (\text{E.18a})$$

$$d_k^j = \sum_n g_{n-2k} c_n^{j+1} \quad (\text{E.18b})$$

while the inverse Fast Wavelet Transform is obtained as

$$c^{j+1} = \sum_n (\tilde{h}_{k-2n} c_n^j + \tilde{g}_{k-2n} d_n^j) . \quad (\text{E.19})$$

E.4 Interpolating wavelets

Interpolating wavelets take their origin in the work of Deslauriers and Dubuc [128] where they propose a recursive way to interpolate a function on a grid: Given a function, $f_i^k = f(x_i)$ with $i = 0, \dots, 2^k$, we are looking for the value of that function on a finer mesh, i.e.

$$f_j^{k+1} = f(x_j) \quad \text{with} \quad j = 0, \dots, 2^{k+1} . \quad (\text{E.20})$$

For every new point j , such that $\text{mod}(j, 2) \neq 0$ at level $k+1$, we construct a d polynomial interpolant, π_k^d satisfying $p = d+1$ relations:

$$f_{\lfloor j/2 \rfloor + m}^k = \pi_k^d(x_{\lfloor j/2 \rfloor + m}) \quad \text{with} \quad m = [-p/2 + 1; p/2] . \quad (\text{E.21})$$

k	-4	-3	-2	-1	0	1	2	3
h_k^p					1			
$\tilde{g}_k^p$					1			
g_k^2			-1/2	1	-1/2			
$\tilde{h}_k^2$				1/2	1	1/2		
g_k^4	1/16	0	-9/16	1	-9/16	0	1/16	
$\tilde{h}_k^4$		-1/16	0	9/16	1	9/16	0	-1/16

Table E.1: Interpolating bi-orthogonal wavelet filter coefficients for a $p - 1$ order polynomial. We refer to [141] for the coefficients value for higher order polynomials.

Finally, we use that polynomial to interpolate the value at level $k + 1$: $f_j^{k+1} = \pi_k^d(x_{j+m})$. Applying this procedure recursively to $f(x_j) = \delta_j$ leads to an interpolation kernel, defined by the degree of the polynomial used in the recursion, ϕ_d , see [143]. The recursive application from one resolution 2^j to another has been shown to be equivalent to an FWT-like approach [143]. The equivalence is not strict as this transform is defined only on point value rather than integrals. However, most of the properties remain valid, such as the use of 4 bi-orthogonal filters to describe the recursion, see van Rees [59] for an examples of the filters obtained for ϕ_3 .

In the case of interpolating wavelets, one may build the Fast Wavelet Transform by considering the interpolation from one resolution, $j + 1$ to a coarser one, j . The detail coefficient at the coarser level is the same as the scaling coefficient at the finer level,

$$c_k^j = \sum_n h_{n-2k} c_n^{j+1} = c_{2k}^{j+1} . \quad (\text{E.22})$$

The detail coefficient is the difference between the finer scaling coefficient and the value created by interpolation from a coarser level using a third order polynomial,

$$d_k^j = \sum_n g_{n-2k} c_n^{j+1} = c_{2k-1}^{j+1} - \left(-\frac{1}{16} c_{2k-4}^{j+1} + \frac{9}{16} c_{2k-2}^{j+1} + \frac{9}{16} c_{2k}^{j+1} - \frac{1}{16} c_{2k+2}^{j+1} \right) . \quad (\text{E.23})$$

Using a third order polynomial interpolation is equivalent to apply the filter coefficient to the scaling and detail coefficient, see Table E.1

Finally, we would like to mention two other interpolating approach presented in the literature. The first one is the interpolating wavelet on the interval, which replaces a decentered interpolation and avoids very large ghost value computations. This procedure, is detailed by Donoho [144], Sweldens [145]. The second is the lifted scheme,

proposed by Sweldens [141, 145], which can be used to obtain the second generation interpolation wavelets which satisfy the moment conservation.