Development of numerical model for green water loading by coupling the mesh based flow models with the meshless models

Bašić, Josip

University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture

Josip Bašić

Development of Numerical Model for Green Water Loading by Coupling the Mesh Based Flow Models with the Meshless Models

DOCTORAL THESIS

Zagreb, 2019

University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture

Josip Bašić

Development of Numerical Model for Green Water Loading by Coupling the Mesh Based Flow Models with the Meshless Models

DOCTORAL THESIS

Supervisor: prof. Nastia Degiuli, PhD

Zagreb, 2019

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

Josip Bašić

Razvoj numeričkog modela opterećenja zalijevanja palube spregom mrežnih i bezmrežnih modela strujanja

DOKTORSKI RAD

Mentorica: prof. dr. sc. Nastia Degiuli

Zagreb, 2019.

BIBLIOGRAPHY DATA

UDC:	629.5 : 532.5
Keywords:	green water; marine hydrodynamics; free surface flows; meshless method; Lagrangian method; finite differences; domain decomposition
Scientific area:	Technical sciences
Scientific field:	Naval architecture
Institution	Faculty of Mechanical Engineering and Naval Architecture (FMENA), University of Zagreb
Supervisor:	Prof. Nastia Degiuli, PhD
Number of pages:	221
Number of figures:	95
Number of tables:	12
Number of references:	227
Date of examination:	29th May 2019
Jury members:	Prof. Željko Tuković, PhD, University of Zagreb Prof. Šime Malenica, PhD, University of Zagreb Prof. Dario Ban, PhD, University of Split
Archive:	Faculty of Mechanical Engineering and Naval Architecture (FMENA), University of Zagreb, Croatia

Acknowledgements

I would like to express my most special appreciation and thanks to my advisor prof. Nastia Degiuli, you have been a tremendous mentor for me. Thank you for encouraging me in my research and for allowing me to grow as a research scientist. Your advice on research as well as on my career have been priceless.

I would like to thank Šime Malenica for discussing and supporting my atypical ideas and encouraging me to direct them into solving actual problems, such as the green water.

Many thanks goes to my colleague, prof. Dario Ban, for his support and giving me boost when needed with his knowledge of mathematics, meshless theory and lamb.

I wish to record my gratitude to prof. Željko Tuković for discussing important matters considering the numerics and implementation, which has helped me revisit my ideas.

Thanks goes to all fellow researchers who have graciously shared their results, ideas and comments with me: prof. Kalman Žiha, Inno Gatin, Charles Monroy, prof. Shin Hyung Rhee, and Jeonghwa Seo. Thanks also goes to Ivana Martić, who often dedicated her time to help me with various things, and Andrea Farkas who was always free for discussing.

Words cannot express how grateful I am to Šef, Boke, Martina, and of course my family, all of whom have made sacrifices on my behalf. Your encouragement and prayer for me was what sustained me thus far.

Statement of Originality

I hereby declare that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Josip Bašić

Abstract

The interaction between a moving vessel and incident waves leads to large relative motions and strong nonlinearities. This can result in violent water dynamics so that water flows onto the deck of the vessel, known as green water, and reaches crucial equipment and other deck structures. Green water events are considered as serious threat to the stability and operability of vessels, which should be reliably predicted and properly assessed in the design stage. Due to complexity of the problem, classification rules are limited in predicting the loads during the green water event.

This thesis describes a novel method that is developed for simulating incompressible flows for the purpose of predicting green water events. The method is: meshless, Lagrangian, volume–conservative, second–order accurate in space, efficient, and suitable for coupling. The foundation of the method is a set of novel spatial operators based on the weighted–least squares, which are used to describe and solve the Navier–Stokes equations in strong form. Volume–conservative Lagrangian advection is used, which naturally handles violent free–surface flows. Boundary conditions are conforming to moving geometry at each time step. This makes coupling with other mesh–based and structural solvers straightforward.

A completely parallel and efficient implementation of the methodology is described, which is validated by simulating cavity–flow, slamming, dam–breaking and sloshing experiments. The methodology is also validated by simulating both isolated and periodic green water events simulated in a domain–decomposed environment. The computed kinematics and dynamics of the flows compare well with experimental data and results obtained by other numerical methods.

Keywords: green water; marine hydrodynamics; free surface flows; meshless method; Lagrangian method; finite differences; domain decomposition.

Prošireni sažetak

Uvod

Nepogodni okolišni uvjeti, tj. valovi, vjetar i morske struje induciraju velika gibanja brodova i pomorskih objekata. Osim prekooceanskih brodova, uzimajući u obzir nove granice za eksploataciju nafte, plutajuće jedinice za skladištenje i prekrcaj (eng. Floating Storage and Offloading, FSO) te plutajuće jedinice za proizvodnju, skladištenje i prekrcaj (eng. Floating Production Storage and Offloading, FPSO) su izložene takvim uvjetima. Međudjelovanjem okolišnih uvjeta i velikih gibanja broda, nailazeći valovi mogu premašiti nadvođe i zalijevati palubu. Uslijed relativnih gibanja morske vode u odnosu na konstrukciju, pri zalijevanju palube udari vode mogu biti dovoljno snažni da oštete opremu na palubi, palubu, nadgrađe, ili pak mogu narušiti stabilitet broda. Osim dinamike udara morske vode pri zalijevanju palube, morska voda koja ostane na palubi doprinosi narušavanju stabiliteta zbog povećavanja težišta mase sustava i utjecaja slobodne površine. Ovisno o procjeni kapetana, značajno zalijevanje palube pri plovidbi ublažava se smanjenjem brzine plovidbe ili promjenom kursa. Međutim, pri projektiranju je potrebno pažljivo izvršiti analizu za određeno stanje mora uključujući neizvjesnost opterećenja uslijed zalijevanja palube, koja mogu narušiti stabilitet broda ili oštetiti opremu na palubi i nadgrađe. Zbog složenosti problema koji uključuje hidroelastičnu konstrukciju sa šest stupnjeva slobode gibanja u međudjelovanju s nepravilnim valovima, pravila klasifikacijskih društava ograničena su pri predviđanju opterećenja udara vode uslijed zalijevanja palube. Također, postojeći eksperimentalni pristupi su preskupi i složeni, a numeričke metode ili ne daju adekvatnu točnost rezultata ili su previše zahtjevne za proračun. U ovom doktorskom radu predložena je nova numerička metoda, koja je pogodna za numeričko simuliranje problema zalijevanja palube.

Matematički model

Strujanje fluida opisano je Navier–Stokesovim jednadžbama u Lagrangeovom obliku. Takav prirodni oblik gibanja ne sadrži konvekcijski član, koji opisuje razliku gibanja promatrača i čestica fluida. Ako se pretpostavi da je polje tlaka funkcija polja brzina, tada

se uvjet nestlačivosti može uključiti u Poissonovu jednadžbu tlaka. Poissonova forma Navier–Stokesovih jednadžbi nema nedostatke kao projekcijske sheme te je efikasnija.

Rješavanjem Poissonove jednadžbe u jakoj formulaciji te jednadžbe očuvanja gibanja, gibanje fluida prikazano je integracijom brzine i položaj domene fluida u vremenu. Lagrangeov opis strujanja može kvalitetno odrediti advekciju fluida i njegove slobodne površine.

Kako bi se navedeni matematički problem analizirao u što kraćem vremenu, koristi se dekompozicija trodimenzijske domene problema. Pomorstvenost broda se rješava linearnom ili nelinearnom potencijalnom teorijom te se strujanje tekućine u blizini trupa broda, koje prelazi nadvođe broda, koristi kao ulaz u lokalni nelinearni rješavač Navier–Stokesovih jednadžbi za predviđanje daljnjeg tijeka zalijevanja palube broda.

Numerička metoda

Volumen fluida u potpuno bezmrežnoj metodi prikazan je nizom, odnosno oblakom točaka bez ikakve topološke povezanosti. Točke se slobodno gibaju, a prilikom međudjelovanja točaka određena točka utječe na prsten točaka koje je okružuju. Metoda se temelji na međudjelovanju susjednih točaka te novih diskretnih prostornih operatora, tj. prvih i drugih derivacija, koji su temeljeni na metodi najmanjih kvadrata. Diskretni operatori se koriste za opis Navier–Stokesovih jednadžbi u jakoj formulaciji. Dakle, Poissonova jednadžba tlaka direktno je diskretizirana s novouvedenim operatorima.

Kako bi se adekvatno postavio sustav jednadžbi, oblak točaka se omeđuje nizom točaka koje služe za narinjavanje rubnih uvjeta. U svakom vremenskom koraku točke fluida blizu zida se projiciraju na zid, gdje se generiraju rubne točke za trenutni vremenski korak. Kako je vektor projiciranja uvijek okomit na projiciranu plohu, moguće je točno diskretizirati rubne uvjete u smjeru normale na plohu.

Rubni uvjeti ulaza i izlaza tekućine iz domene se definiraju preko virtualno postavljenih stijenki. Na tim stijenkama rubni uvjeti se mogu postavljati u bilo kojem vremenskom trenutku, kako bi se omogućilo spajanje bezmrežnog rješavača na rubu proračunske domene s nekim drugim rješavačem izvan te domene. Vrijednosti hidrodinamičkih veličina na dodirnom mjestu dviju domena se interpoliraju u svrhu prenošenja informacija iz jedne domene u drugu. Tako se komunikacija između domena može vršiti neovisno o metodi i diskretizaciji, koju koristi rješavač vanjske proračunske domene.

Nakon generiranja rubnih točaka, identificiraju se točke na slobodnoj površini pomoću kojih se mora narinuti atmosferski tlak na slobodnoj površini. Navedeni Dirichletov uvjet osigurava jedinstveno rješenje sustava jednadžbi. Test kojim se provjerava leži li točka na slobodnoj površini koristi svojstva novouvedenog Laplaceovog operatora. Matrica sustava jednadžbi jest M–matrica, koja osigurava dobra konvergencijska svojstva.

Nakon rješavanja jednadžbe tlaka, aproksimiraju se članovi vremenske derivacije brzine, gradijenta tlaka i difuzijskog člana u Navier–Stokesovoj jednadžbi. Na taj način kao nepoznanica u običnoj diferencijalnoj jednadžbi preostaje jedino brzina, koja se eksplicitno izračunava. Bezmrežne točke koje opisuju fluid napreduju u prostoru izračunatom brzinom.

Poznato je da točke u Lagrangeovom opisu strujanja prate strujnice te se razdvajaju ili sakupljaju, što narušava zakon očuvanja mase. Nakon eksplicitnog pomaka, točke oblaka se organiziraju na način da svaka točka zauzima jednaki volumen što se osigurava rješavanjem optimizacijskog problema Jacobijevim iteracijama dok se ne zadovolji kriterij jednake udaljenosti između susjednih točaka. Hidrodinamičke veličine na novim položajima, nakon reorganizacije točaka dobivaju se interpolacijom na neorganiziranom oblaku točaka. Pokazano je kako ovakav način prikaza strujanja može opisati impulzivne udare, pri kojima je potrebno dopustiti određenu stlačivost fluida.

Implementacija

Pri razvoju metode, napravljena je generička računalna biblioteka za implementaciju bezmrežnih algoritama, koristeći moderne C++ meta–programske tehnike, koje omogućuju jedan, a optimalan kod za različite ulazne parametre, poput broja dimenzija.

Svi implementirani algoritmi izvode se paralelno na centralnim i grafičkim jedinicama za obradu podataka. Prikazano je kako metoda ima odlična konvergencijska svojstva te je pogodna za paralelno izvršavanje, što rezultira numeričkim simulacijama u kratkom vremenu na današnjim računalima.

Verifikacija i validacija

Pomoću računalne implementacije provedena su istraživanja numeričkim simuliranjem različitih problema za koje su već poznata numerička rješenja ili su provedeni eksperimenti. Novouvedeni diskretni Laplaceov operator je verificiran na nizu umjetno stvorenih problema, koji uključuju aproksimaciju Laplaceovog operatora i rješavanje Poissonove jednadžbe. Implementirani rješavač je primijenjen za simulaciju uobičajenog pokusa za verifikaciju rješavača računalne dinamike fluida bez utjecaja slobodne površine tj. pokusa strujanja u šupljini s pomičnim poklopcem. Validacija impulzivnog razvoja tlaka izvršena je simuliranjem udaranja klina o tekućinu te udaranja pramca kontejnerskog broda. U oba slučaja numerički razvoj tlaka je uspješno reproducirao eksperimentalne podatke. Povrh toga, simulirani su eksperimenti pucanja brane te zapljuskivanja tekućine u gibajućim tankovima, kako bi se pokazalo da metoda može simulirati snažne udare tekućine o krute

stijenke, pri kojima uspješno reproducira udarne tlakove. Potom su provedene numeričke simulacije koje opisuju problem zalijevanja palube. Izolirani događaji zalijevanja palube su simulirani na sličan način kao i pucanje brane, pri čemu su stvoreni valovi koji se lome preko palube. Također je uspoređena sila na palubu dobivena numeričkim simulacijama s eksperimentalnim vrijednostima. Najvažniji validacijski pokus izvršen u ovom doktorskom radu je simulacija FPSO modela na pravilnim valovima dekompozicijom domene. Numerički razvoj tlakova na palubi tijekom zalijevanja zadovoljavajuće prati eksperimentalni razvoj tlakova. Također, prikazano je kako se osim valova mogu narinuti i gibanja broda, koja značajno utječu na opterećenja uslijed zalijevanja vode na palubu.

Zaključci

Doneseni su sljedeći zaključci:

- Poissonova forma Navier–Stokesovih jednadžbi u Lagrangeovom opisu strujanja je točnija i stabilnija od projekcijskih shema.
- Novouvedeni diskretni prostorni operatori u obliku konačnih razlika su stabilni te rezultiraju dobrom konvergencijom pri rješavanju Navier–Stokesovih jednadžbi.
- Fluid se prilagođava geometriji tako da se domena fluida projicira na geometriju definirajući rubne uvjete u svakom vremenskom koraku.
- Lagrangeov opis strujanja prirodno upravlja razvojem slobodne površine složenog oblika. Svojstva Laplaceovog operatora mogu poslužiti za prepoznavanje točaka na slobodnoj površini.
- Zakon očuvanja mase u Lagrangeovom opisu strujanja adekvatno se postiže optimiranjem udaljenosti između susjednih točaka.
- Lagrangeove metode moraju imati kontrolu nad ulaznim i izlaznim granicama uklanjajući i generirajući diskretne točke fluida po potrebi, što je preduvjet za dekompoziciju domene.
- Generalizirane ulazne i izlazne granice te bezmrežna aproksimacija omogućuju da se domena bezmrežne metode može spregnuti s bilo kojom metodom koja rješava vanjsku domenu.
- Metoda temeljena na opisanim postavkama uspješno reproducira eksperimente zalijevanja palube, udaranja pramca i zapljuskivanja u tankovima, čime je dokazano da je metoda pogodna za simuliranje problema sa snažnim udarima tekućine.
- Metoda ne ovisi o diskretizaciji geometrije te je pogodna za spregu s rješavačima elastične strukture.

Konačni zaključak: bezmrežne aproksimacije konačnim razlikama u Lagrangeovom opisu strujanja mogu adekvatno riješiti Navier–Stokesove jednadžbe i prikazati strujanje sa slobodnom površinom uz jednostavnu spregu s drugim rješavačima, a u svrhu analize opterećenja broda ili plutajuće konstrukcije uslijed nailaska valova.

Ključne riječi: zalijevanje palube; brodska i pučinska hidrodinamika; strujanje sa slobodnom površinom; bezmrežna metoda; Lagrangeov opis strujanja; metoda konačnih razlika; dekompozicija domene.

Contents

List of Figures

List of Tables

Nomenclature

Latin

$\boldsymbol{a}$ acceleration vector

$\mathbf{A}$ coefficients matrix for the Poisson system

A_z amplitude of the vertical ship motion

b right-hand-side value of the equation

$\mathbf{b}$ vector containing the right-hand-side values

$\mathbf{B}_i$ renormalisation tensor for the gradient

$\widehat{\mathbf{B}}_i$ renormalisation tensor for the Laplacian

$\widetilde{\mathbf{B}}_i$ approximate renormalisation tensor for the Laplacian

$\mathcal{B}$ subset of neighbour boundary points

$c_{\min}$ minimum allowed relative speed in the neighbourhood

$c_{\max}$ maximum relative speed in the neighbourhood

c_S pressure propagation velocity

D Lagrangian derivative

d number of dimensions (2 or 3)

$\boldsymbol{e}$ unit vector

$\boldsymbol{f}$ external acceleration vector, usually the gravitational acceleration

$\boldsymbol{F}$ resultant force vector

$\mathcal{F}$ subset of fluid neighbour points

fb freeboard height

g gravitational constant

$\boldsymbol{g}$ velocity vector at the boundary

$\boldsymbol{g}_{COG}$ body velocity vector at its COG

$\overline{GM}$ metacentric height

H_W wave height

h_E freeboard exceedance height

h_{FB} height of the freeboard

$\boldsymbol{I}$ identity vector

$\mathbf{I}_{COG}$ tensor of the moment of inertia about the COG

k wave number

$\boldsymbol{l}$ vector of second derivatives

m mass of the body

$\boldsymbol{m}$ vector containing mixed derivatives of the Hessian

$\boldsymbol{n}$ surface normal vector pointing inside the fluid

n number of the time step

$\mathcal{N}$ set of neighbour points

$\boldsymbol{o}$ offset vector considering neighbours locations

p pressure

$\boldsymbol{p}$ vector containing mixed derivatives of the Hessian

$\mathbf{p}$ vector containing pressure solutions

P function of the velocity that yields the pressure

$\boldsymbol{q}$ second–order correction vector for Laplacian

$\boldsymbol{Q}$ resultant torque

r_t ratio of the current and previous time step values

r_w distance from a point to the wall

r_W local relative wave elevation

R Lagrange remainder term

$\boldsymbol{s}$ squared components of a location vector

t time instant

U ship speed

$\boldsymbol{u}$ velocity vector

$\boldsymbol{u}_0$ initial velocity vector

$\boldsymbol{u}^\star$ intermediate velocity vector

$\boldsymbol{u}_{irr}$ irrotational velocity vector field

$\boldsymbol{u}_{sol}$ solenoidal velocity vector field

u velocity vector x component

v velocity vector y component

w velocity vector z component

W weighting factor

$\boldsymbol{x}$ position vector

Greek

$\boldsymbol{\alpha}$ body angular acceleration vector

α_R zonal relaxation parameter

Γ boundaries of the fluid domain

δ discrete fraction of

Δ grid spacing, spacing between points

$\boldsymbol{\varepsilon}$ error

η incident wave elevation

θ cone threshold angle

λ_B coefficient for the source term at boundaries

λ_C multiplier for the fluid source term

λ_F coefficient for the source term in fluid

λ_{FS} portion of regular volume, defining free surface threshold

λ_L coefficient accounting for nonlinearity of the Laplacian along the normal

λ_W incident wavelength

ν fluid kinematic viscosity

ξ_W phase angle of the incident wave

ξ_z phase angle of the vertical ship motion

ρ fluid density

τ physical time

ϕ potential function

ϕ_s surface potential function

ψ normalised weighting factor

$\boldsymbol{\omega}$ body angular velocity vector

ω_W incident wave frequency

Ω fluid domain volume

Miscellaneous

∂ continuous partial derivative

∇ gradient operator

$\nabla\cdot$ divergence operator

$\nabla\times$ curl operator

∇^2 Laplace operator

$\cdot$ inner product

$\equiv$ equivalence

$\langle\rangle$ discrete version of

$|_\Gamma$ evaluated at the boundary

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
ALE	Arbitrary Lagrangian–Eulerian
AP	Aft perpendicular
BDF	Backward Differencing
BEM	Boundary Element Method
BiCGStab	Bi–Conjugate Gradient Stabilised
BC	Boundary condition
BIV	Bubble Image Velocimetry
CFL	Courant–Friederich–Lewy
CFD	Computational Fluid Dynamics
CG	Conjugate Gradient
COG	Centre of Gravity
CPU	Central Processing Unit
DD	Domain Decomposition
DNS	Direct Numerical Simulation
DOF	Degrees of Freedom
FD	Finite Difference
FDM	Finite Difference Method
FEM	Finite Element Method
FLIP	Fluid Implicit Particle
FP	Forward perpendicular
FPM	Finite Pointset Method
FPSO	Floating Production Storage and Offloading

FSO	Floating Storage and Offloading
FSI	Fluid–Structure Interaction
FVM	Finite Volume Method
GFDM	Generalised Finite Difference Method
GMRES	Generalized Minimal Residual
GPU	Graphics Processing Unit
GUI	Graphical User Interface
GW	Green Water
HOS	High–Order Spectral
HPC	High Performance Computing
IBM	Immersed Boundary Method
IBVP	Initial Boundary Value Problem
JSON	JavaScript Object Notation
LES	Large Eddy Simulation
LS	Least Squares
MARIN	Maritime Research Institute Netherlands
MLS	Moving Least Squares
MOL	Method of Lines
MPI	Message Passing Interface
MPS	Moving Particles Semi–implicit
NSE	Navier–Stokes Equations
ODE	Ordinary Differential Equation
PBD	Position Based Dynamics
PBF	Position Based Fluids
PDE	Partial Differential Equation
PIC	Particle In Cell
PIV	Particle Image Velocimetry
PPE	Pressure Poisson Equation
PSE	Particle Strength Exchange
QMR	Quasi–Minimal Residual

QMRCGStab	Quasi–Minimal Residual – Conjugate Gradient Stabilised
RANS	Reynolds–Averaged Navier–Stokes
RBF	Radial Basis Function
SPH	Smoothed Particle Hydrodynamics
STL	Stereolithography
TFQMR	Transpose–Free Quasi–Minimal Residual
VOF	Volume of Fluid
V&V	Verification and Validation
WLS	Weighted Least Squares

1. Introduction

1.1. Problem of Water on Deck

Harsh environmental conditions comprise of waves, wind and current, which induce large ship motions. Consequently, encountering waves of high sea states can exceed the free-board of ships and other floating structures. Large relative motions between the vessel and water can result in violent water dynamics that either produce foam and spray causing little harm, or they can result in solid water that flows onto the deck of the vessel, which is known as "green water" or "green sea". Foam and spray, which is sometimes termed "white water", distract the crew on deck and degrade visibility from the bridge. However, large amount of momentum of the water on deck can be destructive to ship equipment, crew and ship itself. Green water has been considered as an important problem for the safety and operability of naval and merchant vessels, and other floating structures with an exposed deck. Temarel *et al.* [1] describe that if the wave breaks and overtops the structure, then the flow becomes multi–phased and chaotic. A large aerated region is formed in the flow in the vicinity of the structure while water runs up on to the structure. The timeline of breaking of a high wave that overtops the bow is shown in figure 1.1. Water on deck may build enough momentum to impact against a structure on the deck. Damages done by water impacts are usually not critical for large vessels, but may lead to loss of production time.

Significant shipping of water is in practice usually handled by changing the course and reducing the speed in order to avoid serious damages. Tan [2] reported that green water was the most important indicator for captains to change the speed and course of Dutch merchant ships to avoid serious damages. For example, large containerships sail with higher speeds, and their containers not covered by any means on deck are highly at risk to green water.

Buchner [3] quotes from the book 'The Battle of the Atlantic': "Their hulls whipped and shuddered in the huge Atlantic seas ... solid green water swept destructively along their decks ... For hour after hour this process repeated itself. Damage mounted, hull plates splitting, boats being smashed, men swept overboard and delicate anti-submarine devices put out of order". A couple of photographs showing historical naval ships in green water events may be seen in figure 1.2.

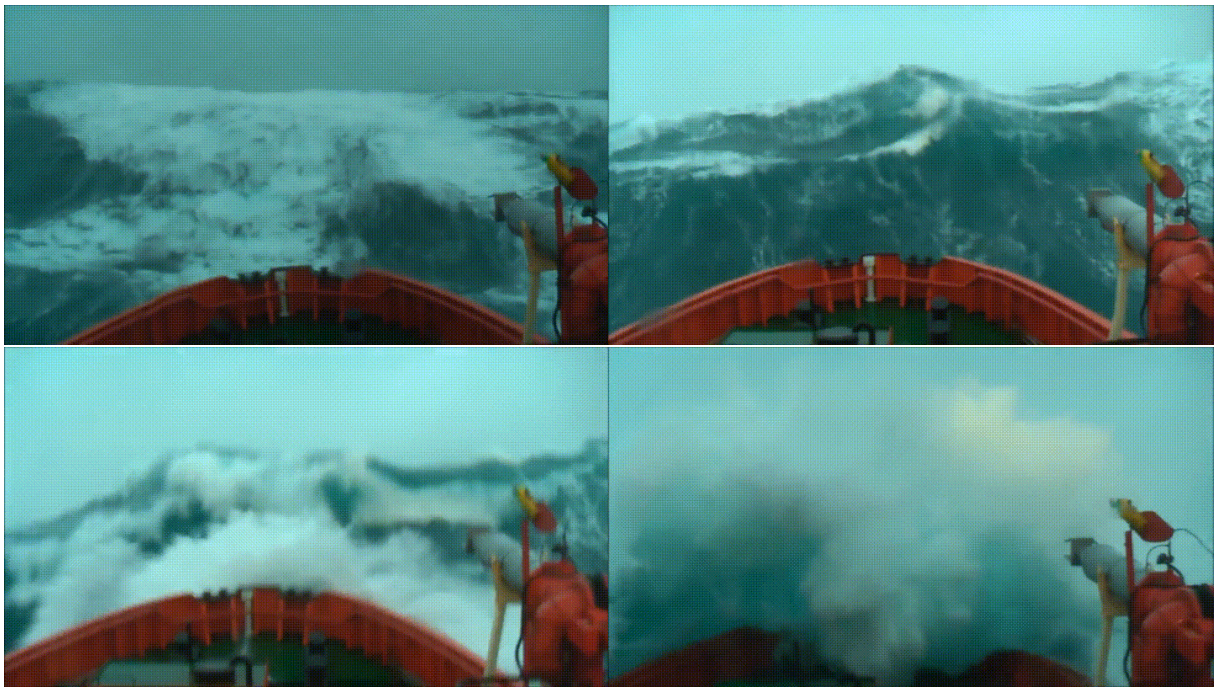

Figure 1.1.: A typical timeline of green water over the bow. The vessels heads into high waves that break just above the bow generating large amount of momentum.

MV Derbyshire was an ore-bulk-oil carrier that was lost on 9th of September 1980 with 44 fatalities. The investigation report [4] concluded that the occurrence of green water was the main culprit. Model tests conducted at MARIN evidenced that Derbyshire had accentuated pitching and considerable amplification of relative vertical motion in the problematic sea state, thus exposing the bow and forward hatch covers to heavy and repeated green water loading. It was concluded with reasonable confidence that the initiating cause of the loss was the destruction of some or all of the ventilators and air pipes located on the foredeck by sustained green water loading over many hours. Water was thereby able to enter vessel that developed a trim by the bow which had the effect of accentuating green water loading on the hatch cover as the sea conditions became more severe. Once that hatch cover collapsed, water rapidly filled the large ullage space above the cargo. It caused the vessel to go further down by the bow by another 3.7 metres within 5 to 16.5 minutes, with a final result of loss of the vessel.

Nowadays, it is common to produce oil from offshore vessels in order to eliminate the need of laying expensive long–distance pipelines. Floating Production Storage and Offloading (FPSO) vessels are permanently moored at a certain offshore position where they produce and process hydrocarbons and store oil. Compared to most of other ship types, FPSOs have a lot of sensitive equipment installed on the deck. Since they are moored by the bow, rotating and exposing the box to waves leads to green water events, which causes damage to its equipment and the superstructure. From 1995 to 2000, 17 green water incidents have been identified on 12 UK FPSOs and Floating Storage Units (FSUs) [5]. The vessel

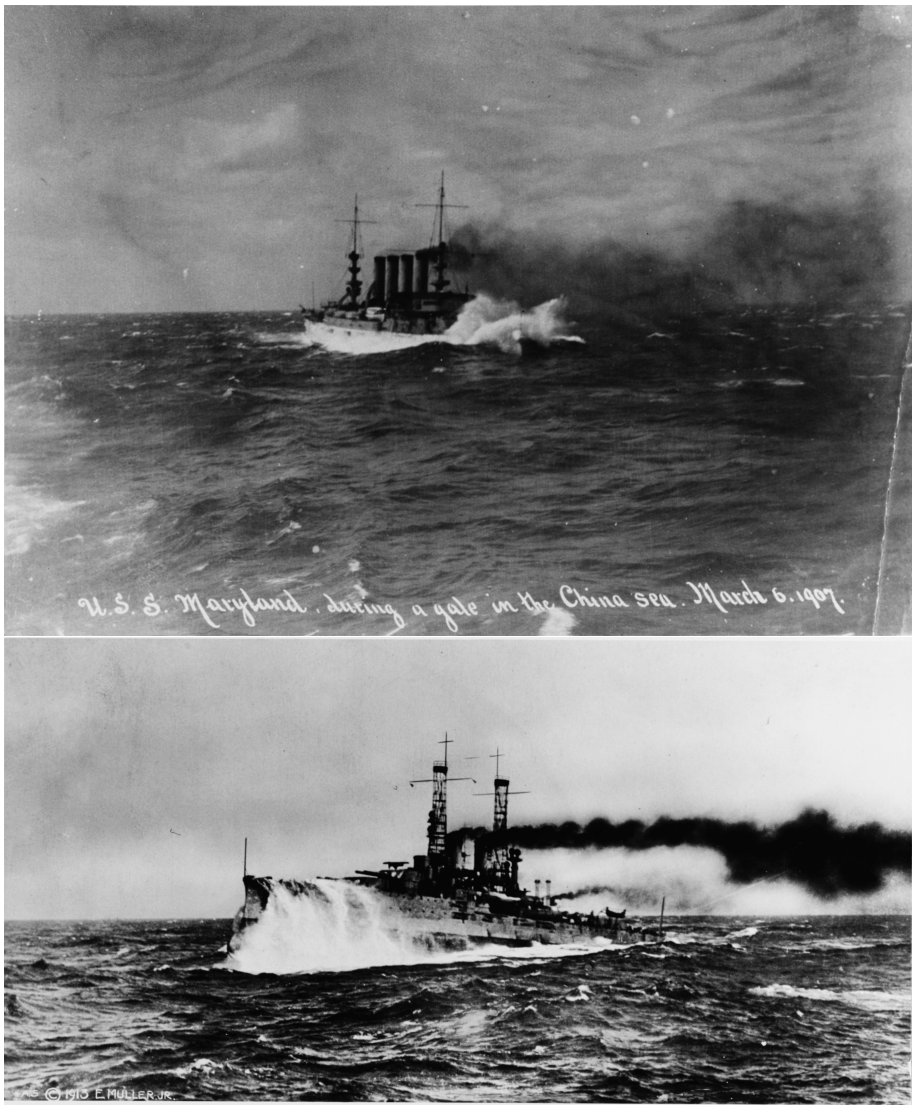

Figure 1.2.: Two historical photographs capturing green water events on naval ships. USS Maryland (top image) steaming through a gale in the South China Sea, 1907. USS Utah (bottom image) taking green water over the bow while returning from the Mediterranean Sea, 1913.

parts on FPSOs specifically designed for green water loading, were also damaged by green water [6]. In conclusion, FPSOs designed according to existing classification rules and regulations still suffer from green water loading damages.

It is also well known that green water can have strong impact on the stability of vessels, i.e. water on deck has also been the cause of casualties for smaller ships and boats. Water flowing on the deck changes loads distribution and consequently damages the stability of the vessel by reducing its metacentric height. In addition, lateral movement of the shipped water with free surface can amplify lateral motions of a ship, and therefore induce a catastrophe. A lot of research has been carried out on the effect of water-on-deck of fishing vessels after a number of capsize incidents.

Numerous damages caused by green water events have prompted researchers to solve the

problem, or at least to try to predict severe events. Green water is an example of a classical hydrodynamic problem with three-dimensional interaction between fluid and a structure. Due to complexity of the problem, classification rules and regulations have limitations in predicting green water loading. Early studies were based on experiments with different bow forms and head waves [7, 8], and statistical methods for the prediction of slamming and shipping of green water [9]. Design methods assessing green water that have been developing through more than half of century haven't managed to reliably predict or completely stop these events. Accidents have taken place, quoting green water and limitations in design standards being the culprits. The Derbyshire report [4] concludes that classification societies need to introduce improved design approval and survey procedures for new buildings.

In conclusion, green water is considered a serious threat to the safety and operability of vessels, which should be reliably predicted and properly assessed in the design stage.

1.2. Physics of Green Water Incidents

Besides an amount of the water that has shipped onto the deck, dynamics of both water and the vessel are responsible for inducing loads onto the deck, deck equipment and superstructures, too. Typically a green water incident can be separated into four stages:

I. the impact of the hull (usually the bow) and water (usually an incoming wave), which results in water sliding up the freeboard,

II. building of the water that enters the deck,

III. flowing of the water on the deck, forming a jet with complex structures,

IV. the impact of the flowing water and deck structures.

The listed stages are visually depicted in figure 1.3. Hirdaris *et al.* [10] argue that the green–water problem in a certain sense is a counterpart of the slamming problem, provided that the water column impinges the structure from above. However, the green water problem shares similar flow features with sloshing. Various components are responsible for the possibility of occurrence and harshness of green water incidents. The components are linked to one another, but their separated overview is given in the following text.

1.2.1. Relative Wave Motions and the Freeboard

Relative wave motions around the vessel hull are the main input to the green water problem, which is given by the following expression:

$$r_W(t) = \eta(t) - z(t), \tag{1.1}$$

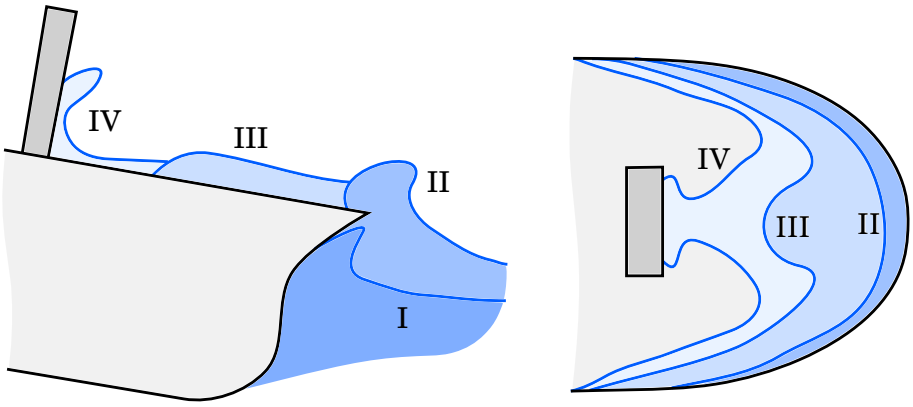

Figure 1.3.: The main stages of water wetting the deck of a ship defined in [11], shown schematically in profile (left image) and top view (right image).

where r_W is the local relative wave elevation, η is the incident wave elevation, and z is the local vertical vessel motion. The relationship is schematically drawn in figure 1.4. In the linear analysis, the incident wave elevation and local vertical vessel motion are respectively defined as:

$$\eta(t) = \frac{H_W}{2} \cos(\omega_W t - k x), \tag{1.2}$$

$$z(t) = A_z \cos(\omega_z t - \xi_z), \tag{1.3}$$

where ω_W is the wave frequency, k is the wave number, A_z is the amplitude of the vertical vessel motion, and ξ_z is the phase angle of the vertical vessel motion. In the linear analysis, if a relative wave elevation exceeds the freeboard level, the water will flow onto the deck. The freeboard exceedance height, h_E, is defined as:

$$h_E(t) = r_w(t) - h_{FB}, \tag{1.4}$$

where h_{FB} is the height of the freeboard. Since this is the basis of the green water prediction, the relative wave motions should be predicted accurately in order to reliably predict green water loading. It can be assumed that the initial shape of green water is sinusoidal with the height equal to the freeboard exceedance, but Buchner [12] has shown that the freeboard exceedance does not necessarily lead to the same value of water height on deck. Buchner and Voogt [13] noted that the relation between the water height on the deck and the freeboard exceedance is almost independent of the underwater hull shape, the wavelength, the current speed and the wave direction. As input to the green water problem the incoming waves can be an important source of nonlinearities in the relative wave motion result [14], and some authors take the nonlinearity in the waves as the main component in the prediction of the extreme relative wave motions, e.g. [15].

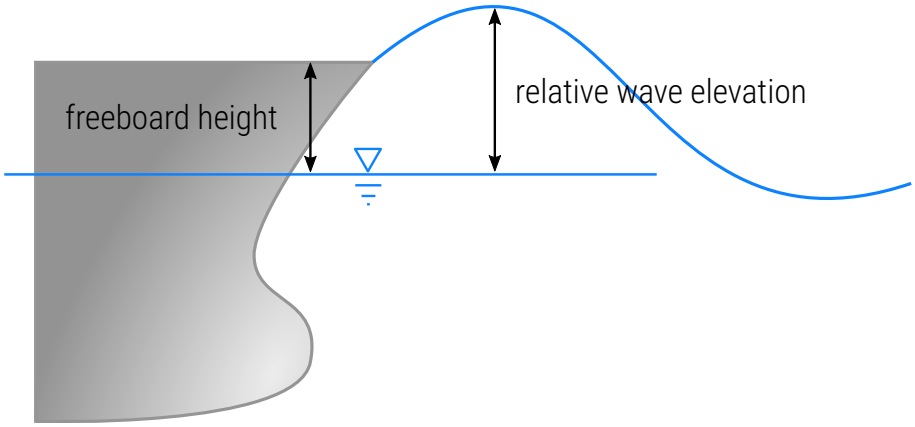

Figure 1.4.: Relative wave elevation compared to the vessel freeboard.

1.2.2. Vessel Motions

Vessel motions and the forward speed have additional influence on the development of green water dynamics [16]. For example, the vessel pitching increases relative vertical motion of the vessel quite substantially. The more the relative vertical motion, the greater will be the head of green water on the bow. For a ship with large forward speed, a very significant amplification of the consequent loads may occur. Linear and nonlinear potential flow methods form the basis for most green–water prediction tools. The main assumptions are that the fluid is assumed to be ideal, that the waves are sinusoidal, and that the waves and vessel motions are small, so the fluid is taken into account up to the initial waterline. Since the problem can be linearised, the frequency domain can be employed to obtain the relation between the ship motion and the incoming wave amplitude for each frequency. Whether these assumptions are justifiable, is still debatable [3].

1.2.3. Flare and Bow Shapes

The effect of the bow shape on the occurrence of green water has always been a point of discussion. It is certain that the flare in the bow region of a ship with its outward curvature of the hull above the waterline, like shown in figure 1.5, affects the relative motion and the diffraction of the incoming wave. The flare does not only push the water up, but also away from the bow. Also, the disturbance of the bow flare stops when the relative wave motion comes above the deck edge [3, 17]. This influences the resulting height of water on the deck and the dynamics of the shipped water on the deck. As an example, O'Dea and Walden [18] have shown how the increase of the flare on traditional ship types can reduce deck wetness. On the other hand, Lloyd [19] has shown how the increase of the flare also increases deck wetness on a frigate type of ship. Buchner and Voogt [13] have deduced linear relation between the freeboard exceedance and the resulting water height on deck by including factor that depends on the flare of the ship bow. They also note that the

effect of the bow flare increases with the distance from the fore perpendicular (FP). In conclusion, bow geometry and motion can decrease or increase relative wave motion and thus decrease or increase the amount of shipped water on deck. There are many different opinions of this complex problem, and consequently, it can be stated that specificity and influence of a single parameter such as flare cannot be analysed straightforwardly, but requires prediction methods that couple as much parameters as possible [17]. Due to the complexity of the problem, a bow shape that is very effective to keep the deck dry in one condition can be less ideal in other environmental conditions [3].

Figure 1.5.: Bow flare of USS Wisconsin at her berth in Norfolk, VA. Photograph taken by Steve Hersey.

1.2.4. Water Flow on the Deck

Vermeer [20] noted that there is resemblance between a usual case of green water flow onto the deck and the theoretical dam breaking problem, which is similar to a shallow water wave. At initial time, it is assumed that the column of liquid is released by immediate removal of a dam that holds the column. Therefore, the water flows in the empty region, as depicted in figure 1.3. The velocity of the flow is assumed proportional to the square–root of the height of the liquid before the dam breaks:

$$U = 2\sqrt{gh}, \tag{1.5}$$

where h is the height of the liquid column, and g is the gravitational constant. Instead of assuming the stage II as an initially static problem, the actual problem is much more complex, because the deck is moving, the height of the freeboard is varying in time, and the initial velocity in not zero, especially for steep waves that reach the deck [21]. Nevertheless, the theoretical dam break problem is useful for understanding the water

flow on the deck. As the wave breaks and overtops the structure, the flow becomes multi–phased and chaotic as a large aerated region is formed in the flow in the vicinity of the structure while water runs up on to the structure [1]. The water on the deck forms a high velocity water jet and violently impacts the structure like an impinging jet. In addition, the green water on the deck can have a significant effect on the ship motions, especially in shorter waves, due to its large moment arm with respect to the centre of gravity of the ship [3]. Generally, pressure on the deck caused by the water flow is influenced by the initial velocity and water column height (stage II), the acceleration of the deck, and the combination of the deck velocity and the rate of change of water height on the deck. Unfavourable combination of the listed conditions leads to significant pressures, which load the deck and form a powerful jet that impacts structures on the deck. For ships with zero speed, or very low speeds, the accelerations usually reach their maximum when the water reaches the deck (stage II).

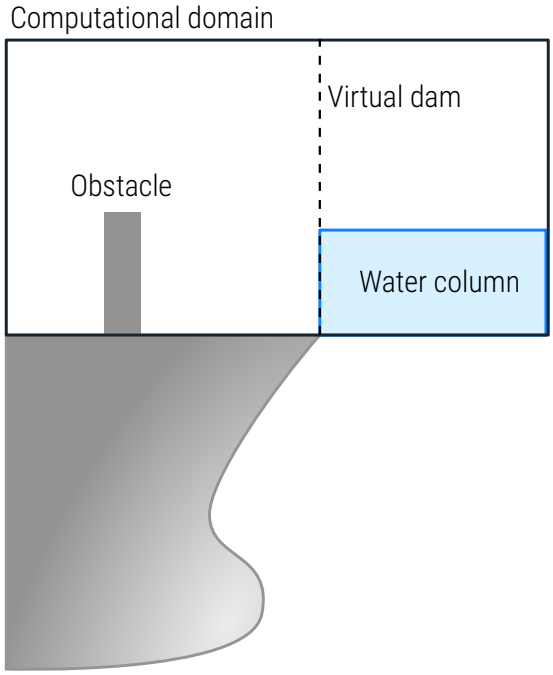

Figure 1.6.: Modelling green water flow as a dam breaking problem.

1.2.5. Water Impact on Structures

When water on the deck reaches a structure with high velocity, it results in a violent impact with significant impact loading, which is depicted as stage IV in figure 1.3. On the impact, the water flows up the structure and builds significant amount of water, which falls back on the deck. Besides the immediate loads at the impact, the loads from the water falling on the deck can also be significant. Therefore, pressure measurements in green water experiments are usually characterised by two pressure peaks: the first one

caused by the water impact on the structure, and the second one caused by the water falling down on the deck near the structure. Worst–case scenarios frequently include breaking waves, which induce significant loads. Snapshots from an in-house experiment of a breaking wave that wets the deck is depicted in figure 1.7. Besides a complex shape of the free surface, large spray, splashes and aerated regions are generated, which makes the problem hard to numerically simulate. Breaking waves can also develop on the deck and violently impact deck superstructures.

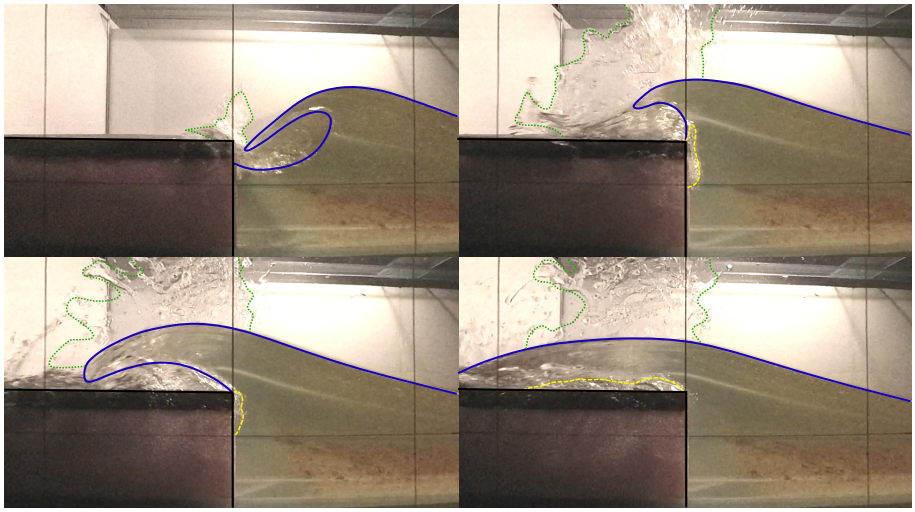

Figure 1.7.: Snapshots of an experiment in which a breaking wave wets the deck. Free surface, splashing and aerated regions are depicted by blue, green and yellow colours, respectively.

1.3. Reducing Green Water Loads

Breakwaters are usually used to reduce green water loads. Buchner [3] tried to qualitatively evaluate the efficiency of breakwaters protecting the deck by comparing V–shaped, vertical and vane–type breakwaters, drawn in figure 1.8. The author reported that a vane–type breakwater effectively reduced water that piles up in front of the breakwater, and that less water reached the deck structures. Some CFD studies on the performance of breakwaters were carried out by Pham and Varyani [22, 23]. The loads on breakwaters and protected structures were compared to find out the advantages and disadvantages of each type of breakwater. However, the authors did not present a validation with experimental data. They also added perforations, and concluded that the arrangement of holes on rectangular breakwaters does not have advantages.

Major classification societies do not give details on the efficient design of the breakwater, but only simplified formulas to predict loads without taking into account the shape of the obstruction. However, it is shown in practice that FPSOs have been benefiting from

vane–type breakwaters that are fitted along the ship sides. It must be noted that any obstruction on the deck restricts flow, which leads to the increase in pressures on the deck plating forward of the obstruction. Vane–type breakwaters are complex to construct, and require support from the deck. Simpler V–shaped breakwaters are often mounted since they are more cost–effective. Due to the lack of experimental data, non-validated numerical methods, and the lack of classification rules, there is significant amount of uncertainty in the design process of breakwaters. Besides breakwaters that physically stop water that has shipped on the deck, modifications of the ship bow may mitigate the shipping of water on the deck. As already noted in section 1.2.3, there exists no general conclusion on how the bow shape correlates with reducing the green water loads.

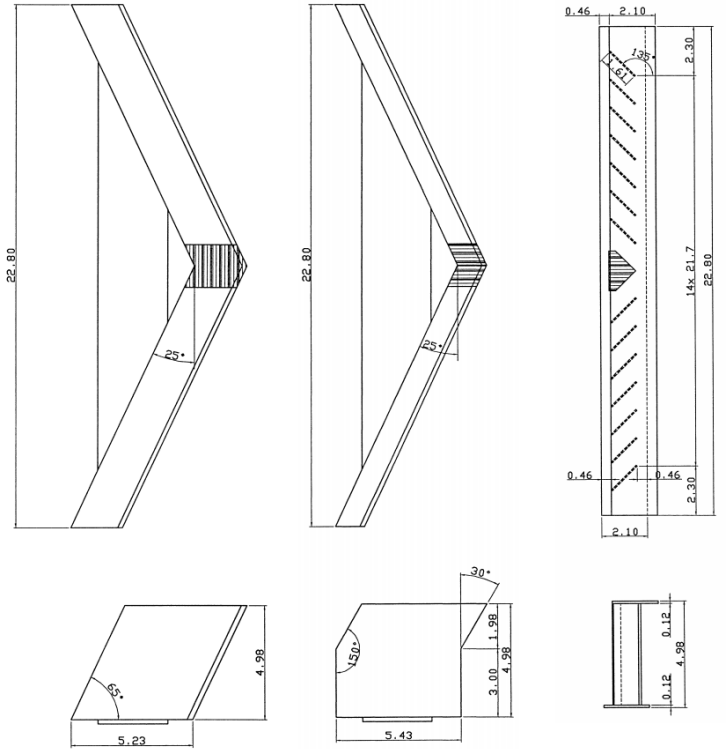

Figure 1.8.: Traditional V–type breakwater (left image), vertical wall breakwater (middle image) and vane type breakwater (right image) [3].

1.4. Past Research Overview

There are hundreds of scientific papers and research reports that deal with green water effects, i.e. the physics of water on deck and possible prediction methods. Some of the most notable and recent studies are listed in the following text.

1.4.1. Numerical Methods

Temarel *et al.* [1] conclude that the green water problem has been investigated experimentally and numerically, but with very limited success. The issues arise due to the fact that green water is very difficult to measure either under laboratory conditions or in the field, and very difficult to simulate numerically. The problem, due its fast moving, multiphase and highly turbulent nature, is not very amenable for accurate numerical simulation, although 3D models help to understand and simulate the mechanisms of green water impinging on deck. Computations of green–water loads with various wave headings is still a challenge. Qualitative benchmark studies are required to validate and improve numerical approaches.

Bellezi *et al.* [24, 25] analysed the green water phenomenon on three-dimensional models using the Moving Particles Semi–Implicit (MPS) method, which is a fully Lagrangian method for incompressible flow. They focused on the validation of the method comparing the numerical results with experimental results for green water on reduced scale models.

Buchner [12] showed how the impact of green water on FPSO design will become more important in the future, and based on a series of model tests he presented an orientation into the various aspects which play a role in the green water loads on FPSO units.

Buchner and Voogt [13] investigated the water shipping pattern and the pressure distribution on the deck of FPSO due to green water with varying bow flare angles. They showed that the amount of green water and the resulting impact loads on structures at the deck are not only functions of the freeboard exceedance level, but also of the bow flare angle of the FPSO.

Colicchio *et al.* [26] analysed the water shipping caused by head sea waves for a FPSO ship at rest with a 3D Domain Decomposition (DD) strategy, where a linear potential flow seakeeping analysis of the vessel is coupled with a local nonlinear rotational–flow investigation for the prediction of water-on-deck phenomena.

Silva and Rossi [27] tried to simplify the green–water loads determination by proposing a methodology to estimate these loads considering the water elevation above deck measured from experiments or numerical tools.

Drake [28] investigated instantaneous wave profiles associated with green water events for a stationary vessel heading into long-crested random waves.

Ersdal *et al.* [6] described five production ships in Norway and included their green water incidents. Furthermore, they described the status of the methods for evaluating the green water phenomenon as well as precautions taken to prevent further incidents.

Faltinsen *et al.* [29] studied green water in the bow region of a FPSO unit in head sea waves by employing a 2D method satisfying the exact nonlinear free–surface conditions

within potential–flow theory, as a step towards a fully 3D method. They concluded that the wave steepness of the incident waves causes important nonlinear effects, and that the hydroelasticity effects can be ignored. They also showed the importance of accounting for the coupled flow between the deck and outside the ship.

Gatin *et al.* [30] validated a FVM solver for green water simulations by employing Volume of Fluid (VOF) and Ghost Fluid Method (GFM) to discretise the free–surface boundary conditions at the interface. The solver was later used for calculating extreme green water loads upon a vertical–deck structure of an Ultra Large Container Ship (ULCS) in full scale.

Gómez–Gesteira *et al.* [31] analysed the phenomena within the framework of the Smoothed Particle Hydrodynamics (SPH) method, and noted how the flow in the wave crest is split into two, showing a different behaviour above and below the deck.

Gong *et al.* [32] established an engineering approach, named moving foredeck model, to simulate green water flow and its impact on deck structure for high speed vessels. The motion of a vessel was obtained with a potential flow solver, which was used as an input to a commercial CFD solver.

Greco *et al.* [33, 34] developed a 3D domain decomposition (DD) strategy to deal with violent wave–ship interactions involving water-on-deck and slamming occurrence, by coupling a linear potential–flow seakeeping solver with a Navier–Stokes solver. The coupling was applied for the case of a freely floating patrol ship in head regular waves and compared against experiments in terms of flow evolution, body motions, and pressure on the hull.

Greco *et al.* [35, 36] conducted a synergic 3D experimental and numerical investigation for wave–ship interactions involving the water-on-deck and slamming phenomena. A weakly nonlinear external solution for the wave–vessel interactions was combined with a 2D shallow–water approximation. They confirmed that the water shipping features are qualitatively and quantitatively affected by the parametric roll of the ship.

Hamoudi and Varyani [37] investigated the probability of green water occurrence by taking into account the threshold of the vertical relative motion exceeding the freeboard. They concluded that there is no direct relation between the velocities in the waves and the water velocity over the deck. The water velocities around the bow are heavily distributed by the presence of the bow.

Kendon *et al.* [38] considered results from a 2D model test setup and compared the measured vertical loads on the deck against two simple potential theory based methods and against results from a commercial CFD code. They concluded that for isolated impact events the simple potential flow based models, which do not consider the influence of one impact event on another, are adequate to predict vertical loads.

Kleefsman *et al.* [11] simulated green–water loads including vessel motions through a domain decomposition: far–field and ship motions were calculated by the potential theory and were used to simulate the local flow around the deck using a VOF method, improved by introducing a local height function.

Landrini *et al.* [39] studied free–surface bow flow around a fast and fine ship, with an emphasis on the generation and evolution of the breaking and splashing bow wave using the 2D+t theory. The Boundary Element Method (BEM) was coupled with the Smoothed Particle Hydrodynamics (SPH) method to simulate the jet evolution and splashing.

Le Touzé *et al.* [40] applied the SPH method to predict the fluid behaviour for two different dynamic flooding scenarios: the interaction between a vessel and travelling waves, and the transient flooding behaviour that occurs on a side collision between two vessels.

Lee *et al.* [41] studied the behaviour of green water impacting a fixed rectangular structure, and investigated the flow kinematics of a series of experiments and CFD simulations based on the Finite Volume Method (FVM).

Lu *et al.* [42] performed CFD simulations of wave overtopping 2D and 3D fixed decks, and a green water impact on the deck and deckhouse of a moving FPSO model. The introduced Finite Element Method (FEM) solver in arbitrary Lagrangian–Eulerian (ALE) frame was thus verified by comparison to experimental measurements of the each simulated case.

Nielsen and Mayer [43] numerically modelled green water loads on a moored FPSO exposed to head sea waves by employing a FVM CFD solver, and the results were compared with the experimental ones [36, 13].

O'Dea and Walden [18] examined the importance of nonlinearities in large-amplitude motions, and conducted a series of experiments with a model of a frigate in steep head waves. They pointed out the disadvantages of the linear theory and that a time–domain nonlinear solver should be used for such complex problems.

Ogawa *et al.* [16] conducted a series of model tests to develop a method for practical estimation of ship motion and green–water loads using a model of general cargo ship. It was concluded that the difference of ship type should be taken into account for the determination of rational criteria of green sea loads.

Östman *et al.* [44] presented numerical simulations of green water events and wave impacts on a FPSO using a commercial FVM solver. The computations showed that CFD tools can be used when studying the physics of green water. However, due to heavy time requirements, this type of CFD analysis is not yet a practical tool for parametric design studies and deck structure optimizations.

Pakozdi *et al.* [45] showed the feasibility of the SPH method to give more detailed forecast of the hydrodynamics on the deck than the simplified water-on-deck estimation.

Pham and Varyani [22] developed a model of dam breaking with initial velocity using CFD, with which they investigated the green water loads on a high–speed containership.

Qin *et al.* [46] built a 2D numerical wave flume in which a nonlinear freak wave is generated. An elastic deck was used to consider the fluid–structure interaction (FSI) during the green water event. The simulations showed that although the elasticity of the deck barely influenced the global evolution of the fluid motions, it affected the local fluid pressures significantly.

Ryu *et al.* [47, 48] applied Ritter's dam–break flow solution for the prediction of overtopping green water on an offshore structure.

Schønberg and Rainey [49] developed a tool for the calculation of water velocities on the deck of a ship as a result of green water incidents. The flow of water is modelled using a numerical method, which applies the potential flow theory and uses a desingularised boundary integral equation method combined with an implicit time–stepping procedure.

Shibata *et al.* [50] have proposed an incompressible variant of the Moving Particle Semi–implicit (MPS) method. They compared the estimated pressure with the experiments for a ship in head waves. The investigation has shown that there is still some lack of agreement in terms of both pressure and forces acting on the deck due to relevant oscillations in time.

Shibata *et al.* [51] have extended the MPS method for a three-dimensional ship motion under high wave-height conditions where shipping of water occurs. The nonlinear effect of shipping water was successfully simulated by the MPS method, although there quantitative differences between the calculated and experimental results still remained.

Silva *et al.* [52, 53] evaluated the green water on FPSOs subjected to beam and quartering irregular seas through a combination of ocean basin model tests and CFD simulations. The model test was performed with measurements of water elevations and loads during green water events. The identification of critical regions near the deck edge and the water on deck propagation are characterized.

Song *et al.* [54] investigated green water velocities and impact pressures caused by the impact of overtopping waves on a fixed deck structure in a large–scale, deep–water wave basin. The impact pressures on the structures were strongly affected by the changing front shape of the broken wave and the impulsiveness of the impinging wave that contains a considerable amount of air entrainment.

Sun *et al.* [55] numerically investigated the green water on the tumblehome hull, which is different from that of hulls with flare freeboard. The motions of the hull were calculated with the potential theory, and the dam–break flow model was used to calculate green water height and pressure distribution, which resulted in lower values than those obtained in the experiment.

Stansberg and Karlsen [15] presented results from model tests with an FPSO in irregular waves. The results showed that critical events often occur in steep and energetic nonlinear waves and that the ship pitch motion is also essential. Wave spectra with moderate wave periods may therefore be more critical than long waves.

Wang *et al.* [56] presented a frequency–domain numerical approach of green water prediction for FPSOs in irregular waves, including the effects of bilge keels, spread mooring system, and asymmetrically arranged risers. Analysis of the results indicated that the relative wave elevation at sides of the FPSO in oblique waves was strongly affected by bilge keel, mooring, and risers.

Warmowska [57] introduced a numerical model, based on shallow water flow, which describes water flow on the deck.

Xiao *et al.* [58] conducted an experimental study to investigate the wave run-up and green water of a FPSO in shallow water, considering non-collinear environments and different water depths. The results showed that wave run-up and green water along the broadside can be significant in oblique waves, especially at midship, and that the water depth has an important influence on wave run-up.

Yamasaki *et al.* [59] developed a Finite Difference Method (FDM) to predict the water impact pressure caused by green water phenomena, where the density function method was employed in the framework of a locally refined overlapping grid system.

Yilmaz *et al.* [60] carried out experiments with models of FPSOs, which showed that the flow of water over the deck resembled a suddenly released wall of water rather than a breaking wave. The developed semi-analytical method predicted a jet-like formation at the forefront of dam in the simulation of green–water flow on the deck.

Zhu *et al.* [61] introduced a dynamic mesh technique and numerically simulated 2D green water occurrence on a fixed FPSO model in head waves and an oscillating vessel in beam seas.

Zhu *et al.* [62] extended a commercial CFD code to model green water occurrence on floating structures. The motions of an FPSO were calculated by potential theory, and CFD tools were used to investigate the details of green water impacts.

1.4.2. Experimental Methods

Experiments reproducing green water incidents are still being carried out. The objective of experiments is to determine extreme loads or to validate numerical methods. Scaled experiments violate scaling laws with respect to the effects of air entrapment. Most of the experiments are done for fixed structures at zero speed (e.g. FPSOs) using simplified wave conditions (e.g. regular waves or one focused wave).

Lloyd [19] conducted experiments to examine the deck wetness process and to determine the effect of bow form above the waterline on deck wetness in head seas, with a systematic series of bow forms with varying flare and overhang. The author found that the freeboard bow form had remarkably little effect on relative motions or deck wetness and the greatest incidence of deck wetness occurred with one of the extremely flared forms.

Cox and Ortega [63] conducted a small–scale laboratory experiment to quantify a transient wave overtopping a horizontal deck, fixed above the free surface. Free surface and velocity measurements were made with and without the deck structure to quantify the effect of the deck on the wave kinematics.

Fundamental experiments were carried out by Ryu *et al.* [47], in a wave flume with a focused wave overtopping a fixed structure. The authors have developed Bubble Imaging Velocimetry (BIV) technique to measure fluid velocities. The experiments correlate to analytical dam break models if the initial water height, i.e. the front velocity of the analytical model, is properly tuned [48].

Ariyarathne *et al.* [64] conducted experiments to serve as validation material for numerical codes. They generated an extreme event on a simplified wedge–shaped model of an FPSO, rigidly connected to the bottom of the tank. They have investigated green water impact pressures due to plunging breaking waves impinging on a simplified, 3D model structure [65]. Impact pressure was found to be quite different between the wave conditions even though the incoming waves are essentially identical.

Ryu and Chang [66] measured the water velocity fields of a plunging wave breaker at the front face of the deck and bubbly flow after overtopping of the deck using Particle Image Velocimetry (PIV) and Bubble Image Velocimetry (BIV) techniques.

Varyani and Pham [23] experimentally investigated the employment of whaleback forecastle, studying its effectiveness and adequacy as an option to reduce green water loads on high–speed container vessels.

Lee *et al.* [67] analysed the experimental results for three different FPSO bow shapes in regular head waves, and compared them to each other. They used a fixed model with a vertical bow, an inclined bow and a rounded bow to measure the pressures on the deck. Based on the results, a database for computational fluid dynamics code validation was built, and some design considerations were proposed.

Guo *et al.* [68] conducted model experiments for an FPSO in shallow water exposed to wind, waves and current. The authors used the data to validate their CFD calculations, without sharing the details of the experiments for the sake of reproduction by other researchers.

Table 1.1.: Comparison of various advantages and disadvantages of numerical simulations and experiments.

Category	Physical experiment	Numerical simulation
Results	Flow measurements	Flow prediction
Readings	Equipment–limited number of points	All quantities at any time/space
Price	Relatively expensive	Relatively cheap
Time	Slow process	Relatively quick
Scaling	Smaller models	Any
Setup	Limited range of operating conditions	Any operating conditions
Repeatable	Mostly	Yes
Safe	Not all	Yes
Error sources	Measurement errors, flow disturbances, etc.	Discretisation, numerical method, etc.

1.5. Numerical Hydrodynamics

Fluid flows are described by PDEs, which represent laws for the conservation of mass, momentum, and energy. CFD provides qualitative and quantitative prediction of fluid flows through software tools: solvers, preprocessing and postprocessing tools. Various numerical methods, i.e. discretisation and solution techniques, are used in order to solve the problem described by relevant PDEs. CFD enables scientists and engineers to perform numerical simulations on computers. As a rule, CFD still cannot completely replace the real experiment and its measurements, but the amount of experimentation and the overall cost can be significantly reduced. The amount of trustworthiness of the results of a CFD simulation depends on the level of uncertainty and on the accumulative effect of introduced errors. Uncertainty can come from the lack of comprehension of the phenomena, e.g. turbulence modelling. Errors can emerge from various sources, such as physical modelling uncertainty or introduced simplifications or spatial and temporal discretisation errors. In addition, the implementations may enforce low convergence criterions, add finite–precision errors while doing arithmetic operations or programming “bugs”. Lastly, the user may introduce errors while forming the solver input. Table 1.1 compares well known aspects of numerical simulations and experiments.

1.5.1. Mesh–Based Methods

Mesh–based methods for solving Partial Differential Equations (PDEs) have achieved remarkable success and they have been successfully applied in various fields of engineering. They divide a continuum domain into discrete subdivisions. A set of subdivisions is usually called a mesh or grid, which requires connectivity information based on a topological map.

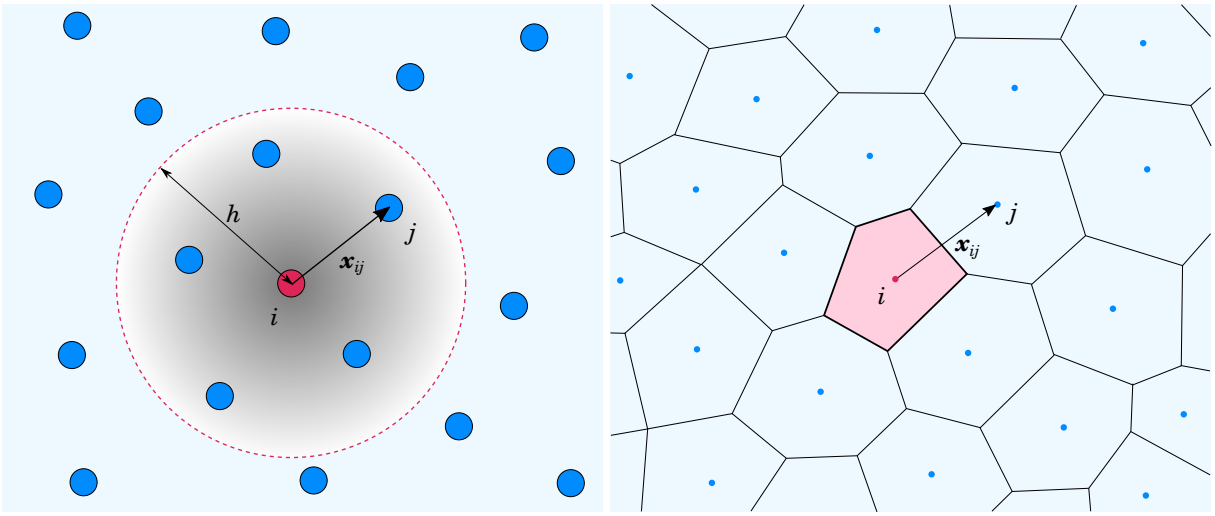

Figure 1.9.: Meshless or point–cloud description of a fluid continuum compared to the polygonal/polyhedral mesh counterpart.

An example of a mesh part is shown in figure 1.9.

Creating topologically correct and geometrically conforming mesh, which needs to be appropriate for an investigated problem, is time–consuming work and often requires user interventions within mesh generation steps. In addition, a simulation that requires large mesh deformations is difficult to maintain in geometrically complex problems, e.g. fluid–structure interaction. Deformations require cautious node movements or re-meshing of deforming areas to avoid mesh tangling and loss of mesh regularity.

Popular classes of mesh–based methods are listed in the following text, and their characteristics are briefly described.

1.5.1.1. Boundary Element Method

Boundary Element Methods (BEMs) are usually used where simplifications of the potential–flow theory can be employed, i.e. for inviscid non-rotational flows. The integrals over the whole fluid domain can be transformed to integrals over the boundaries of the fluid domain. This 3D-to-2D transformation simplifies grid generation and often accelerates computations, even though the problem–matrix is dense. Practical applications for potential flows around ships that deal with wave problems use BEM, which is often termed the panel method. In the panel method, the wetted surface of the hull and the surrounding free surface is divided into discrete elements, quadrilaterals and triangles, called panels. Each of these elements fulfils the Laplace equation. Indirect methods determine the element strengths so that at the collocation points (usually centres of the panels) a linear boundary condition of zero–normal velocity is fulfilled. This involves finding the solution to a dense system of linear equations with the source strengths as unknowns. The required velocities are computed in a second step, hence it is an “indirect” method. Direct

methods determine the potential directly. They are less suited for boundary conditions involving higher derivatives of the potential, but yield higher accuracy for lifting flows. Most commercially used codes for ship flows are based on indirect methods, where the pressure field is obtained through Bernoulli's equation.

1.5.1.2. Finite Difference Method

Finite Difference Methods (FDMs) were one of the first class of methods that was developed to solve diverse initial-boundary-value problems (IBVPs). FDMs divide the whole fluid domain into discrete nodes, almost always forming a regular mesh. Starting with the governing equations in partial–differential form, the differential terms are replaced by finite differences (FDs) via a truncated Taylor series expansion or polynomial fitting to the first or second order in terms of nodal values at each grid point. A single algebraic equation at each node represents some flow characteristic at that location by a specific number of neighbouring locations, which are unknowns. Discretisation errors can lead to a violation of conservation of mass or momentum, i.e. during the simulation the amount of fluid might continuously diminish. Due to the need of a regular mesh, it is difficult to automatically discretise boundary conditions, especially in the case of complex–shaped domains. On the other hand, the FDM is conceptually simple, easy to implement, and effective on structured grids.

1.5.1.3. Finite Volume Method

Finite Volume Methods (FVMs) use a numerical grid defined by a finite number of control volumes, rather than the computational nodes, as in the FDM. Connected control volumes fill the fluid domain usually in a complex manner, defining an unstructured mesh. Frequently the flow variables are collocated at computational nodes that can be placed at the centre of control volumes. FVMs also employ FDs for the temporal discretisation. However, spatially they are based on integrated equations for mass and momentum conservation over the individual cell before variables are approximated by values at the cell centres. Conservativeness is ensured by applying the Gauss divergence theorem, i.e. mass and momentum are conserved because errors at the exit face of a cell cancel with errors at the entry face of the neighbour cell. Most Reynolds–averaged Navier–Stokes (RANS) solvers nowadays are based on the FVM that work with unstructured body–conforming meshes. An effective alternative to dealing with dynamic conforming meshes are mesh methods that capture the geometry of the free boundary. Examples include methods based on the adaptive grid refinement of Quadtree and Octree grids, for which re-meshing is straightforward and efficient [69]. Such methods have satisfactory parallel scalability and can be extended to perform massively parallel computations [70].

1.5.1.4. Finite Element Method

The Finite Element Method (FEM) is similar to the FVM, but the domain is subdivided into discrete volumes called finite elements. In the FEM, a function that describes variations of the flow variable, known as the shape function, is substituted in governing equations obtained by multiplying a weighting function before the integration. The method is robust for variable grid refinements. FEM dominate structural analysis, but is also applicable to fluids. For ship hydrodynamics they play only a minor role. As a rule of thumb, FEMs for CFD generally require more memory and have slower solution times than FVMs, although discontinuous Galerkin method receives some attention in the CFD community. Finally, while the conservation is natural in the FVM, the FEM requires special treatments to ensure a conservative solution.

1.5.1.5. Immersed Boundary Method

Over the last few decades, it has become popular to extend FDMs and FVMs to handle immersed boundaries. Otherwise stated, Immersed Boundary Methods (IBMs) are formed so that they can handle complex geometries that are simply overlaid or "immersed" in the background grid. This enables the flow to be calculated through the regular grid where the geometry is not intersecting it, but the spatial derivatives around cells that are intersected by the geometry must be carefully discretised. To enforce the desired solid boundary, a body force is introduced in the momentum equations at the desired points in the fluid domain, without the necessity of performing the mapping procedures. As a result, generation of grids is greatly simplified and bodies immersed into the grid are free to move. For these reasons, it has become attractive to test IBMs on problems of ship hydrodynamics.

1.5.2. Meshless Methods

A wide variety of meshless methods have been proposed over the years and are currently under active development [71]. One of the main motivations is the recent advances in data–parallel algorithms and corresponding parallel–execution hardware, and the ease of applying such algorithms to methods that are based on compact support. The concept of meshless points employing spherical compact support is shown in figure 1.9 and compared with the equivalent mesh. During recent years, meshless methods have emerged as a class of effective numerical methods which are capable of avoiding the difficulties encountered in conventional computational mesh–based methods.

A meshless or mesh-free method can be used as a general approach for the numerical solution of problems in continuum mechanics, such as the simulation of fluid flows. The

medium is numerically represented by a finite set of points, which may be seen either as particles (discrete chunks of the medium) or as interpolation points (discrete nodes used to describe continuous fields in space). In any case, each point is endowed with the relevant local properties of the medium such as density, velocity, pressure, and temperature. The points can move with the medium, as in the Lagrangian approach to fluid dynamics, or they may be fixed in space while the medium flows through them, as in the Eulerian approach. In the Eulerian approach, the points are fixed in space, but new points may be added where there is need for increased accuracy. A schematic comparison of the Eulerian approach and the Lagrangian approach is shown in figure 1.10. Alternatively, a mixed Lagrangian–Eulerian approach may also be used. Due to the freedom of movement without mesh, meshless methods are ideal to be described by the Lagrangian approach. From figure 1.10 it can be noticed that the Lagrangian approach is natural as compared to the Eulerian, in which the observers are fixated to specific locations. For this reason, Eulerian approaches require special care when calculating the spatial derivatives. Nonlinear convective derivative (Latin *convectio* is "act of carrying") must be added to the partial derivative, $\mathrm{D}/\mathrm{D}t = \partial/\partial t + \boldsymbol{u}\cdot\nabla$, which means that the physical quantity is carried by the velocity field $\boldsymbol{u}(\boldsymbol{x}, t)$. More than three decades of research have shown that the simplicity of the convection term does not extend to its discretisation.

Meshless Lagrangian methods are free of the term $\boldsymbol{u}\cdot\nabla$ due to the natural observation of the movement, and thus easily adapt to domains with complex and/or time–evolving geometries and moving phase boundaries without the software complexity that would be required to handle those features with topological data structures. They can be useful for nonlinear problems involving viscous fluids, heat and mass transfer, linear and nonlinear elastic or plastic deformations, etc. Meshless methods have been developing under two branches of formulations: methods based on the weak form of PDEs, and methods based on the strong form of PDEs. Classification and overview of mesh-free methods is given in [72, 73]. Brief descriptions of some of the most used and researched meshless methods are given in the following text.

1.5.2.1. Smoothed Particle Hydrodynamics

The most popular, and nowadays widely used meshless method is the Smoothed Particle Hydrodynamics (SPH) method, which represents quantities as discrete particles and uses a kernel function to smooth their volumetric contributions. The method was originally introduced to solve problems in astrophysics without boundaries [74, 75]. The method has been further extended to solve varieties of problems like compressible inviscid flows, incompressible inviscid flows, multiphase flows. A weakly compressible SPH (WCSPH) approach was introduced by Monaghan [76], who used it to simulate free–surface flows [77]. The incompressible limit is obtained by choosing a very large value of the speed of

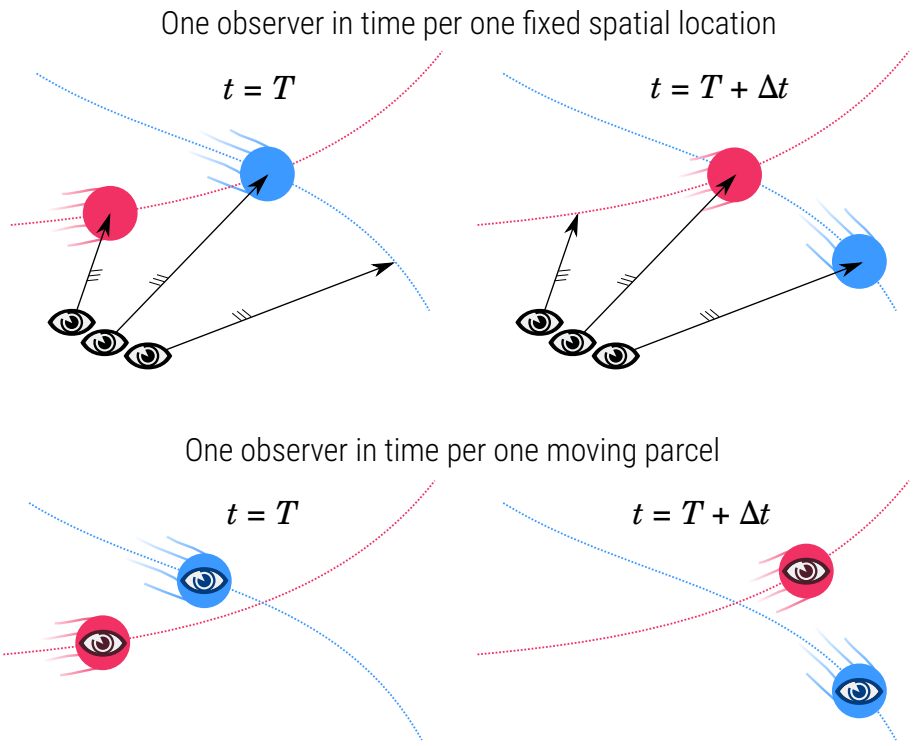

Figure 1.10.: Comparison of the Eulerian viewpoint where the quantities are observed at fixed points in space (top image), and the Lagrangian viewpoint where the points are carrying physical quantities (bottom image).

sound in the equation of state, such that the Mach number becomes small. However, a large value of the speed of sound restricts the time step to be very small. SPH methods have a number of problems that require special fixes, which sometimes come at great cost, but do not eliminate all low–order inconsistencies [73]. The other principal difficulty of the SPH method is imposing boundary conditions. Other inherent characteristics of the method are tensile stability, energy conservation, and lack of interpolation consistency.

1.5.2.2. Moving Particle Semi–Implicit

The Moving Particle Semi–Implicit (MPS) method is similar to the incompressible SPH method, because both methods provide approximations to the strong form of PDEs on the basis of integral interpolants. However, the MPS method applies simplified differential operator models solely based on a local weighted averaging process without taking the gradient of a kernel function. The standard MPS is formulated assuming a set of particles located inside obstacles and along their boundary. Tamai and Koshizuka [78] improved the MPS method with the Least Squares (LS) approach for high–order accurate spatial discretisation with consistent time integration and generalized treatment of boundary conditions.

1.5.2.3. Generalized Finite Difference Method

The Generalized Finite Difference Method (GFDM), sometimes referred to as the meshless FDM, evolved from the classical FDM. It can be applied over general or irregular clouds of points. The basis of the GFDM was published by Jensen [79], where Taylor's series expansion on six–node stars was used to derive the FD formulae approximating derivatives of up to the second order. Benito *et al.* [80] reported that the solution of the generalized FD method depends on the number of nodes in the cloud, the relative coordinates of the nodes with respect to the star node, and on the weight function employed. The GFD methods have been used in many engineering applications and in papers where irregular geometries and free–moving boundaries are involved. In the GFDM, the MLS approach is adopted to derive the expressions of the spatial derivatives and approximation of the primary variable, which are expressed as linear combination of nearby function values. To enforce the satisfactions of governing equations at every interior node and boundary conditions at every boundary nodes via a collocation approach, a sparse system of algebraic equations is solved. Then the numerical solutions are acquired by solving the resultant system of algebraic equations. The GFDM may achieve high order of accuracy, depending on the order of the MLS. On the other hand, for unsteady problems MLS operations need to be redone each time step, which yields poor efficiency of the method for such problems.

1.5.2.4. Finite Pointset Method

The Finite Pointset Method (FPM) is a particle method, fully Lagrangian and mesh-free, in which a fluid is replaced by a finite number of particles (a pointset). Boundaries are also approximated by a finite number of boundary particles and boundary conditions are prescribed on them. The robustness of this method is shown by the simulation results in the field of airbag deployment in car industry, since the boundary of the airbag changes very rapidly in time and takes a quite complicated shape [81]. Initially, the FPM simulations of incompressible flows were performed as the limit of the compressible Navier–Stokes equations with the quasi compressible equation of state, and later the method was extended to employ the Chorin's projection method [82] in a local iteration procedure [83]. The Poisson problem in the projection method was adopted in the MLS approximation procedure with the condition that the Poisson equation and the boundary condition must be satisfied on each particle [84]. In conclusion, the FPM is a variant of GFDM as the strong solution of the considered problem is determined by direct approximation of the differential operators.

1.5.3. Hybrid Methods

Of course, the listed approaches have different strengths and weaknesses. The Lagrangian approach adequately handles the advection, but has problems with the evaluation of the pressure and incompressibility constraint. On the other hand, the Eulerian–grid approach is suitable for solving the pressure and incompressibility constraint, but the approach encounter problems with the advection. Hybrid methods were developed in order to minimise the issues on advection and pressure arising when simulating incompressible flows.

1.5.3.1. Particle In Cell

The Particle In Cell (PIC) concept represents the earliest hybrid approach where a grid evaluates the particles that represent the motion of a fluid and its properties [85]. The advection step is handled using particles and the rest of the simulation steps, i.e. the diffusion, external forces, pressure projection, and boundary conditions, are computed on a grid. One of the main drawbacks of the PIC method is that it suffers for severe numerical diffusion due to the averaging and interpolation of velocities when transferring them from the grid onto the particles. Basically, particle characteristics are diffused, because their velocities are overwritten by a fluid flow computed on the grid. A fluid simulated with the PIC method will, therefore, appear more viscous than it should be in reality.

1.5.3.2. Fluid Implicit Particle

Brackbill and Ruppel [86] solved the main problem of the PIC method by instead of interpolating the newly calculated velocities to the particles from the grid and replacing the velocities, they interpolated the change in velocity and added it to the already existing particle velocities. Except for the velocity update step, the Fluid Implicit Particle (FLIP) and PIC methods are basically identical. However, resulting fluid simulations with the FLIP method is less viscous than with the PIC method, and therefore is suited for simulation of water. FLIP method that simulate fluids with the free surface experience undesirable noise on the free surface, which is often assessed by linearly combining the PIC and FLIP methods.

1.6. The Research

1.6.1. Problem Definition and Objectives

The problem described in section §1.1 arising due to complex physics described in section §1.2 is a significant problem for vessels and offshore structures. The listed experimental and numerical methods give limited insight in determining the occurrence and magnitude of green water loads. There are no validated tools that are both computationally efficient and reliable for evaluating vessels and offshore structures for the risks of green water loads. Consequently, classification societies seek for new procedures that may approve and improve designs of new buildings.

The aim of this thesis is to introduce a new meshless numerical methodology for single–phase (water without air) flow modelling in marine hydrodynamics, which can be coupled with some other mesh-based solver for the purpose of simulating a ship on waves and corresponding green water incidents. The methodology should be: accurate and validated, possible to couple with other solvers, computationally efficient, and easy to use.

The research is based on the following hypotheses:

1. The Lagrangian description of flow naturally describes the advection of fluid with the free surface, as compared to the Eulerian description of flow.
2. It is possible to solve the nonlinear problem of green water with high–fidelity by solving Navier–Stokes equations based on the Lagrangian description of flow.
3. The global domain can be decomposed to improve computational efficiency, by coupling the Lagrangian nonlinear region with some simpler solver for the outside region.

1.6.2. Rationale

The high–hierarchical level of the mind map, showing rationale for establishing the method proposed in this thesis, is rendered in figure 1.11. While the main arguments for the methodology are given in the following text, assertion details of the each methodology item/step are given in the remainder of the thesis.

1.6.2.1. Requirements

The nonlinearities in green water incidents can be related to the effect of water-on-deck on the ship motions, the effect of the hull shape above water, and the effect of nonlinearities in the waves. Therefore, high-fidelity prediction of the flow in the vicinity of the vessel is needed, which accounts for all the nonlinearities of the problem, and for a complex

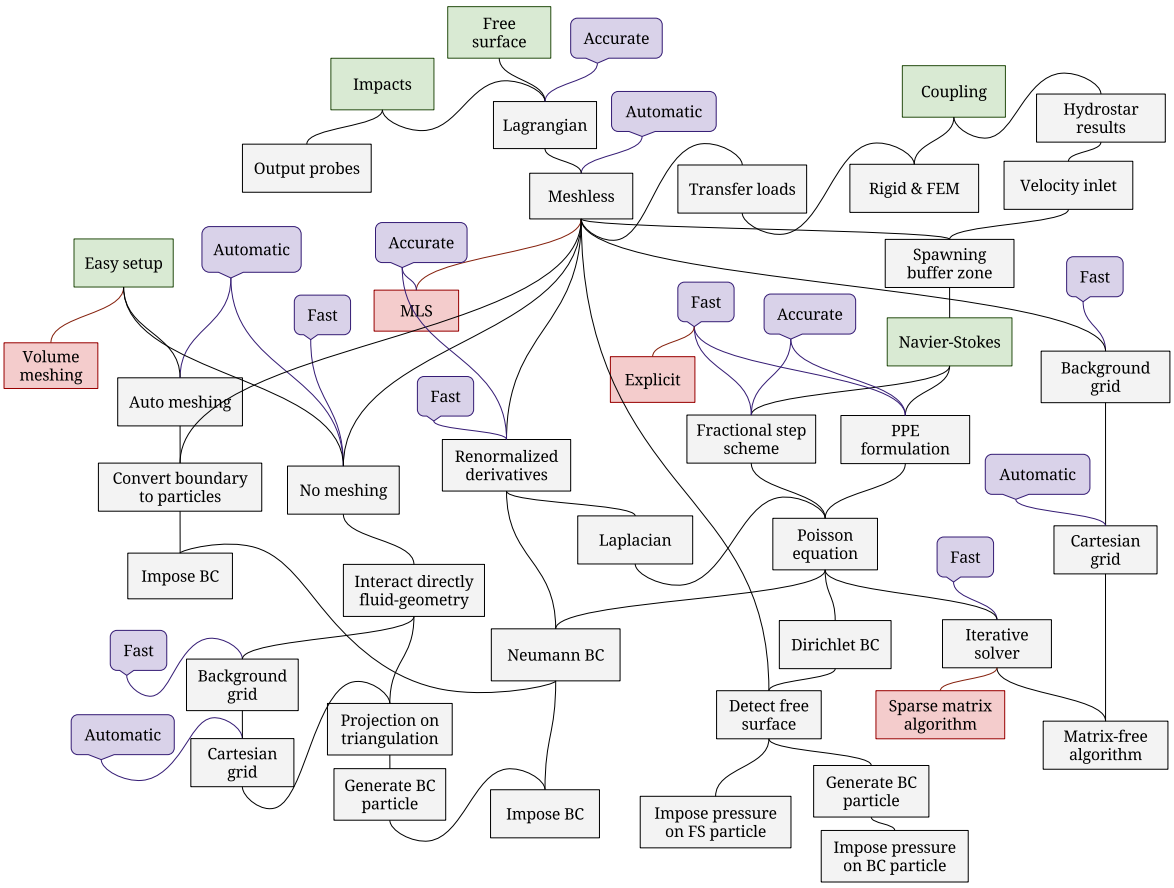

Figure 1.11.: High-hierarchical level of the mind map, showing rationale for establishing the novel method. Requirements are colourised green, deal–breakers red, and favourable properties blue.

nature of violent impacts of fluid onto the structure. When steep waves and other factors contribute to forming a complex shape of the free surface that surpasses the freeboard, it is important to further simulate the flow as accurate as possible. In conclusion, even if it is assumed that the potential flow theory can be used to simulate ship motions on waves, an Euler or Navier–Stokes equations solver of high–fidelity that can capture the nonlinearities must be used both in the *vicinity* of the vessel and *on the deck*. The solver should simulate complex free surface shapes and handle their fragmentation and reconnection before, during, and after the violent impact of water on deck structures. These requirements termed “Navier–Stokes”, “free surface” and “impacts”, are colourised green in figure 1.11. Other two requirements include coupling of multiple domains, i.e. the speeding up of the simulations by a potential–flow solver for the ship motions and far–field waves, and the requirement that a simulation should be easily set-up so that an engineer using the procedure does not waste enormous amount of time on pre-processing tasks, which is usually the case when using mesh–based flow solvers.

1.6.2.2. Discretisation

A solution to the requirements of predicting violent fluid–structure impacts with complex free surface fragmentation, is to employ a meshless and Lagrangian method for the spatial discretisation and representation of water flow. Lagrangian advection has various advantages over the Euler representation of water flow. The main advantage is that it can handle fluid domains, whose boundaries change rapidly without any additional computational effort. As it was schematically depicted in section 1.5.2, meshless methods intrinsically do not require mesh generation with topology in the pre-processing stage, which is the most substantial task when setting up a numerical simulation for a mesh–based solver. In the meshless context, a simpler and automatic generation of the continuum is used, i.e. a fluid domain is simply filled with points bounded by the input solid geometry, which makes a simulation fast to set-up.

1.6.2.3. Numerics

Commonly used meshless methods presented in section 1.5.2 have different advantages and disadvantages. Most promising methods concerning the solution accuracy are based on MLS, which can be extended to reach arbitrary order of accuracy. However, they require high computational power, and therefore, they are not suitable for transient problems with rapidly changing geometry. A compromise between fast and relatively inaccurate SPH methods, and slow and accurate MLS–based methods, are fast second–order accurate Weighted Least Squares (WLS) methods, used in this thesis. The disadvantage of the WLS methods is that the second derivatives are not directly obtained from their formulation. Therefore, this thesis deals with finding a solution to this problem in order to obtain high–fidelity solutions to Navier–Stokes equations, which depend on the robustness and accuracy of the Laplace operator.

1.6.2.4. Boundaries

The Lagrangian nature of the newly introduced method allows for direct interaction between the fluid and geometry, i.e. the geometry can simply be moved in time, and the fluid adjusts to moving boundaries by respecting the imposed boundary conditions. Moreover, geometry can be transformed into discrete points at the start of the simulation, or the points close to walls can be projected at each time step to generate boundary points. Consequently, the only input are the boundary surfaces defining the domain and bodies of the simulation, without the need of any kind of mesh preparation. The combination of the Lagrangian approach for the fluid advection and arbitrarily moving (or even deforming) boundaries is suitable for the simulation of green water events, where the ship hull moves along with complex flows around the hull and on the deck.

1.6.2.5. Domain Decomposition

A rigid–body solver and a Navier–Stokes solver may be coupled to simulate waves and vessel motions in waves in time, along with water flow on the deck. The unified simulation is alluring, but is not feasible in everyday engineering projects due to time–cost, i.e. the current status of hardware. State-of-the-art seakeeping software based on the potential flow theory are validated, and their constraints are well known. The solvers based on the potential flow theory swiftly simulate characteristics of vessel motions under different environmental conditions. Therefore, they are usually used to predict worst case scenarios and their probability. In order to speed up the simulation of a green water incident, the solver introduced in this thesis allows for coupling with another solver though the domain decomposition approach. In other words, the boundaries of the fluid domain communicate with the other solver to transfer the flow information between each other. The most feasible approach for the evaluation of a green water incident is to calculate ship motions with a potential flow solver and to impose those motions and inlet conditions to the Navier–Stokes solver with a domain built just around the hull where nonlinearities occur. In conclusion, the reduced–size domain of the Navier–Stokes solver coupled with some fast external–domain solver allows the engineer to quickly simulate necessary scenarios, in lieu of making unified simulations in a large domain using only a Navier–Stokes solver.

1.6.3. Contributions

Based on the rationale argumented in section 1.6.2, new contributions to the field of numerical hydrodynamics are made. The research resulted in a new meshless and Lagrangian numerical methodology, which can be used to simulate green water phenomena on dynamically floating vessels and to obtain the corresponding loads on the deck and its structures. The main contributions of this thesis to the field of CFD and marine hydrodynamics are summarised as follows.

Lagrangian velocity–pressure formulation of the NSE is used to evolve fluid flow. Contrary to vast majority of implicit meshless methods that use projection schemes, proposed numerical methodology is built on top of the velocity–pressure formulation. Without intermediate sub-steps, it requires solving one Poisson equation per time step, irrespective of the dimensionality of the problem. This is, performance-wise, the best that one can achieve for incompressible flows.

The Poisson equation is solved in finite difference (strong) form by employing the novel discrete Laplacian. The main ingredient for adequately solving a Poisson equation is the discrete form of the Laplacian. In this thesis, the Laplacian is defined by extending the renormalisation technique, i.e. a least–squares technique used for the gradient. The introduced Laplacian is efficient, reproduces the central finite difference

results for regular arrangements, and has a linear convergence rate even for extremely irregular point arrangements.

Boundaries are represented by directly importing geometry models. Lagrangian meshless methods are well known due to the fact that they don't require volumetric meshes, but most of the methods prepare boundaries by converting them to (multiple layers of) points. The introduced method goes a step further and directly interacts with boundaries by projecting near fluid points onto them to generate boundary conditions each time step. It is shown that skipping the step of preparing boundary points benefits accuracy through better boundary conditions.

Novel second–order directional derivatives. Directional derivatives evaluated at boundaries to impose the Neumann boundary condition are often discretised less accurate than the discrete gradient, which deteriorates accuracy of the solution. Since boundary points are generated by projecting fluid points, there always exists an adjacent fluid point along the normal of the boundary. This enables new discrete forms of the directional derivative to be derived by using information from the wall and relevant fluid point. The new form of the directional derivative has been proven to yield second–order accurate approximation and consistent system of equations for the Poisson equation.

The free surface is robustly detected. Meshless method have troubles detecting the points on the edge of a point cloud, i.e. points that are located on the free surface. The detected points impose the free surface boundary condition that is imperative for obtaining a unique and correct pressure field each time step. The renormalisation correction for the novel Laplacian in combination with a spherical geometrical test are used to quickly and robustly detect the points on the free surface.

Volume–conservative Lagrangian advection. The advection of meshless points is done in Lagrangian manner for each point, and simultaneously the volume conservation is handled by solving a set of geometrical constraints. The conservation can be interpreted as a re-meshing or optimisation method enforcing virtual volumes of points by enforcing equidistances between neighbouring points. The global volume of fluid is conserved at all time despite relying on finite differences and Lagrangian movement.

Unified inlet and open boundaries for the coupling. While flux–based methods control fluxes on boundary faces to generate/release flow from the domain, Lagrangian methods need to physically generate and remove parcels of the fluid. At boundaries of the domain, any boundary condition describing inlets or open boundaries may be implicitly imposed. On the other hand, explicit movement of the point cloud needs to be handled near the boundaries: a part of fluid leaving the domain is deleted, while a part of fluid that tries to enter the domain has to be created. A procedure for unifying these requirements is introduced.

Validation of the method for green water and other violent free surface flows. The method was used to simulate water entry, dam break, and sloshing experiments. The simulated characteristics of the flow were found to be in a good agreement with the experimental results. For impulsive fluid–structure impacts, where potential flow and FVMs can flounder, the proposed method was found to adequately reproduce velocity fields and the pressure rise/decline and peak values.

The implementation runs fully in parallel. State-of-the-art high performance computing requires of algorithms to be locally (by-node) parallelised, and only then distributed to nodes in order to be competitive. The global time–stepping procedure is sequential, but each step is a parallelised algorithm. New parallel techniques described in this thesis have been implemented to speed up: neighbour searching, linear–system solving, visualising results.

1.6.4. Structure of the Thesis

The remainder of the thesis is organised as follows.

Chapter 2 summarises the governing equations of incompressible flows, and presents the solution–scheme for the NSE in the Lagrangian context to model water on deck. The NSE equations are then simplified in order to model the waves and ship motions. The domain decomposition is introduced to couple the complex and simplified solvers for efficient solving of the global problem.

Chapter 3 deals with establishing a numerical method that solves the introduced governing equations efficiently. The chapter introduces the novel spatial operators and boundary conditions, for both solid boundaries and the free surface, that are used to model the problem. Lagrangian marching in space and time is described, which conserves the volume of fluid by constraining distances between closest neighbours.

Chapter 4 gives the overall solution procedure. Then the implementation of the method steps are discussed in detail. The chapter discusses how the system of linear equations is efficiently solved, and how the neighbours in the point cloud are quickly found.

Chapter 5 presents the verification and validation of the numerical method and its implementation, which is done by simulating various test cases and comparing the computed results to data from literature. The tests include artificial Poisson problems, cavity flow, water entries of various bodies, dam breaks, sloshing experiments, and isolated and periodical green water events.

Chapter 6 presents the conclusions regarding this thesis. In addition, proposals for future research are discussed.

2. Mathematical Model

This chapter presents the mathematical modelling of the physics of incompressible fluid flow. The chapter starts by introducing the Navier–Stokes equations that describe the flow in the Lagrangian context. Two efficient approaches for solving the Navier–Stokes equations applicable to green water problems are described, and their advantages and disadvantages are explained. The equations are then simplified to describe the potential flow theory. The coupling of two domains using complex and simplified equations is presented in order to model the global domain.

2.1. Navier–Stokes Equations

For an incompressible fluid, the time–dependent initial-boundary-value problem (IBVP) for Navier–Stokes equations (NSE) is given as:

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f} \qquad \boldsymbol{x} \in \Omega, \tag{2.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \boldsymbol{x} \in \Omega \cup \Gamma, \tag{2.2}$$

$$\boldsymbol{u} = \boldsymbol{g} \qquad \boldsymbol{x} \in \Gamma, \tag{2.3}$$

$$\boldsymbol{u}\,(t=0) = \boldsymbol{u}_0 \qquad \boldsymbol{x} \in \Omega, \tag{2.4}$$

where $\mathrm{D}/\mathrm{D}t$ is the Lagrangian derivative, $\boldsymbol{u}$ is the velocity vector, ρ is the fluid density, p is the pressure, ν is the kinematic viscosity of the fluid, $\boldsymbol{f}$ is the external acceleration vector acting on the fluid, $\boldsymbol{g}$ is the boundary velocity vector, $\boldsymbol{u}_0$ is the initial velocity vector. The time and spatial dependency is not explicitly written, but is implied. The fluid is moving in a $\mathbb{R}^d$ domain Ω, bounded by Γ, where $d = 2$ or $d = 3$ is the number of spatial dimensions. The system of equations (2.1)–(2.4) is sometimes referred to as the primitive–variable formulation of the incompressible NSE. Formally, the initial condition (2.4) should also satisfy the divergence-free condition through equation (2.2), i.e. $\nabla \cdot \boldsymbol{u}_0 = 0$ should hold everywhere in the domain. The governing equations describe a mixed elliptic–parabolic system of equations which must be solved simultaneously. The system of equations is parabolic in time, and elliptic in space. The unknowns in the equations are the velocity field $\boldsymbol{u}\,(\boldsymbol{x})$ and the pressure field $p\,(\boldsymbol{x})$, which evolve through time t.

The advantage of the Lagrangian description of the system of equations is the absence of the convective term, $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$, which is included in the Eulerian description of the system of equations. The convective velocity is defined as the difference between fluid particle velocity and the mesh velocity, and without a mesh, the fluid particles physically move by their velocities. This inertial term is nonlinear and responsible for the transfer of kinetic energy in the turbulent cascade. In the Lagrangian description of the flow, the nonlinear characteristics of the flow are reproduced by direct movement of fluid particles, avoiding the need of the discretisation of the convective term, which is a cumbersome task for any Eulerian CFD method.

The system of equations for the incompressible NSE (2.1)–(2.4) can be discretised in a variety of ways. Straightforward discretisations of equations (2.1) and (2.2) can lead to checker-board instabilities [87]. In any case, the pressure must maintain solenoidal or divergence-free condition of the velocity field described by the continuity equation (2.2) at every time instant. Since there exists no direct connection for the pressure between the continuity and momentum equations, some mathematical manipulations have to be made based on assumptions in order to establish a connection between the equations. An accurate, but also costly way to numerically advance equation (2.1) is to solve for $\boldsymbol{u}$ and p in a fully coupled manner. For better efficiency, it is useful to decouple the solution of the velocity from the solution of the pressure. The efficient split–step schemes that decouple the velocity and pressure are often used in meshless Lagrangian methods (e.g. [88]).

This decoupling is the source of a long discussion related to the proper numerical discretisation of NSE in the presence of boundaries, since the pressure term in equation (2.1) is not explicitly dependent on time, $p(\boldsymbol{x})$. Many numerical approaches require imposing extra boundary conditions, either for the pressure or for an intermediate velocity field, which can be non-trivial to choose and difficult to implement. Projection schemes have limited temporal accuracy as a result of matrix splitting errors and produce errors near boundaries where the pressure equation is not solved, which results in the formation of a numerical boundary layer, i.e. area of compressible fluid with non-solenoidal velocity [89]. This raises a question of proper boundary conditions for the pressure to satisfy equation (2.2) everywhere in the domain, which should be assessed in order to achieve first or second order of accuracy for the pressure and velocity [90, 91].

2.2. Projection Schemes

The projection schemes (also known as splitting methods, pressure correction methods, etc.) have been introduced as efficient way to reduce the computational cost of time–dependent incompressible viscous flow simulations in the velocity–pressure formulation. Meshless methods commonly utilise the classical projection scheme introduced by

Chorin [82] and Temam [92], or its derivatives, which are based on the Helmholtz–Hodge decomposition theorem for a vector field.

A vector field, such as the velocity vector field, may be written as the sum of the solenoidal (divergence–free) part and the irrotational (curl–free) part:

$$\boldsymbol{u} = \boldsymbol{u}_{sol} + \boldsymbol{u}_{irr}. \tag{2.5}$$

The divergence of the sum yields:

$$\nabla \cdot \boldsymbol{u} = \nabla \cdot \boldsymbol{u}_{irr}, \tag{2.6}$$

and conveniently, a curl-free vector field can be substituted by the gradient of some scalar function ($\boldsymbol{u}_{irr} = \nabla\phi$):

$$\nabla \cdot \boldsymbol{u} = \nabla^2\phi, \tag{2.7}$$

which is called the Poisson equation for the scalar function ϕ. If $\boldsymbol{u}$ is known, equation (2.7) can be solved for the scalar function ϕ and the solenoidal part can be simply obtained from:

$$\boldsymbol{u}_{sol} = \boldsymbol{u} - \nabla\phi. \tag{2.8}$$

The main idea of the projection methods in CFD is to split the resolution of the momentum equation into two steps. In the first step, an estimation of the velocity is computed, which does not satisfy the incompressibility constraint. Then, the projection operator is used to project this estimated velocity field onto the space of divergence-free vector fields.

2.2.1. Non-Incremental Projection Scheme

The classical, or non-incremental, projection scheme that is written in Lagrangian context [93] is described in the following text. In the first step, sometimes called the prediction step, estimated velocity is computed based on the viscous and external forces, without taking the pressure gradient of the previous time step into account:

$$\frac{\boldsymbol{u}^\star - \boldsymbol{u}^n}{\delta t} = \nu\nabla^2\boldsymbol{u}^\star + \boldsymbol{f}, \tag{2.9}$$

where $\boldsymbol{u}^\star$ is the estimated intermediate velocity field, δt is the time step size, and n is the number of the iteration in time domain. Equation (2.9) can be made explicit if some diffusion approximation is introduced, e.g. if the previous time–step velocity $\nabla^2\boldsymbol{u}^n$ is used instead of $\nabla^2\boldsymbol{u}^\star$, or if some extrapolation method is used to estimate $\nabla^2\boldsymbol{u}^\star$. Due to the Lagrangian nature of the method, locations of discrete points are also updated by an explicit integration in time without enforcing the incompressibility, i.e. the points are

shifted to their intermediate positions:

$$\boldsymbol{x}^{\star} = \boldsymbol{x}^{n} + \delta t \boldsymbol{u}^{\star}. \tag{2.10}$$

In the second step, often referred to as the correction step, the estimated velocity is corrected through its projection onto the vectorial space of divergence-free vectors, by employing the pressure gradient:

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^{\star}}{\delta t} = -\frac{1}{\rho}\nabla p^{n+1}, \tag{2.11}$$

and Lagrangian points are moved explicitly using the corrected velocity, e.g.:

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^{n} + \delta t \left(\frac{\boldsymbol{u}^{n+1} + \boldsymbol{u}^{n}}{2} \right).$$

Before applying the correction equation (2.11), the pressure field is computed through the pressure Poisson equation (PPE), which is defined as:

$$\nabla^2 p^{n+1} = \frac{\rho}{\delta t} \nabla \cdot \boldsymbol{u}^{\star}, \tag{2.12}$$

which corresponds to the enforcement of the incompressibility constraint (2.2) on equation (2.11). The boundary condition to solve equation (2.12) is obtained by projecting equation (2.11) onto the wall normal, which yields:

$$\boldsymbol{n} \cdot \nabla p^{n+1} = -\frac{\rho}{\delta t} \boldsymbol{n} \cdot \left(\boldsymbol{u}^{n+1} - \boldsymbol{u}^{\star} \right) = 0, \quad \boldsymbol{x} \in \Gamma, \tag{2.13}$$

since no-slip boundary condition must hold, $\boldsymbol{n} \cdot \boldsymbol{u}^{n+1} = 0$. This homogeneous Neumann condition is artificial and was shown to induce a numerical boundary layer which deteriorates the scheme convergence [94].

When the explicit version of equation (2.9) is used, projecting equation (2.11) onto the wall normal yields non-homogeneous boundary condition, which reads:

$$\boldsymbol{n} \cdot \nabla p^{n+1} = \frac{\rho}{\delta t} \boldsymbol{n} \cdot \boldsymbol{u}^{\star} = \rho\, \boldsymbol{n} \cdot \left(\boldsymbol{f} + \nu \nabla^2 \boldsymbol{u}^{n} \right), \quad \boldsymbol{x} \in \Gamma,$$

which results in more accurate pressure solutions near walls than those obtained by the homogeneous Neumann condition [90]. However, particularly for moderate and low Reynolds numbers, the effects of the numerical boundary layers can still be problematic [89].

2.2.2. Incremental Projection Schemes

By including the pressure gradient from the previous time step, it is possible to increase the accuracy of the scheme, which corresponds to incremental projection schemes. The standard incremental projection scheme includes the pressure gradient in the first step of the scheme, but keeps the wall boundary condition for the pressure homogeneous. The rotational incremental projection scheme besides including the pressure gradient solves the PPE for χ, which is defined as $\chi = p^{n+1} - p^n + \nu\nabla^2\boldsymbol{u}^\star$. More information on the increased accuracy of rotational projection schemes, validated in the context of the incompressible SPH method, is given in [95]. In the prediction step, any order of backward difference formula may be used to approximate the time derivative of the velocity (if it is considered as continuous in time). Although, non-incremental schemes with inaccurate boundary conditions do not benefit from higher–order difference formulae [89].

2.2.3. Velocity–Correction Schemes

Within velocity–correction schemes, the roles of the pressure and viscous terms are inverted as compared to previously introduced pressure–correction schemes. In the prediction step, the pressure gradient is kept, while the viscous term is either ignored or treated explicitly. The pressure is obtained through a Poisson equation, and the schemes can be built in non-incremental or incremental form.

2.3. Velocity–Pressure Formulation

Velocity–pressure formulations retain the efficiency of the projection methods, while not suffering from the numerical boundary layers, or restrictions in temporal accuracy. The price for the increased accuracy are more complicated boundary conditions.

If one takes the divergence of the momentum equation (2.1) and applies the divergence-free constraint equation (2.2), then equation (2.2) is replaced by the elliptic equation for the pressure, i.e. Poisson pressure equation (PPE), and the divergence-free boundary condition. The new IBVP is expressed as:

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f} \qquad \boldsymbol{x} \in \Omega, \tag{2.14}$$

$$\nabla^2 p = \rho\,(\nabla \cdot \boldsymbol{f}) \qquad \boldsymbol{x} \in \Omega, \tag{2.15}$$

with the boundary and initial conditions:

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \boldsymbol{x} \in \Gamma, \quad t \geq 0, \tag{2.16}$$

$$\boldsymbol{u} = \boldsymbol{g} \qquad \boldsymbol{x} \in \Gamma, \quad t \geq 0, \tag{2.17}$$

$$\boldsymbol{u} = \boldsymbol{u}_0 \qquad \boldsymbol{x} \in \Omega, \quad t = 0. \tag{2.18}$$

The viscous term and the term with a time derivative is eliminated from equation (2.15) due to equation (2.2). Equation (2.15) also suggests that the pressure field can be obtained provided that the velocity field is known, which can be formulated as:

$$p\left(\boldsymbol{x}\right) = P\left(\boldsymbol{u}\left(\boldsymbol{x}\right)\right), \tag{2.19}$$

where P is the function that yields the pressure for any given velocity field that solves the NSE. On the other hand, the formulation in the presented form does not provide boundary conditions for the pressure. Therefore, for the solution of the system of equations (2.14) and (2.15), an additional boundary condition should be introduced in order to make the problem well posed. It is often not emphasised in the literature, but methods implicitly or explicitly impose divergence-free boundary condition, equation (2.16), because the continuity constraint equation (2.2) should hold everywhere in the domain $\Omega \cup \Gamma$, and equation (2.15) has replaced equation (2.2) in Ω. Such boundary condition cannot easily be known *a priori* as a function of $\boldsymbol{u}$ [96]. Even though this additional boundary condition does not look like a direct pressure boundary condition, it is responsible for satisfying the following important conditions:

1. it makes the system of equations (2.14) and (2.15) *well posed*,
2. it is *consistent* with the original formulation of NSE (2.1) and (2.2),
3. it makes the velocity–pressure formulation *equivalent* to the velocity–divergence formulation.

2.3.1. The Pressure Equation

The explicitly imposed divergence-free boundary condition requires a global constraint [97], which is in practice rarely used. Alternatively, a natural candidate for the implicitly imposed divergence-free boundary condition is given by the normal component of the momentum equation along Γ, i.e. by dotting equation (2.14) through by $\boldsymbol{n}$ and evaluating

at the boundary. The PPE with the appropriate boundary condition is given as:

$$\nabla^2 p = \rho \left(\nabla \cdot \boldsymbol{f} - \frac{\partial \left(\nabla \cdot \boldsymbol{u} \right)}{\partial t} \right), \qquad \boldsymbol{x} \in \Omega, \tag{2.20}$$

$$\boldsymbol{n} \cdot \nabla p = \rho \, \boldsymbol{n} \cdot \left(\boldsymbol{f} - \frac{\partial \boldsymbol{g}}{\partial t} + \nu \nabla^2 \boldsymbol{u} \right), \qquad \boldsymbol{x} \in \Gamma. \tag{2.21}$$

Although the continuity constraint demands that the velocity field is free of divergence at any time, the term $\nabla \cdot \boldsymbol{u}$ missing from equation (2.15) is formally preserved in equation (2.20) in order to reinforce assumptions from which the discrete version is derived in section §3.6.

Recall the "curl of the curl" vector identity, which applied to the velocity is written as:

$$\nabla^2 \boldsymbol{u} = -\nabla \times \nabla \times \boldsymbol{u} + \nabla \left(\nabla \cdot \boldsymbol{u} \right). \tag{2.22}$$

By applying the incompressibility constraint (2.2), the Neumann boundary condition (2.21) can be rewritten as:

$$\boldsymbol{n} \cdot \nabla p = \rho \, \boldsymbol{n} \cdot \left(\boldsymbol{f} - \frac{\partial \boldsymbol{g}}{\partial t} - \nu \nabla \times \nabla \times \boldsymbol{u} \right), \qquad \boldsymbol{x} \in \Gamma. \tag{2.23}$$

The idea of the correct pressure boundary condition equation (2.21) was first proposed for projection methods by Orszag *et al.* [98], which was later validated by various researchers. Henshaw [99] showed that using the curl–curl boundary condition (2.23) is numerically more stable than equation (2.21), because it adds new information to the system and removes the velocity divergence on the boundary in the highest order term. If equation (2.16) holds at the start of the simulation, the normal derivative of equation (2.16) will be zero for all time. Some remarks on the boundary condition for the pressure are given in Appendix A.

2.3.2. The Momentum Equation

The non-iterative velocity–pressure formulation relies on the fact that the pressure term is always treated explicitly in time so that the velocity and pressure updates are completely decoupled. In the Eulerian specification of the flow, the nonlinear convection term (which is absent in the Lagrangian case) is also treated explicitly in time discretisation, together with the pressure.

What remains to be decided is the temporal discretisation of the viscous term, which evidently does not affect the discretisation of the pressure in the PPE formulation. Whether to treat the viscous term explicitly or implicitly in time should depend on the Reynolds number and the required spatial resolution. Implicit treatment of the viscous term does

nothing to stabilise the convection for large Reynolds numbers, but e.g. semi-implicit time discretisation of the momentum equation with Crank–Nicholson scheme applied to the viscous term could stabilise the convection for low Reynolds number [97].

In conclusion, the numerical scheme is free to solve equation (2.14) for the velocity by any integration technique in which the pressure is obtained beforehand.

2.3.3. Advantages of the Formulation

Velocity–pressure formulations of the NSE have important advantages over the standard formulation:

- The pressure is not implicitly coupled to the velocity through the momentum equation and incompressibility. Therefore, it can be directly recovered from the known velocity field by solving a Poisson equation. This allows for marching the velocity field in time straightforwardly, using the momentum equation with the pressure interpreted as a function of the velocity.
- There are no spurious boundary layers for the velocity or the pressure, since there are no boundary conditions ambiguities concerning the pressure, i.e. errors induced by incorrect boundary conditions do not occur, while the incompressibility is enforced at all times.

It should be noted, however, that the PPE formulation and the classical formulation of the NSE are not equivalent in the case of steady state flows [97].

2.4. Boundary Conditions

Imposing appropriate boundary conditions (BCs) is of preeminent importance for the success of every numerical algorithm. For the incompressible Navier–Stokes equations, the type of boundary conditions to be imposed are dependent on the physics of the flow once the geometry and topology of the selected problem have been determined. In this thesis, the applications and the flow geometry solved in general belong to external flow problems, where the normal unit vector out of the solid surface points away from the surface into the fluid. For external flows, boundary conditions are grouped into:

- physical boundaries due to solid bodies and water–air free surfaces,
- domain truncation boundaries due to limiting infinitely unbounded fluid to an area of interest.

Mathematical formulations of the relevant boundary conditions are listed in table 2.1, which are explained in the following text.

Table 2.1.: Various boundary conditions for the NSE.

Type	Location	Mathematical formulation
No-slip wall	Γ_{wall}	$\begin{cases} \boldsymbol{u} = \boldsymbol{g} \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases}$
Free-slip wall	Γ_{slip}	$\begin{cases} \boldsymbol{n} \cdot \boldsymbol{u} = \boldsymbol{n} \cdot \boldsymbol{g} \\ \boldsymbol{n} \cdot \nabla (\boldsymbol{t} \cdot \boldsymbol{u}) = 0 \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases}$
Free surface	Γ_{fs}	$\begin{cases} \boldsymbol{u} = \check{\boldsymbol{g}} \\ p = P \end{cases}$
Imposed velocity	Γ_{in}	$\begin{cases} \boldsymbol{u} = \boldsymbol{g} \\ \boldsymbol{n} \cdot \nabla p = 0 \end{cases}$
Imposed velocity and pressure	Γ_{in}	$\begin{cases} \boldsymbol{u} = \boldsymbol{g} \\ p = P \end{cases}$
Symmetry	Γ_{sym}	$\begin{cases} \boldsymbol{n} \cdot \boldsymbol{u} = 0 \quad \boldsymbol{n} \cdot \nabla \boldsymbol{u}_t = 0 \\ \boldsymbol{n} \cdot \nabla p = 0 \end{cases}$
Open	Γ_{open}	$\begin{cases} (\boldsymbol{n} \cdot \nabla) \boldsymbol{u} = \boldsymbol{0} \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases}$

2.4.1. Free Surface

The free–surface boundary condition can formally be treated as a condition that truncates the domain. Since the air flow can be ignored when predicting events under consideration (not to be mixed with the water compressibility), the air is not explicitly modelled. In contrast to two-phase methods that naturally embed interfacial jump conditions within their formulation, here the dynamic jump condition is enforced explicitly. Therefore, the free surface boundary condition serves solvability of the NSE and additionally truncates the domain. The kinematic free–surface boundary condition used to compute the evolution of the free surface is naturally handled by the Lagrangian advection, of course, if the NSE are accurately solved. The dynamic free–surface boundary condition requires continuous pressure and shear stress across the free surface. The external stress and surface tension are assumed zero, the gradients of the normal velocity in the tangential directions are assumed small. Under these assumptions, an adequate approximation to enforce the boundary condition is to take the pressure as constant:

$$p(\boldsymbol{x}) = p_{air}, \qquad \boldsymbol{x} \in \Gamma_{fs}, \tag{2.24}$$

where p_{air} is the imposed pressure, taken as zero or as atmospheric pressure value measured at the interface, Γ_{fs}. Surface tension can be easily included by adding the term $\sigma\kappa$ to the right-hand-side of equation (2.24), where σ is the surface tension coefficient and κ

is the interfacial curvature defined as the divergence of the normals of the free surface. Equation (2.24) is a Dirichlet boundary condition for the PPE described in section 2.3.1, which ensures a unique solution to the problem.

2.4.2. Solid Walls

Bodies, i.e. solid surfaces, may be stationary or moving by either imposed or calculated velocities. Physically, the fluid molecules at the solid boundary move with the boundary, i.e. the following holds for each time instant:

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma_{wall}, \tag{2.25}$$

where $\boldsymbol{g}$ is the boundary velocity vector at a specific location on the boundary Γ_{wall}. This boundary condition is usually referred to as the "no-slip" or "non-slip" wall boundary condition.

The divergence-free condition for the no-slip wall, listed in table 2.1, results in the Neumann boundary condition equation (2.23) for the PPE as described in section 2.3.1, with the boundary velocity $\boldsymbol{g}(\boldsymbol{x})$. Some researchers note that the no-slip condition is physically correct, but numerically may be too strict which yields discrepancy with experiments (e.g. see [100, 101]). This may be due to the spatial discretisation near the walls, the correctness of the diffusion discretisation near walls, etc. In such cases, they suggest using the "free-slip" wall condition, listed in table 2.1. The primary goal of the free-slip boundary condition is to eliminate the velocity component perpendicular to the boundary.

2.4.3. Inlet Boundaries

When forcing the fluid to flow into the computational domain, the profile of the flow should be specified while retaining the properties of the NSE. The velocity of the fluid on the inlet boundary is imposed:

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma_{in}, \tag{2.26}$$

where $\boldsymbol{g}$ is the vector of the fluid velocity defined on the inlet boundary surface Γ_{in}. In most cases, the pressure distribution on the inlet boundary surface is known, and therefore, can be imposed when solving the PPE:

$$p(\boldsymbol{x}) = P(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma_{in}, \tag{2.27}$$

where $P(\boldsymbol{x})$ is the scalar field of the fluid pressure defined on the inlet boundary surface or volume. However, when the Dirichlet boundary conditions for the pressure are imposed on

some boundaries, the incompressibility constraint is not necessarily satisfied on them [102]. One should be careful when imposing the pressure without using relaxation and small time steps, since the numerical velocity divergence $\langle \nabla \cdot \boldsymbol{u} \rangle$ can build up and destabilise the simulation. This is described in chapter 3 in the context of the numerical methodology used in this thesis.

If $P(\boldsymbol{x})$ is not known or the Dirichlet boundary condition is to be avoided, the Neumann boundary condition for the PPE must be defined by the rate of change in pressure in the normal direction of the boundary. Generally, equation (2.23) should hold. For flows with pressure gradient mostly tangential to the boundary, as for waves, it can be taken that the pressure does not change in the direction normal to the inlet boundary surface:

$$\boldsymbol{n} \cdot \nabla p = 0, \qquad \boldsymbol{x} \in \Gamma_{in}, \tag{2.28}$$

where one should pay attention that the bodies in the simulation are not too close to the inlet, in order to let the flow fully develop before meeting the solid surfaces. In addition, relaxation zone in front of the inlet boundary is usually considered to alleviate the problems of the discrepancy between the numerical and target solution, which is described in section 3.9.4.

2.4.4. Open Boundaries

Ideally, an infinite unbounded fluid represents the physical domain for a problem featuring external–flow hydrodynamics. Computationally realistic domains are truncated to sizes which should have no influence on the computed solution. Open boundaries enable the fluid to enter and leave the computational domain, while insignificantly affecting the development of the flow. Fluid parts are removed when they cross the open boundary. In addition, fluid parts are generated when flow pushes the fluid into the domain. Sometimes open boundaries are improperly equalised to outlet boundaries, which force the flow to only leave the domain. With open boundaries, there is no general agreement on which kind of boundary conditions are physically and mathematically correct, and numerically appropriate [103]. To simplify the implementation of the numerical scheme, it can be assumed that at open boundaries far away from bodies, the velocity does not change in space, which is written in table 2.1. As in the case of inlet boundaries, a relaxation zone adjacent to the open boundary can be set-up to force non-reflecting outlet flow.

Liu [104] suggested to use the Dirichlet type boundary condition for the PPE, and proved unconditional stability of a semi-implicit scheme for the Stokes equation. The condition is defined as:

$$p = \nu \, \boldsymbol{n} \cdot (\boldsymbol{n} \cdot \nabla) \, \boldsymbol{u} - c_o \nu \nabla \cdot \boldsymbol{u}, \qquad \boldsymbol{x} \in \Gamma_{open}, \tag{2.29}$$

where c_o is a multiplier term in the range [0, 2] that is important to the accuracy and stability of scheme [105]. The Dirichlet boundary condition ensures a unique solution to the PPE, which is favourable for internal flows [95], but for free surface flows it would not be an easy task to add the missing hydrostatic part to the equation, based on the dynamic free surface elevation. In [106] an additional Poisson equation is introduced, which is solved for surfaces of open boundaries. The result is then taken as the Dirichlet boundary condition for the main PPE.

In this thesis, Neumann type boundary condition for the PPE is simply used. Since the fluid body is bounded by free surface, a unique solution is guaranteed. There is no point in adding complexity by introducing another Poisson equation, but to specify directional derivative with approximated values, assuming that the boundary is far enough from the area of interest not to alter the solution. With respect to equation (2.21), the boundary condition is written as:

$$\boldsymbol{n} \cdot \nabla p = \rho \, \boldsymbol{n} \cdot \left(\boldsymbol{f} - \frac{\partial \boldsymbol{u}}{\partial t} + \nu \nabla^2 \boldsymbol{u} \right), \qquad \boldsymbol{x} \in \Gamma_{open}, \tag{2.30}$$

where the acceleration of the flow, $\partial \boldsymbol{u}/\partial t$, and the velocity Laplacian, $\nabla^2 \boldsymbol{u}$, is either known or extrapolated at the boundary, $\boldsymbol{x} \in \Gamma_{open}$.

2.5. Turbulence

NSE present chaotic behaviour for high values of Reynolds number, which is known as the turbulence. Turbulent flows are varying rapidly in time, and are very sensitive to initial conditions. If the NSE are numerically solved on a extremely fine spatial discretisation, turbulence does not need to be modelled. This straightforward approach is called the Direct Numerical Simulation (DNS). DNS is used mostly to study turbulent behaviours, and is not suitable for industrial applications due to its computational cost, since it requires fine three-dimensional discretisation, and therefore, small time steps. A coarse grid is able to resolve larger eddies in the flow but not the ones smaller than the cell size. Physically, there is an interaction between the motions on all scales, in a way that the result for the large scales is somewhat inaccurate without taking into account the influence of the fine scales on the larger ones.

2.5.1. Dam Break as Reference

When talking about the dam break as an event that is similar to the event of wetting the deck, resolving the turbulence leads to most accurate numerical solutions of the wave front. Nevertheless, the shape of the wave front, when resolved with the direct discretisation of

the diffusion term, compares well with experimental observations both in horizontal and sloped channels [107]. This direct discretisation of the diffusion term exactly reproduces laminar flows. If applied to turbulent flows with non-DNS grids, the simulation can formally be described as an under–resolved DNS. Park *et al.* [108] presented numerical results of dam breaks obtained by the modelled turbulence and inviscid flow, and noted that inviscid flow model should not be used to simulate dam breaks. On the other hand, artificially amplifying the turbulence intensity yielded smoothed results of the pressure at impact, but the authors did not account for the effects of water compressibility at the impact or physically justify the turbulence intensity values. Arnold [109] obtained similar results with turbulence models and direct simulation. Janosi *et al.* [110] experimentally found that the vertical cross-section of the flow in a dry channel is smooth, while a front that flows over a fluid layer generates unstable jets, wave breaking, bubble trapping, etc.

In conclusion, the laminar model (or under–resolved DNS) with sufficient spatial discretisation reproduces similar flow to experiments of dam break over a dry bed.

2.5.2. Green Water and Turbulence

For events like green water, where impulsive fluid impacts on the structure dominate the physics, turbulence effects are of minor or negligible importance just around the time instant of the impact. The impact is usually an inertia–dominated event, which happens in just a few milliseconds. As described, viscosity effects should not be neglected at the time interval at which the flow develops before the wave front hits an obstacle. Nonlinearities that occur when the wave hits the ship (often the bow flare) are of high importance for further development of the flow of water on the deck. The interaction of high waves, and usually strong ship motions, produces large jets and backflow near the point of impact. The backflow usually contains large eddies, which can be reproduced by under–resolved DNS, and are of importance in fluid–structure problems. Lagrangian nature of the NSE, along with fine spatial discretisation where needed, favourably answers to such requirements of reproduction of eddies and rapid jets. Gatin *et al.* [30] haven't employed turbulence modelling in their numerical study on green water, which was done by a FVM–VOF solver. The authors consider that the modelling has a negligible influence on pressure distribution at the structure, but that the influence of turbulence should be investigated in the future.

In conclusion, fine spatial discretisation of the problem, i.e. dense meshless point cloud to represent the water, by direct simulation is sufficient to capture relevant turbulent structures in green water incidents. Therefore, NSE are solved without the additional modelling of the turbulence. Future work should include an investigation of the effects of a turbulence model on the flow development before the impact of water on deck. Most

often meshless methods implement Large Eddy Simulation (LES) for the turbulence, which is introduced in Appendix B.

2.6. Potential Flow

2.6.1. The Theory

In the potential flow theory, it is assumed that the fluid is incompressible, inviscid and that the flow is irrotational. Therefore, a velocity potential function $\phi(\boldsymbol{x})$ can be introduced to describe the flow velocity field:

$$\boldsymbol{u}(\boldsymbol{x}) = \nabla\phi(\boldsymbol{x}), \tag{2.31}$$

since the velocity field of an irrotational flow is curl-free ($\nabla \times \boldsymbol{u} = 0$) and the curl of the gradient of any scalar field is always null vector ($\nabla \times \nabla\phi = \boldsymbol{0}$). Considering that the velocity field of an incompressible fluid is solenoidal ($\nabla \cdot \boldsymbol{u} = 0$), the divergence of equation (2.31) leads to the Laplace equation of the potential flow:

$$\nabla^2\phi(\boldsymbol{x}) = 0. \tag{2.32}$$

Physically, the introduced flow restrictions prevent the reproduction of important fluid flow behaviours such as the separation, skin–friction drag and transonic shocks [111]. The fluid domain is bounded by the free surface, the wetted body surface, the bottom, and the control surface in the far field. The flow is strictly tangential to the solid boundary and the resulting potential is zero in the body interior, i.e. fluid is neither able to enter or leave a closed surface.

The impermeability boundary condition is imposed at the bottom, which states that the velocity is strictly tangential to the bottom:

$$\boldsymbol{n} \cdot \nabla\phi = 0, \tag{2.33}$$

where $\boldsymbol{n} = \{0,\, 0,\, 1\}$ if the bottom is considered to be horizontal.

The particles on the free surface remain there at all time, bearing constant pressure across the interface. The fully nonlinear free surface boundary conditions can be written in terms

of surface quantities:

$$\frac{\partial \phi_s}{\partial t} = -g\eta + \frac{1}{2}\left[(1 + \nabla\eta \cdot \nabla\eta)\left(\frac{\partial \phi}{\partial z}\right)^2 - \nabla\phi \cdot \nabla\phi\right], \tag{2.34}$$

$$\frac{\partial \eta}{\partial t} = (1 + \nabla\eta \cdot \nabla\eta)\frac{\partial \phi}{\partial z} - \nabla\phi_s \cdot \nabla\eta, \tag{2.35}$$

where $\phi_s(x,\, y,\, t)$ is the surface potential, and $\eta(x,\, y,\, t)$ is the free surface elevation where z is evaluated at. Therefore, $\phi_s(x,\, y,\, t) = \phi(\{x,\, y,\, \eta\},\, t)$. Equations (2.34) and (2.35) are nonlinear PDEs defining boundary conditions that can be marched in time, while the remaining volumetric quantity of the vertical velocity, $\partial\phi/\partial z$, may be evaluated according to [112].

For small elevations compared to wavelengths $(\partial\phi/\partial x)^2 \ll \partial\phi/\partial t$, so equation (2.34) becomes:

$$\frac{\partial \phi_s}{\partial t} = -g\eta, \tag{2.36}$$

while equation (2.35) becomes:

$$\frac{\partial \eta}{\partial t} = \frac{\partial \phi}{\partial z}. \tag{2.37}$$

Since the wave elevation, which is proportional to the wave amplitude, is small compared to the wavelength, equations (2.36) and (2.37) may be further simplified taking the Taylor's series expansion of ϕ_s about $z = 0$. It can be shown that nonlinear terms are very small and can be neglected. Substituting $z = 0$ for $z = \eta$ everywhere, the free surface boundary conditions are written as:

$$\frac{\partial \phi}{\partial z} = -\frac{1}{g}\frac{\partial^2 \phi}{\partial t^2}, \tag{2.38}$$

$$\eta = -\frac{1}{g}\frac{\partial \phi}{\partial t}. \tag{2.39}$$

2.6.2. Waves Modelling

Linear–wave boundary–value problem consists of the Laplace equation (2.32) for $z < 0$, sea floor boundary condition (2.33) and the free surface boundary conditions (2.38) and (2.39). The simplest wave description that is used in CFD methods is the linear wave description, also referred to as the Airy wave theory. The solution for linear free–surface waves can be obtained as follows. Since the free surface slope is very small, the potential may be written as:

$$\phi(x,\, z,\, t) = P_W(z)\, \sin(\omega\, t - k\, x), \tag{2.40}$$

where k is the wave number, and P_W is obtained by substituting the equation into the Laplace equation and solving for P_W:

$$\phi\,(x,\,z,\,t) = \{C_1 \exp\,(kz) + C_2 \exp\,(-kz)\}\,\sin\,(\omega\,t - k\,x)\,, \tag{2.41}$$

where C_1 and C_2 are the constants that will be obtained from the boundary conditions at the free surface and sea floor. Substituting equation (2.33) into the above equation yields:

$$\phi\,(x,\,z,\,t) = C\,\cosh\,\{k\,(z+h)\}\,\sin\,(\omega\,t - k\,x)\,, \tag{2.42}$$

where h is the water depth. Now the unknown constant C is determined from the linearised dynamic boundary condition at the free surface, equation (2.39). The free surface elevation defined by equations (2.39) and (2.42) writes:

$$\eta\,(x,\,t) = \eta_a \cos\,(\omega\,t - k\,x)\,, \tag{2.43}$$

where $\eta_a = -\omega/g\,C\,\cosh\,(k\,h)$, from which the C is obtained and substituted back into equation (2.42) yields the final expression for the velocity potential:

$$\phi\,(x,\,z,\,t) = -\frac{\eta_a\,g}{\omega}\,\frac{\cosh\,\{k\,(z+h)\}}{\cosh\,(k\,h)}\,\sin\,(\omega\,t - k\,x)\,. \tag{2.44}$$

The linearised free surface kinematic boundary condition equation (2.38) leads to the dispersion relation, which connects the wave number and wave frequency:

$$k\,\tanh\,(k\,h) = \frac{\omega^2}{g}, \tag{2.45}$$

which for deep water reduces to $k = \omega^2/g$. The velocity vector $\boldsymbol{u}$ is obtained by equation (2.31), i.e. the components of the velocity vector are written as:

$$u_x\,(x,\,z,\,t) = \frac{\eta_a\,g\,k}{\omega}\,\frac{\cosh\,\{k\,(z+h)\}}{\cosh\,(k\,h)}\,\cos\,(\omega\,t - k\,x)\,, \tag{2.46}$$

$$u_z\,(x,\,z,\,t) = -\frac{\eta_a\,g\,k}{\omega}\,\frac{\sinh\,\{k\,(z+h)\}}{\cosh\,(k\,h)}\,\sin\,(\omega\,t - k\,x)\,. \tag{2.47}$$

The range of the applicability of the theory is constrained to waves with small height $H_W = 2\,\eta_a$ compared to its wavelength λ_W, as strict as $H_W/\lambda_W < 0.0062$ according to Le Méhauté [113]. The linear wave theory can be used to generate irregular sea states, by applying the superposition principle. By specifying vectors of frequencies, amplitudes and phases, an irregular linear wave can be built as the sum of the individual components. The linear theory gives elliptic particle paths in Lagrangian coordinates, while the free surface and lines of equipressure are trochoidal. On the other hand, the theory of Stokes at some

order of approximation is characterised by the sum of number of sinusoidal components equal to the order of the approximation, i.e the solution is represented by Fourier series. The coefficients in the series can be written as perturbation expansions found by satisfying boundary conditions on the free surface, and solving the resulting set of ordered equations. As a result, the wave crest are more peaked and the troughs are flatter. The validation problems solved in this thesis are modelled by the linear wave theory, but the introduced numerical method is not limited to the linear wave theory.

Linear wave theory is limited to infinitesimally small waves, and assumptions regarding wave kinematics above the mean waterline must be introduced. Rough approximation is that above the mean waterline, wave kinematics may be assumed equal to the value at $z = 0$. An alternative is to stretch the limits of vertical coordinates adjust to instantaneous free surface elevation, which is referred to as Wheeler's modification. This is achieved by substituting the vertical coordinate z with the scaled coordinate $\tilde{z}$:

$$\tilde{z} = [z - \eta(x,\, t)] \frac{d}{d + \eta(x,\, t)}. \tag{2.48}$$

It is more appropriate to use Wheeler's modification than the constant extrapolation, since it always satisfies zero pressure at the instantaneous free surface.

2.6.3. Seakeeping

The problem of vessel seakeeping may be solved by introducing the wetted hull surface to the boundary–value problem, which is shortly presented here. In addition to the boundary conditions for waves, the impermeability boundary condition is imposed at the hull surface:

$$\boldsymbol{n} \cdot \nabla\phi = \boldsymbol{n} \cdot \boldsymbol{g}, \tag{2.49}$$

where $\boldsymbol{g}$ is the vector of body velocity, and $\boldsymbol{n}$ is the surface normal vector pointing inside the fluid. The hull surface is usually discretised into quadrilateral or triangular panels. The panel method is based on a form of the Green theorem where the velocity potential of the fluid at any point is represented by the surface distribution of singularities over the boundary surfaces. To solve the first–order boundary value problem, one may consider the fundamental solution:

$$\nabla^2 G(\boldsymbol{P},\, \boldsymbol{Q},\, t) = 4\,\pi\,\delta(\boldsymbol{P} - \boldsymbol{Q}), \tag{2.50}$$

where $\boldsymbol{P}$ is the field point with coordinates $\{x,\, y,\, z\}$, $\boldsymbol{Q}$ is the singular point with coordinates $\{x',\, y',\, z'\}$, and the Dirac function is defined as $\delta(\boldsymbol{P} - \boldsymbol{Q}) = \delta(x - x')\,\delta(y - y')\,\delta(z - z')$. By applying Green's formula to the couple of harmonic functions $\{\phi,\, G\}$, it

follows:

$$4\,\pi\,\phi\,(\boldsymbol{P}) = \int_S \left[\frac{\partial\phi\,(\boldsymbol{Q})}{\partial n\,(\boldsymbol{Q})} G\,(\boldsymbol{P},\,\boldsymbol{Q}) - \phi\,(\boldsymbol{Q})\,\frac{\partial G\,(\boldsymbol{P},\,\boldsymbol{Q})}{\partial n\,(\boldsymbol{Q})}\right] \mathrm{d}S\,(\boldsymbol{Q})\,, \tag{2.51}$$

where S combines the boundary surfaces: the hull (Γ_H), mean free surface (Γ_F), sea floor (Γ_B) and the surface at infinity (Γ_C). The left-hand-side is the result of the domain integral while the terms on the right-hand-side come from the transformation of the domain integral to the surface integral on the boundaries according to the formula of Ostrogradsky.

Since the integral over the surface at infinity is zero, and the integral over the sea floor is zero as well, equation (2.51) is then reduced to evaluating at the hull surface and adding the integral over the free surface by employing equation (2.38). This free surface integral is simplified or zero in most cases of wave radiation and diffraction without forward speed, or can be transformed into a line integral for the wave radiation and diffraction around an advancing ship at a uniform speed [114]. Therefore, equation (2.51) is solved for ϕ on Γ_H and Γ_F. The velocity potential and the Green function are assumed to be harmonic, so the time–harmonic potential is expressed as the sum of radiation components due to 6–DOF oscillations, incoming waves and the potential due to diffracted waves. The solution to the problem is thoroughly explained in [114]. The pressure determined from the potential on each panel is integrated in order to obtain required forces and moments.

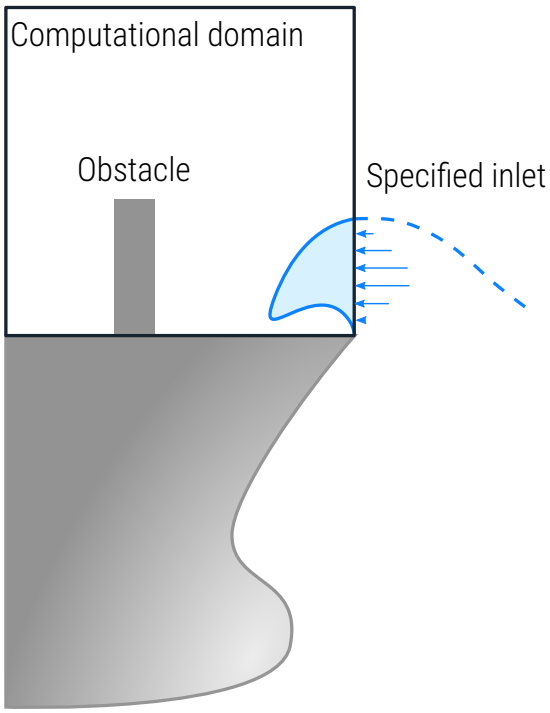

Figure 2.1.: Modelling green water flow by generating flow that wets the deck.

2.7. Domain Decomposition

The domain decomposition technique, also known as the zonal modelling, separates the problem spatially into two parts that are calculated by different solvers, even though they are physically overlapping. There are multiple advantages in using the zonal modelling.

The most rigorous approximation introduced to predict green water loads is described by figure 1.6. A water column is released and flows along the deck until it reaches an obstacle where loads of the impact are measured. This technique was a precursor to the zonal modelling, because it deals only with a local domain describing the deck and an obstacle, while the input is pre-calculated from various assumptions [20]. The improved local modelling includes an inlet boundary that generates a wave into the local domain capturing the deck, which is depicted by figure 2.1. The domain may or may not move with an imposed motion to simulate the ship movement. The disadvantages of this technique are:

- inlet boundary conditions must be imposed as a function in time,
- the function must be known, i.e. assumed, by using some simpler methods that neglect flow nonlinearities near the ship,
- the simulation can properly capture a single impact, before the reverted flow mixes with the generated flow by the inlet.

It is close to impossible to adequately assume the flow to be generated at the inlet shown in figure 2.1 by using linear solvers. Therefore, in order to suppress the errors that come out of assumptions needed to model the flow locally, but to keep the computational performance on a high level, the complete domain is decomposed into two physically overlapping domains which may communicate to simulate the problem in whole. This concept is drawn in figure 2.2, where the two domains and their communication are described as follows.

2.7.1. Lower–Fidelity Domain

Waves are generated, diffracted and radiated around the vessel by a fast solver solving a large fluid domain, which is usually based on the linearised potential–flow equations, described in section §2.6. In addition to the flow, the motions of the vessel are usually solved by the solver. This solver is employed to evaluate a wide range of scenarios, usually in the frequency domain. From the performed simulations, critical events are isolated to be simulated in another domain of interest, i.e. another zone.

Alternatively, High–Order Spectral (HOS) methods enable the simulation of highly non-linear wave–fields [115]. They are applied to study freak waves made of high count

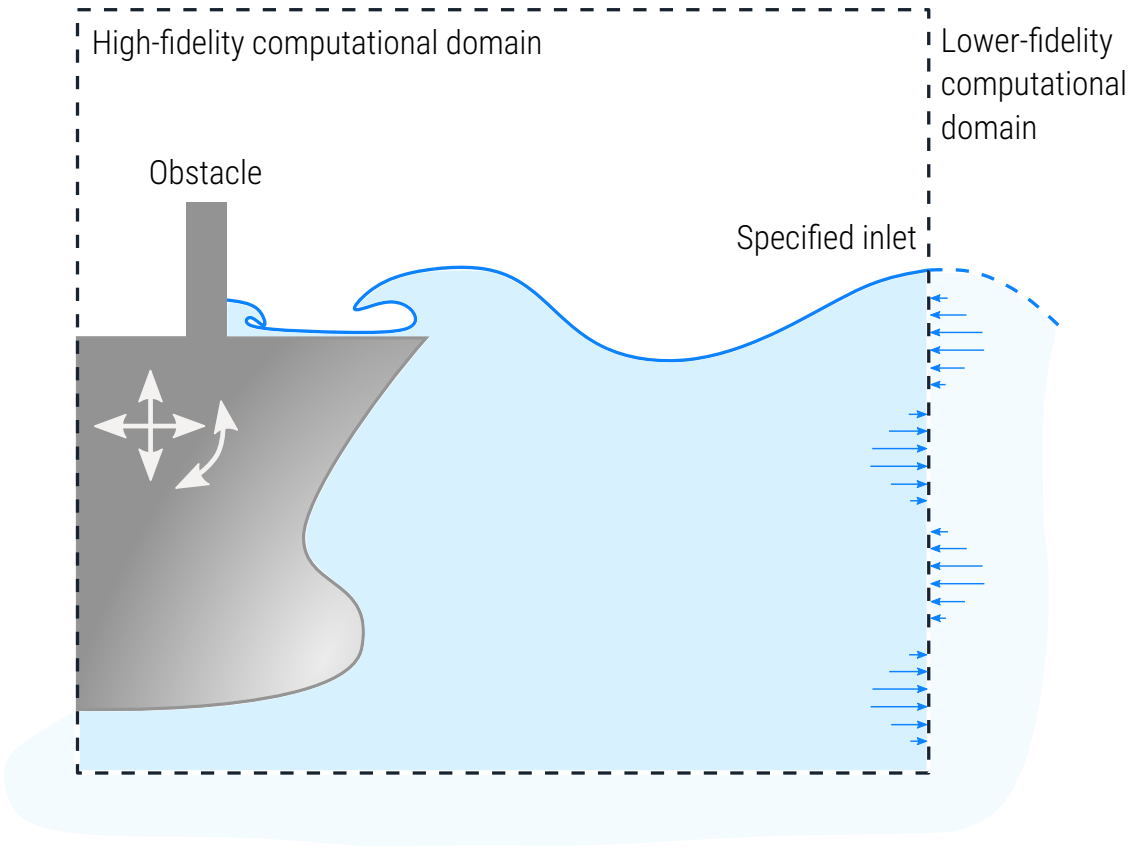

Figure 2.2.: Modelling green water flow by coupling two domains that overlap, where the localised domain takes input from the global domain.

of wave components. Fast calculation of long–term evolution of wave–fields makes the HOS methods attractive for coupling with higher–fidelity localised solvers that evaluate wave–structure interaction.

2.7.2. High–Fidelity Domain

The second domain, used for the higher–fidelity simulation can be of small size, i.e. encircling only the area of interest while imposing the flow calculated by the lower–fidelity solver for the large domain. It is still debatable how small this domain that contains the area of wave–structure interaction can be made, while still providing reliable output due to the linearised input.

In modern Eulerian CFD methods, the overset grid technique is used to allow rigid body motion in 6–DOF free floating–body simulations [44]. In other words, the high–fidelity domain composes of domains moving inside a background domain. It is important to enforce conservation properties between the sub-domains and the background domain. A very important advantage of Lagrangian methods, as the one introduced in this thesis, is that by their definition they do not require overset grids. The whole fluid body and solid bodies in the simulations are physically moving in time, as compared to Eulerian CFD

methods which need to unite flux–based flow with an object movement.

As the performance of computing is getting better, engineers are extending the size of domains of interest for high–fidelity unsteady simulations. The final goal is to create tools that will simulate vessel motions on nonlinear wave–fields in a reasonable amount of time. This thesis deals with providing the proofs of the novel CFD concept using the approach rendered in figure 2.2, but the high–fidelity domain may be extended without constraints to enable computation of vessel motions on highly nonlinear wave–fields, e.g. evolved by the HOS method.

2.7.3. Communication Between Domains

An important ingredient to the success of the spatially decomposed simulation is the communication between the domains that is taking effect through boundary conditions. If the domains are overlapping, the lower–fidelity domain shares its information with the meshless domain. To couple high–fidelity solvers with some linearised or HOS solver, the latter must provide the velocity field, $\boldsymbol{u}_{lo.fi.}(\boldsymbol{x}, t)$, and free surface elevation information, $\eta_{lo.fi.}(x, y, t)$. This information is imposed at the boundaries of the high–fidelity domain, shown in figure 2.2. One may wonder what happens when the wave–structure interaction strongly affects the flow in a way that it reaches the boundaries by diffraction, radiation or by repulsing wave impacts. To prevent the unwanted effects, the high–fidelity domain is either made larger or relaxation zones are employed. In chapter 3, the numerical solution to the domain decomposition problem will be described in detail. The described approach is sometimes referred to as the one–way coupling, since the flow at boundaries of the meshless domain is imposed by taking relevant information from another mesh–based (or even meshless–based) solver, but the information from the meshless domain is not shared back. This is an adequate approach when investigating local events, such as water on deck, providing that all relevant nonlinearities are taken into account.

It should be noted that two–way coupling is not complex to implement, although it is not crucial for the set of problems this thesis investigates. Marrone *et al.* [116] showed that an explicit Lagrangian solver, based on the weakly compressible SPH method, can be reliably coupled to a classical FVM solver. The FVM solver was used to resolve the bulk flow while the SPH solver captured flow near the free surface region. Simply termed, the two domains have an overlapping zone in which the pressure and velocity are blended. Since the space and temporal discretisation differ, interpolation in space and time is used to obtain the hydrodynamic variables and share them between the domains in weakly coupled manner. The consistent meshless boundary conditions, interpolation and relaxation zone technique described in chapter 3 qualify the introduced methodology to be weakly coupled in a two–way manner with another mesh–based solver.

2.8. Vessel Motions

Since the two domains described above are overlapping, the motions of the floating body may be calculated by the lower– or higher–fidelity solver. A floating body moving due to hydrostatic and hydrodynamic forces is considered to be rigid. The rigid body assumption does not consider forces acting between individual elements of mass. The second assumption eliminates forces due to the Earth's motion relative to a star–fixed inertial reference system. The forces on a marine craft due to the Earth's rotation of $7.2921 \cdot 10^{-5}$ rad/s are small compared to the hydrodynamic forces [117].

2.8.1. Rigid Body Kinetics

Newton's second law relates the force, object mass and its acceleration:

$$\boldsymbol{F} = m\,\boldsymbol{a}, \tag{2.52}$$

where $\boldsymbol{F}$ is the resultant force acting on the body centre of gravity (COG), m is the body mass, and $\boldsymbol{a}$ is the acceleration vector measured at COG of the rigid body. Newton's second law can be expressed in terms of conservation of linear momentum and angular momentum. The rotational dynamics following Euler's second law states:

$$\boldsymbol{Q} = \mathbf{I}_{COG}\,\boldsymbol{\alpha} + \boldsymbol{\omega} \times \mathbf{I}_{COG}\,\boldsymbol{\omega}, \tag{2.53}$$

where $\boldsymbol{Q}$ is the resultant torque acting on the body, $\mathbf{I}_{COG}$ is the tensor of the moment of inertia about the COG, $\boldsymbol{\alpha}$ is the angular acceleration vector, and $\boldsymbol{\omega}$ is the angular velocity vector. $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ are considered for a body coordinate axes. The moment of inertia tensor about COG is defined as:

$$\mathbf{I}_{COG} = \begin{bmatrix} \mathrm{I}_x & -\mathrm{I}_{xy} & -\mathrm{I}_{xz} \\ -\mathrm{I}_{yx} & \mathrm{I}_y & -\mathrm{I}_{yz} \\ -\mathrm{I}_{zx} & -\mathrm{I}_{zy} & \mathrm{I}_z \end{bmatrix},$$

which is a symmetric tensor filled by values of the products of inertia about chosen body axes. The tensor is constant in the body frame, not in an inertial frame. Therefore, the second term in equation (2.53); zero torque $\boldsymbol{Q}$ does not imply constant angular velocity $\boldsymbol{\omega}$. It is generally possible to find some orientation in which the products of inertia are zero and the tensor is diagonal.

2.8.2. Fluid–Structure Coupling

Due to efficiency, motions of floating objects are often calculated by a potential flow solver. It is straightforward to impose the calculated motions in the simulation by moving boundary surfaces representing the body. The no-slip boundary condition, equations (2.16) and (2.17), is imposed through equation (2.23) in the PPE on each boundary point placed on the body surface. Since the location of the boundary point does not coincide with the body COG, the velocity and acceleration at that location has to be corrected due to the body rotation [117]. Therefore, the velocity vector of a point $\boldsymbol{x}$ moving with the rigid body is defined as:

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}_{COG} + \boldsymbol{\omega} \times (\boldsymbol{x} - \boldsymbol{x}_{COG}), \tag{2.54}$$

where $\boldsymbol{g}_{COG}$ is the velocity vector at the body COG, and $\boldsymbol{x}_{COG}$ is the global coordinate of the body COG. The acceleration vector of a point $\boldsymbol{x}$ moving with the rigid body is given as:

$$\frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial t} = \frac{\partial \boldsymbol{g}_{COG}}{\partial t} + \boldsymbol{\alpha} \times (\boldsymbol{x} - \boldsymbol{x}_{COG}) + \boldsymbol{\omega} \times [\boldsymbol{\omega} \times (\boldsymbol{x} - \boldsymbol{x}_{COG})]. \tag{2.55}$$

Equation (2.55) is used in equation (2.23) to impose equation (2.54) within the fluid at solid boundaries. Equation (2.54) is used when calculating spatial derivatives of the fluid points near rigid walls.

One–way coupling implies that the fluid flow conforms to the imposed body motion through the fluid. The two–way coupling of disturbed water and a moving vessel requires the integration of pressure and shear stress on the wetted surface to obtain the fluid force acting on the body. The integration of some function along the surface described by meshless points is not straightforward, because the surface area around a boundary meshless point in not strictly defined. An option for obtaining the relevant surface area is to either perform the Delaunay triangulation to obtain the discrete surface or to use the underlying triangulated boundary surface to perform quadrature. Therefore, the added implementation complexity is currently avoided by calculating the motions using a potential flow solver.

3. Numerical Methodology

3.1. Interpolation

A truly mesh-free system is represented only by a point cloud that contains a set of points Ω, and does not incorporate any topological information. Each point in the cloud $i \in \Omega$ carries some system properties in a specific time instant t at its coordinate $\boldsymbol{x}_i(t)$. The points are classically moved along the characteristic curves of the field $\boldsymbol{v}$. Due to a lack of topology, the differential volume $\mathrm{d}V$ at any coordinate $\boldsymbol{x}$ in the continuous domain must be fractionally partitioned among a set of near points $\mathcal{N}$, usually referred to as neighbour points. Hence it is necessary to introduce a continuous and symmetric distance–based weighting function $W(r,\, h)$ for the interpolation points. The weighting function, which is also known as the smoothing kernel function, is compactly supported based on the chosen smoothing radius h, i.e. $W = 0$ for $r \geq \kappa h$. A fraction $\psi_i(\boldsymbol{x})$ of a differential volume $\mathrm{d}V$ is associated with an interpolation point i around an arbitrary location in the following way:

$$\psi_i(\boldsymbol{x}) = \frac{1}{\omega(\boldsymbol{x})} W\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|, h(\boldsymbol{x})\right), \tag{3.1}$$

where the term ω is used to normalise the weights:

$$\omega(\boldsymbol{x}) = \sum_{i \in \mathcal{N}} W\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|, h(\boldsymbol{x})\right), \tag{3.2}$$

and guarantees that the sum of the weights equals unity:

$$\sum_{i \in \mathcal{N}} \psi_i(\boldsymbol{x}) = 1, \tag{3.3}$$

which makes absolute normalisation of the weighting function irrelevant. Equation (3.1) that satisfies the partition of unity equation (3.3) reproduces a constant, and is traditionally referred to as Shepard's interpolant. The interpolant can be derived from the Moving Least Squares (MLS) using only the constant in the monomial basis, and thus MLS techniques can be used to obtain higher order interpolation [73]. Shepard's volume partitioning resembles Voronoi tessellation with smooth transition over edges of cells,

which consequently avoids discontinuities when the point cloud deforms. From equation (3.1) it follows that a weighting function can be designed in such a way that all the weight is attributed to the nearest interpolation point, which would exactly recover Voronoi tessellation [118, 119]. Two neighbour cells in Voronoi tessellation are divided by a perpendicular bisector of the line segment that joins two cell centres, in the meshless case the interpolation points, consequently producing convex–only cell shapes. It might also be observed that this is a desirable feature when applying the divergence theorem, which is based on the normal vector of the edge of a control volume [119]. Finally, interpolation of a function $f(\boldsymbol{x})$ at an arbitrary coordinate $\boldsymbol{x}$ uses discrete function values at the neighbour points in the support domain, which can be written as:

$$\langle f(\boldsymbol{x})\rangle = \sum_{i\in\mathcal{N}} \psi_i(\boldsymbol{x})\, f(\boldsymbol{x}_i)\,, \tag{3.4}$$

where the term $f(\boldsymbol{x}_i)$ describes the discrete value at the interpolation point i. It should be noted that equation (3.4) is more appropriately depicted as an approximation instead of an interpolation expression, since the weighting function does not reach infinity at the origin compared to the traditional usage of inverse distance weighting. Equation (3.4) converges to the true value of $f(\boldsymbol{x})$, when the average distance to near neighbours tends to zero. The standard SPH formulation of a field interpolation does not use a normalisation, and does not guarantee zero–order consistency (the exact reconstruction of a constant field). Unlike in the SPH method where the smoothing function must meet the requirements of the quadrature, here a smoothing function is used for an interpolation weighting and thus can be of any shape. The comparison of equation (3.4) and cell–based and SPH interpolation is shown in figure 3.1.

One should be careful with the meshless interpolation, because a compact radius that is too large or the shape of the interpolation weighting function that is too blunt can lead to excessive numerical diffusion. Generally, a compact sphere should include the nearest neighbour points, and the weighting function should monotonically decrease with the increase in the distance. Moreover, additional weights of interpolation points, such as a particle's mass, can be incorporated by including a multiplication term along with the weighting function $W(r,\, h)$ inside equations (3.1) and (3.2). Popular weighting functions are listed and described in Appendix C.

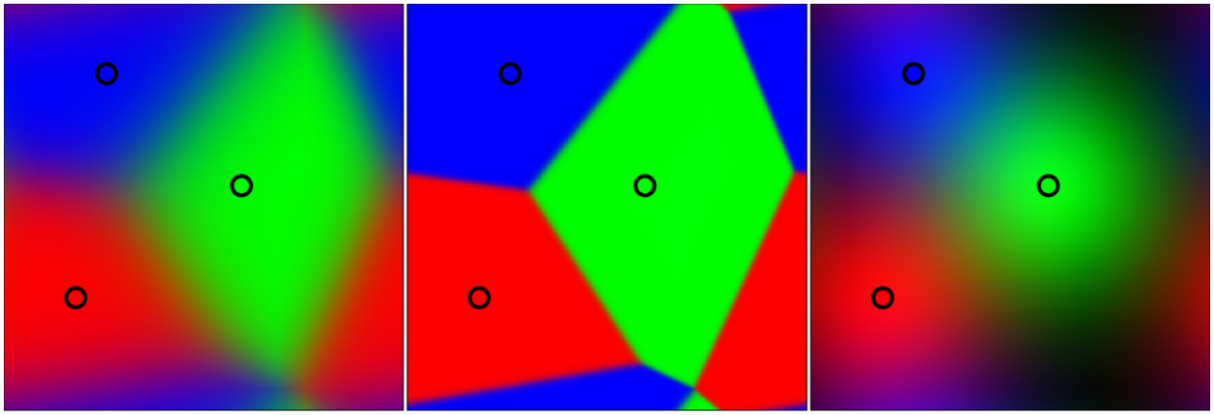

Figure 3.1.: Types of interpolation: Shepard's (left image), cell–based (centre image), SPH (right image), taken from [119].

3.2. Hamilton operator

3.2.1. Gradient

Generally, approximate spatial derivatives of a function $f(\boldsymbol{x})$ should be derived using the known discrete values. In practice, at least a linearly consistent discrete gradient is desirable and this can be obtained by the MLS interpolation or by the LS method that adjusts a lower order polynomial through the data in a best–fit manner. The approximation starts from the Taylor series expansion, which is a representation of a function with an infinite sum of terms that are calculated from the values of the function's derivatives at a single point:

$$f_j = f_i + \boldsymbol{x}_{ij} \cdot \nabla f_i + \frac{1}{2} \left(\boldsymbol{x}_{ij} \nabla\right)^2 f_i + \ldots + R_n, \tag{3.5}$$

where the simplifications are introduced as $f_i \equiv f(\boldsymbol{x}_i)$, $f_j \equiv f(\boldsymbol{x}_j)$, $\boldsymbol{x}_{ij} \equiv \boldsymbol{x}_j - \boldsymbol{x}_i$, and R_n is the Lagrange remainder term. The exact content of Taylor's theorem is not universally agreed on, but mostly any smooth and continuous function can be approximated via equation (3.5). If the first two terms from the right-hand-side of equation (3.5) are kept for the analysis of a linear approximation, the expression can be rearranged as follows:

$$\nabla f_i \cdot \boldsymbol{x}_{ij} = f_{ij} - R_1, \tag{3.6}$$

where $f_{ij} = f_j - f_i$. By multiplying equation (3.6) by (3.1) and $\boldsymbol{x}_{ij}$, the gradient operator at point i with linear accuracy is given by $\mathcal{N}$ equations at node i:

$$\psi_{ij} \boldsymbol{x}_{ij} \left(\boldsymbol{x}_{ij} \cdot \nabla f_i\right) = \psi_{ij} \boldsymbol{x}_{ij} f_{ij} - \psi_{ij} \boldsymbol{x}_{ij} R_1, \; j = 1, \ldots, \mathcal{N}, \tag{3.7}$$

where $\psi_{ij} \equiv \psi_i(\boldsymbol{x}_j)$. The system described by equation (3.6) is overdetermined for a sufficient number of neighbour points $\mathcal{N} > d$, and the solution can be found by a best–fit in the LS sense [120, 121]. Applying the sum on equation (3.7) for neighbour interpolation points yields:

$$\left(\sum_{j\in\mathcal{N}} \psi_{ij}\boldsymbol{x}_{ij}\boldsymbol{x}_{ij}^T\right)\nabla f_i = \sum_{j\in N} \psi_{ij}\boldsymbol{x}_{ij}f_{ij} - \sum_{j\in N} \psi_{ij}\boldsymbol{x}_{ij}R_1, \tag{3.8}$$

where ∇f_i is taken out of the sum, because the sum is valid for $j \neq i$. The LS problem described by the system (3.8) is solved by multiplication by the inverse of the tensor expressed by the sum on the left-hand side. Let $\mathbf{B}_i$ be the so-called renormalisation tensor, defined as:

$$\mathbf{B}_i = \left(\sum_{j\in\mathcal{N}} \psi_{ij}\boldsymbol{x}_{ij}\boldsymbol{x}_{ij}^T\right)^{-1}. \tag{3.9}$$

From equations (3.8) and (3.9), it follows that the discrete gradient operator of an arbitrary scalar function at some coordinate $\boldsymbol{x}_i$ is given by the following expression:

$$\langle\nabla f\rangle_i = \mathbf{B}_i \sum_{j\in\mathcal{N}} \psi_{ij} f_{ij}\boldsymbol{x}_{ij}, \tag{3.10}$$

where the error of equation (3.10) equals:

$$\boldsymbol{\varepsilon}_{\langle\nabla\rangle} = \mathbf{B}_i \sum_{j\in\mathcal{N}} \psi_{ij}\boldsymbol{x}_{ij}R_{1,ij}, \tag{3.11}$$

which includes the missing Taylor series terms and the irregularity of the neighbour points. Let $\boldsymbol{o}_i$ be the offset vector of the i–th point, defined as:

$$\boldsymbol{o}_i = \sum_{j\in\mathcal{N}} \psi_{ij}\boldsymbol{x}_{ij}, \tag{3.12}$$

which points from the location of the point i to the point where the neighbourhood point distribution dominates. Now it can be written that the approximation error equals:

$$\boldsymbol{\varepsilon}_{\langle\nabla\rangle} = C_i \cdot \mathbf{B}_i\boldsymbol{o}_i, \tag{3.13}$$

where C_i is the error accumulated from the terms left out of the Taylor series. The offset vector $\boldsymbol{o}_i$ is a null vector for a point surrounded by regularly positioned neighbours that encircle it. In such a case, the error of equation (3.10) depends only on the spacing of the neighbour points and is analogous to the error of the central difference scheme applied to a regular FD stencil. The size of the "stencil" in a mesh-free method is controlled by the compact radius value of the weighting function.

The renormalisation tensor $\mathbf{B}_i$ depends only on the placement of the neighbour points, and

thus can be precomputed for each point before evaluating equation (3.10). The calculation of the renormalisation tensor $\mathbf{B}_i$ requires the computation effort of an analytical inversion of a small symmetrical tensor with size $d \times d$, which has a memory footprint of 3 scalars in two dimensions, or 6 scalars in three dimensions. This method can be used to calculate approximate spatial derivatives on a highly irregular distribution of points where the ill-conditioned inversion of the tensor can occur for some poor spatial arrangements, e.g. where weighting points are extremely close to each other or aligned in a nearly co-linear or planar manner [122]. These situations can be treated straightforwardly, either by enlarging the radius of the support domain to reach other scattered points, or by falling back to other kinds of discrete gradient operators for the problematic points (e.g. SPH, FV or FD depending on the solution methodology). On the other hand, a compact radius that is too large on a scattered point arrangement introduces some numerical diffusion and discrepancies in accuracy between the nearby grid points. In such a case, the compact sphere which occupies a nonlinear part of a function includes the weights of undesirable and relatively distant neighbour points. Readers interested in the rigorous mathematical derivation and convergence theorems are referred to the original papers [120, 122, 123]. When a meshless method is used in the context of a Godunov scheme, the available information is usually an inter–particle flux evaluated at the centroid of the overlapping region and then multiplied by the effective area of two particles to achieve second–order accuracy [124]. Based on this idea, a class of mesh-free FV methods which are both high–order consistent (convergent) and fully conservative have been developed [122, 123, 118, 119]. Theoretical analysis based on a Taylor series expansion for regular and irregular particle distributions in [125] has shown that this renormalisation technique is the proper way of calculating the gradient for a SPH method. Furthermore, equation (3.10) does not incorporate a smoothing function gradient, but only a smoothing function similar to that in the KGF–SPH method.

3.2.2. Directional Derivative

The rate of change of a function f in the direction $\boldsymbol{n}$ is called the directional derivative, $\partial f / \partial \boldsymbol{n}$. Formally, the directional derivative is the dot product of the gradient and the directional vector:

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{n}} \equiv \boldsymbol{n} \cdot \nabla f(\boldsymbol{x}), \tag{3.14}$$

where $\boldsymbol{n}$ is assumed to be a unit directional vector. Directional derivatives are needed at boundaries, where the Neumann boundary condition imposes a specific value of the first derivative along the boundary normal. The accuracy of the solution to a IBVP is tightly connected to the consistency and robustness of the discrete directional derivative used to impose the Neumann boundary condition.

Assume that the point i is located at the boundary Γ. Straightforward discretisation of equation (3.14) by procedure presented in section §3.2.1 yields:

$$\langle \boldsymbol{n} \cdot \nabla f \rangle_i = \boldsymbol{n} \cdot \mathbf{B}_i \sum_{j \in \mathcal{N}} \psi_{ij} f_{ij} \boldsymbol{x}_{ij}, \tag{3.15}$$

which is equivalent to:

$$\langle \boldsymbol{n} \cdot \nabla f \rangle_i = (\mathbf{B}_i \boldsymbol{n}) \cdot \sum_{j \in \mathcal{N}} \psi_{ij} f_{ij} \boldsymbol{x}_{ij}. \tag{3.16}$$

The linear fitting procedure shown in section §3.2.1 at the edge of a point cloud will moderately miscalculate the gradient of nonlinear functions due to void, i.e. missing neighbour points on one side, which would normally contribute to the fitting procedure. An interpretation would be that the generalised central FD expression is reduced to a generalised one–sided FD expression. Therefore, a study on improving the directional derivative at the edge of a point cloud was needed.

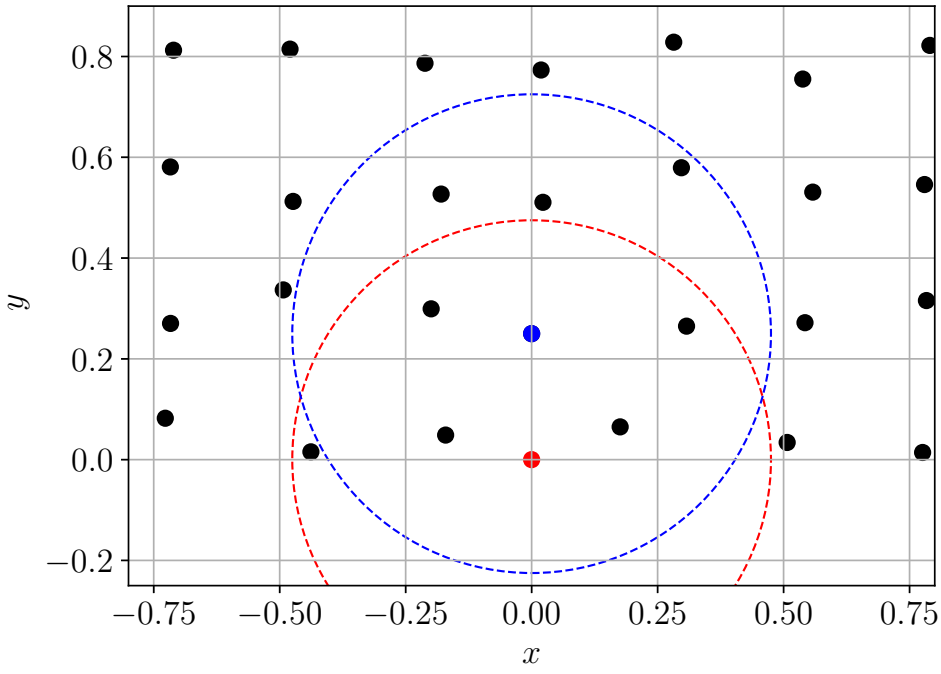

Figure 3.2.: An example of irregular point arrangement near a boundary. The red point and circle renders the boundary point and its compact circle, the blue point and circle renders its parent fluid point and compact circle, while other black points are possible neighbours.

Many meshless numerical methods prepare points that describe boundaries before the simulation [126]. In such cases, points describing fluid do not necessarily lie on lines normal to the boundary points. Consequently, straightforward usage of the discrete gradient is needed to approximate equation (3.14). As an alternative, some meshless methods mirror fluid points along the boundary which keeps them perpendicular [127, 128]. The numerical

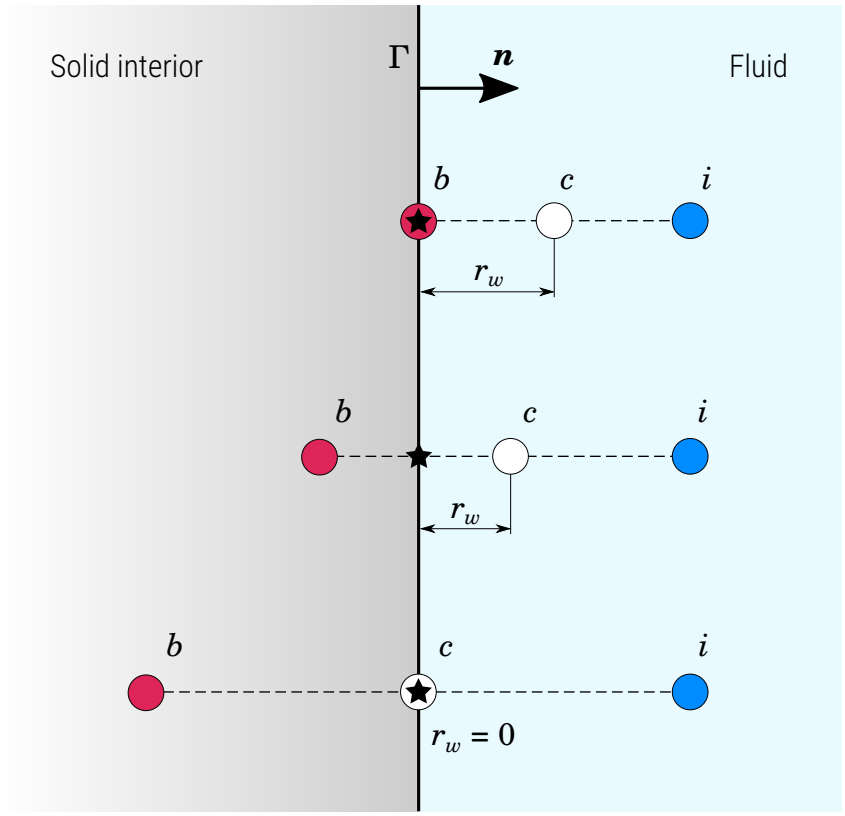

Figure 3.3.: The relationship of the boundary point b and its parent fluid point i. The directional derivative at the boundary $\star$ is derived with the help of virtual average point c.

method introduced in this thesis generates appropriate boundary points each time step by projection onto/inside near boundary surfaces. This grants the method a valuable feature that there always exists one fluid point near the boundary, which by definition lies along the normal of the generated boundary point. This is explained in detail in section §3.4. An example of an irregular neighbourhood distribution around the boundary point with an adjacent point positioned along the normal is shown in figure 3.2.

Figure 3.3 shows how a generated boundary point b can interact with its parent fluid point i. The directional derivative equation (3.14) needs to be evaluated exactly at the boundary Γ, at points marked as $\star$ in the figure. The one–sided FD between points b and i is exact only for linear functions $f(\boldsymbol{x})$. Since the solution to a Poisson equation is crucial for the success of the numerical method, one higher order of accuracy is needed. In other words, FD between points b and i yields the central FD expression for the point c shown in figure 3.3:

$$\langle \boldsymbol{n} \cdot \nabla f \rangle_c = \frac{1}{\|\boldsymbol{x}_{ib}\|} (f_i - f_b) , \tag{3.17}$$

where the point c is located at the middle of line segment connecting the points i and b, $\boldsymbol{x}_c = (\boldsymbol{x}_i + \boldsymbol{x}_b)/2$.

There are now two options how to obtain the directional derivative at the boundary by

the help of imaginary point c with known directional derivative at its location. The first one is the linear extrapolation:

$$\langle \boldsymbol{n} \cdot \nabla f \rangle_\star = \left(1 + 2\frac{r_w}{\|\boldsymbol{x}_{ib}\|}\right) \langle \boldsymbol{n} \cdot \nabla f \rangle_c - \left(2\frac{r_w}{\|\boldsymbol{x}_{ib}\|}\right) \langle \boldsymbol{n} \cdot \nabla f \rangle_i \,, \tag{3.18}$$

where r_w is the distance from the point c to the boundary, and the term $\langle \boldsymbol{n} \cdot \nabla f \rangle_i$ is obtained by equation (3.16) at $\boldsymbol{x}_i$, because the fluid point i is encircled by other fluid and boundary points to yield second–order accurate approximation.

Another option is to expand the value of directional derivative in Taylor series, which gives:

$$\langle \boldsymbol{n} \cdot \nabla f \rangle_\star = \langle \boldsymbol{n} \cdot \nabla f \rangle_c - \lambda_{L,i}\, r_w \left\langle \nabla^2 f \right\rangle_i \,, \tag{3.19}$$

where $\langle \nabla^2 f \rangle_i$ is the discrete Laplacian evaluated at the point i location, whose derivation is shown in section §3.3. Here the expansion was done about the location $\boldsymbol{x}_c$ instead of $\boldsymbol{x}_i$, where the second order accuracy is guaranteed and not dependent on the function change in the direction tangential to the boundary. Since the Laplacian cannot be derived at the location $\boldsymbol{x}_c$ that is closer to the boundary, $\langle \nabla^2 f \rangle_i$ is used with the multiplication term $0 < \lambda_L \leq 1$ that accounts for nonlinearity of the Laplacian along the normal.

Figure 3.4 shows four simple tests made on random point distributions, like the one shown in figure 3.2. First thing noticeable in the figure is the discrepancy of equation (3.15). The regular gradient expression evaluated at boundary is first–order accurate or worse if it is contaminated by tangential nonlinearities. In the best case it reproduces the results of one–sided FDs. Therefore using it as a boundary condition in the system of linear equation does not make much sense. The equation (3.18) that linearly extrapolates from known values at $\boldsymbol{x}_i$ and $\boldsymbol{x}_c$ is second–order accurate, since both $\langle \boldsymbol{n} \cdot \nabla f \rangle_c$ and $\langle \boldsymbol{n} \cdot \nabla f \rangle_i$ are second–order accurate. Taylor–based extrapolation equation (3.19) yields similar results to the linear interpolation, and has a tuning parameter to improve its results. If it is used for a boundary condition of the Poisson equation, it is a convenient option considering the consistency with the Laplacian. On the other hand, the Taylor–based extrapolation mixes one-dimensional and d-dimensional operators. Bearing in mind that discrete operators evaluated around the fluid point i are sensitive to the change of f in the direction perpendicular to $\boldsymbol{n}$, equation (3.19) may result in overshooting if it is not controlled by the parameter λ_L. In conclusion, linear or Taylor–based extrapolation should be used in lieu of direct evaluation of the WLS operator at boundary locations.

3.2.3. Divergence and Curl

Physically the divergence of a two- or three-dimensional vector field is the extent to which the vector field behaves like a source at a given location. If the divergence is not

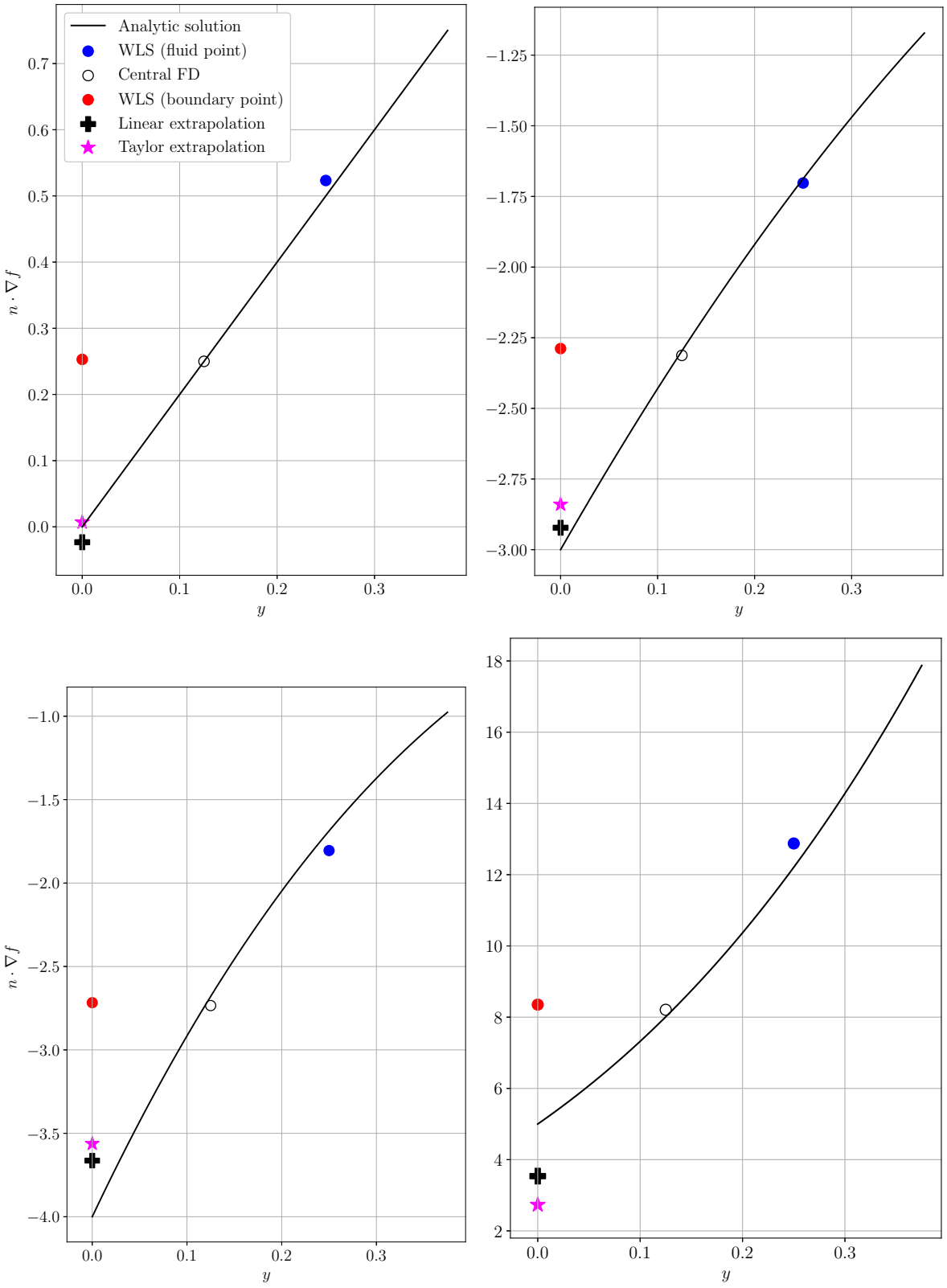

Figure 3.4.: Test of approximating the directional derivative along the boundary normal by various techniques. Scalar fields are functions of y^2, y^3, y^4, and y^5, respectively.

zero at the location, then there must be a source or sink at the location. Consequently, the *incompressibility* of fluid is enforced by constraining the velocity divergence to zero, equation (2.2). By definition the divergence a vector field is the scalar–valued function obtained as the sum of partial derivatives of the field components:

$$\nabla \cdot \boldsymbol{f} = \frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y} + \frac{\partial f_z}{\partial z}, \tag{3.20}$$

where subscripts indicate components of the vector field. The equivalent expression written in the gradient form:

$$\nabla \cdot \boldsymbol{f} = \boldsymbol{e}_x \cdot \nabla f_x + \boldsymbol{e}_y \cdot \nabla f_y + \boldsymbol{e}_z \cdot \nabla f_z, \tag{3.21}$$

where $\boldsymbol{e}_x$, $\boldsymbol{e}_y$ and $\boldsymbol{e}_z$ are unit vectors of a Cartesian system in three-dimensional Euclidean space. If each term from equation (3.21) is substituted by equation (3.16) it follows:

$$\langle \nabla \cdot \boldsymbol{f} \rangle_i = \sum_{j \in \mathcal{N}} \psi_{ij} \mathbf{B}_i \boldsymbol{x}_{ij} \cdot \left(f_{x,ij} \boldsymbol{e}_x + f_{y,ij} \boldsymbol{e}_y + f_{z,ij} \boldsymbol{e}_z \right), \tag{3.22}$$

that reduces to:

$$\langle \nabla \cdot \boldsymbol{f} \rangle_i = \sum_{j \in \mathcal{N}} \psi_{ij} \mathbf{B}_i \boldsymbol{x}_{ij} \cdot \boldsymbol{f}_{ij}, \tag{3.23}$$

The curl is a local property at a point which defines the extent of a rotation around that point. If the curl operator is applied to the velocity vector field, it describes the vorticity of the flow. The curl of a vector function at some location yields a three-dimensional vector that points along the axis of the rotation and whose length corresponds to the speed of the rotation. The curl of a in-plane flow is perpendicular to the plane. The spatial operator is defined as:

$$\nabla \times \boldsymbol{f} = \left\{ \frac{\partial f_z}{\partial y} - \frac{\partial f_y}{\partial z}, \frac{\partial f_x}{\partial z} - \frac{\partial f_z}{\partial x}, \frac{\partial f_y}{\partial x} - \frac{\partial f_x}{\partial y} \right\}, \tag{3.24}$$

and with mathematical manipulations analogous to the above description, the discrete version of the operator is given as follows:

$$\langle \nabla \times \boldsymbol{f} \rangle_i = \sum_{j \in \mathcal{N}} \psi_{ij} \mathbf{B}_i \boldsymbol{x}_{ij} \times \boldsymbol{f}_{ij}. \tag{3.25}$$

By analysing equations (3.10), (3.23) and (3.25), the term $\mathbf{B}_i \boldsymbol{x}_{ij}$ can be interpreted as a part of the Hamilton (nabla or del) operator, ∇. It is a "part" in a way that it forms the complete operator by summing separate neighbour weights $\forall\, j \in \mathcal{N}$. As in the formal definition, that term in combination with a specific product operator defines the gradient, divergence and curl spatial operators. Hence, second–order first spatial operators are

introduced in the WLS context, based on the linear terms from the Taylor series.

3.3. Laplace Operator

3.3.1. State of the Art

Reliable and accurate approximations of a gradient and a Laplacian are needed when solving various PDEs described by strong formulations: Navier–Stokes equations, the diffusion equation, the Helmholtz equation, the Schrödinger equation, the wave equation, and, of course, other general Laplace and Poisson equations. The Laplace operator, or the Laplacian, is defined as the divergence of the function gradient in Euclidean space, which is equivalent to the sum of unmixed second partial derivatives. The accuracy of discrete spatial operators is not the same on irregular grids as it is for corresponding operators on structured grids, i.e. the accuracy diminishes with an increase of mesh irregularity. In the finite volume (FV) method, the Laplacian used to model viscous forces is obtained by applying the Green–Gauss theorem, often with an explicit artificial dissipation added at each time step. De Foy and Dawes [129] have shown that the repeated action of a FV Laplace operator could lead to zero–order accuracy, with errors up to 50% on irregular meshes. In addition, the positivity of coefficients is not even satisfied on regular grids, which leads to a lack of robustness when the operator is used for smoothing or solving the Poisson equation. Juretic *et al.* [130] have shown that the accuracy of the diffusion term is dependent on the interpolation scheme for skewed cells' faces, and on the discretisation procedure for surface normal gradients. The Laplacian in non-continuous form cannot guarantee the "divergence of gradient" identity, and Owen *et al.* [131] have shown that the discrete Laplacian should be used to obtain a stable velocity correction scheme that depends on the pressure Poisson equation.

Brookshaw [132] and Monaghan and Gingold [133] have introduced smoothing kernel gradient–based SPH discretisations of the Laplacian, which still today are one of the most widely used formulas, although they are inconsistent and should be corrected [126, 134, 125]. Chaussonnet *et al.* [135] have shown how sensitive the accuracy of typically used SPH Laplacian schemes can be to the particle disorder and smoothing radius. Fatehi and Manzari [125] have reviewed typically used schemes for the second derivative using theoretical analysis, and have introduced first–order consistent second derivatives based on the renormalisation tensors.

The original Laplacian used in the moving particle semi–implicit (MPS) method was proposed by Koshizuka *et al.* [136]. Isshiki [137] successfully reproduced it via the Gauss divergence theorem. Zhang *et al.* [138] have shown its inconsistencies and the numerical difficulties that emerge when solving the Poisson equation, and introduced the enhanced

version of the operator [139]. Ng *et al.* [140] have evaluated the accuracy of various MPS Laplacian models by solving the Poisson equation subjected to both Dirichlet and Neumann boundary conditions, and proposed a more general model with the altered kernel function. They deduced that refining the grid spacing while retaining the number of neighbour interpolation points is not applicable due to the numerical errors, and, moreover, the optimal number of neighbours is dependent on the degree of grid irregularity. Ikari *et al.* [141] have proposed a corrected higher order Laplacian scheme, which is derived by taking the divergence of the corrected LS gradient model. The resulting correction tensor is used in combination with the first– and second–order derivative of a smoothing kernel function, which produces more accurate results. Ma *et al.* [134] have reviewed the meshless pressure projection procedure by solving the Poisson equation with the most popular discretisations of the Laplacian, often adopted in the incompressible smooth particle hydrodynamics (ISPH) method and moving particle semi–implicit (MPS) methods. Recently, Tamai *et al.* [126] have tested the accuracy and consistency of meshless discretisation schemes for the Laplacian, and employed the studied schemes to find the solution to the Poisson equation in strong formulation with the imposed Dirichlet boundary conditions.

Huang *et al.* [142] have proposed that the Laplacian for the kernel gradient free (KGF) SPH method is obtained by carrying out two first derivative operations, i.e. by means of two inversions of symmetric tensors of size $(d+1) \times (d+1)$, where d is the number of dimensions. The computational cost of the two–pass Laplacian done for each particle is inconvenient for dynamic meshless simulations, and it is difficult to implement boundary conditions. Lei and Peng [143] have proposed an approximate Laplacian model, which is sensitive to particle distribution without using a large smoothing radius, thus requiring frequent particle shifting for dynamic simulations.

In the finite point–set method (FPM) [83], a function and its derivatives are approximated by the second–order moving least squares (MLS) method for each point in a domain. Higher order derivatives are obtained directly since the Hessian coefficients are included while solving the MLS problem. The method proved to be accurate when solving the Poisson equation for incompressible flows [144, 83], although the vector of ten unknowns solved for each point in three dimensions is not computationally efficient for non-static point distribution, e.g. for transient Lagrangian applications.

In the particle strength exchange (PSE) method, the Laplacian is approximated by an integral operator, which is then transformed into the discrete form by a quadrature over the particles [145]. The symmetry in the PSE method can conserve properties that are inherent when two particles "exchange strength" with one another. On the other hand, the PSE method requires that the uniformity of the particle distribution is periodically restored on a well–ordered field, which often requires mesh interpolation [146] or other

correction functions at each time step [145].

The meshless radial basis function finite difference (RBF–FD) method tries to alleviate the high computational cost of global RBF methods by using virtual FD stencils. Bayona *et al.* [147] have noted that the optimal value of the shape parameter at each particle location can be obtained if the value of a function and its derivatives is known, which in practical cases is not true. Finally, Davydov *et al.* [148] have investigated the influence of the shape parameter within the RBF–FD method with irregular centres on the quality of the approximation of the Laplacian, and consequently on the solution of the Poisson equation.

3.3.2. Novel Operators

Based on the referenced literature, it can be deduced that employing corrective tensors derived from a Taylor series expansion is a common approach to guarantee the consistency of the first spatial derivatives. Least–squares methods are computationally more efficient than full–matrix MLS methods, and they are often used with parallel algorithms that drive dynamically changing meshless domains. Accurate and reliable approximations of first and second derivatives are a requirement for a variety of numerical techniques that describe physical phenomena. The aim of the following study is to deduce stable and accurate meshless Laplace operators, based on the least–squares corrective tensors that are already used in many so-called renormalised meshless methods, in order to keep the simplicity and computational efficiency that are needed for dynamic simulations of large meshless systems.

After the second–order accurate discrete gradient operator is defined, the discrete Laplacian operator of a scalar function $\Delta f \equiv \nabla^2 f$ should also be defined. Unfortunately, the operator in a non-continuous form cannot guarantee "divergence of gradient" identity, i.e. it holds for a numerical method that $\nabla^2 f \neq \nabla \cdot (\nabla f)$. Therefore, approximations of the operator are needed in order to avoid solving equation (3.5) with higher–order terms included, which is computationally inefficient for a frequently changing neighbourhood around a considered point. The starting point of the derivation of the discrete Laplacian is again the Taylor series expansion, which includes the first three terms in equation (3.5), and is conveniently written as:

$$f_j = f_i + \boldsymbol{x}_{ij} \cdot \nabla f_i + \frac{1}{2}\boldsymbol{l}_i \cdot \boldsymbol{s}_{ij} + \boldsymbol{m}_i \cdot \boldsymbol{p}_{ij} + R_{2,ij}, \tag{3.26}$$

where $\boldsymbol{l}_i$ is the vector of the second derivatives at $\boldsymbol{x}_i$ (e.g. $\{f_{i,xx},\ f_{i,yy}\}$ for $d = 2$), $\boldsymbol{s}_{ij}$ is the vector containing the squared components of $\boldsymbol{x}_{ij}$, $\boldsymbol{m}_i$ is the vector of the mixed derivatives at $\boldsymbol{x}_i$ (e.g. $\{f_{i,xy}\}$ for $d = 2$), and $\boldsymbol{p}_{ij}$ is the vector of the mixed components of $\boldsymbol{x}_{ij}$ (e.g. $\{x_{ij}y_{ij}\}$ for $d = 2$). Hessian matrix components are decomposed into two

vectors, $\boldsymbol{l}_i$ and $\boldsymbol{m}_i$, which contain second and mixed derivatives, respectively. This way, equation (3.26) can be solved for $\boldsymbol{l}_i$, and the Laplacian is obtained using the following definition:

$$\nabla^2 f_i = \boldsymbol{l}_i \cdot \boldsymbol{I}, \tag{3.27}$$

where $\boldsymbol{I}$ is the identity vector, e.g. for 2D $\boldsymbol{I} = \{1, 1\}$. The terms in equation (3.26) are rearranged to include the second derivatives on the left-hand-side in the following way:

$$\boldsymbol{l}_i \cdot \boldsymbol{s}_{ij} = 2\left(f_{ij} - \boldsymbol{x}_{ij} \cdot \nabla f_i - \boldsymbol{m}_i \cdot \boldsymbol{p}_{ij} - R_{2,ij}\right). \tag{3.28}$$

The following sections will show how equation (3.28) can be used to obtain the Laplacian using neighbour points from the set $\mathcal{N}$.

3.3.2.1. Naive Version

In order to leave only the Laplacian on the left-hand-side, equation (3.28) is multiplied by $\boldsymbol{I} \cdot \boldsymbol{I} / \left\|\boldsymbol{x}_{ij}\right\|^2$ according to equation (3.27):

$$\nabla^2 f_i = \frac{2d}{\left\|\boldsymbol{x}_{ij}\right\|^2}\left(f_{ij} - \boldsymbol{x}_{ij} \cdot \nabla f_i - \boldsymbol{m}_i \cdot \boldsymbol{p}_{ij} - R_{2,ij}\right), \tag{3.29}$$

where d is the consequence of double multiplication by identity vectors, i.e. $\boldsymbol{I} \cdot \boldsymbol{I} = d$. Equivalent to the derivation of the discrete gradient operator described in Section 3.2.1, both sides of equation (3.29) can be summed with the weights of the neighbour interpolation points around i taken into account. In that case, the left-hand-side remains unmodified due to equation (3.3), and the term that contains mixed derivatives is cancelled due to the symmetrical smoothing function property. The discrete Laplacian of an arbitrary scalar function at some coordinate $\boldsymbol{x}_i$ expression is written as:

$$\left\langle \nabla^2 f \right\rangle_i = 2d \sum_{j \in \mathcal{N}} \frac{\psi_{ij}}{\left\|\boldsymbol{x}_{ij}\right\|^2}\left(f_{ij} - \boldsymbol{x}_{ij} \cdot \left\langle \nabla f \right\rangle_i\right), \tag{3.30}$$

where the gradient term is precalculated by equation (3.10). The absolute error of equation (3.30) includes the error of the discrete gradient operator described by equation (3.13), along with the error accounting for the truncated higher–order Taylor series terms:

$$\varepsilon_{\langle \nabla^2 \rangle} = 2d \sum_{j \in \mathcal{N}} \frac{\psi_{ij}}{\left\|\boldsymbol{x}_{ij}\right\|^2}\left(R_{2,ij} + \boldsymbol{x}_{ij} \cdot \boldsymbol{\varepsilon}_{\langle \nabla \rangle}\right). \tag{3.31}$$

The formulation of an approximate Laplacian described by equation (3.30) is called the "naive version" of the operator, since it is expected that the sum of error contributed by the discrete gradient operator and truncated higher–order terms is insignificant. Fur-

thermore, equation (3.30) straightforwardly sums contributions of neighbour particles by relying on Shepard's interpolation. It can be expected that the utmost regular distribution or amortisation by a high number of points in set $\mathcal{N}$ are needed in order to achieve accuracy suitable for reliable numerical simulations. It will be shown in Section 5.1.1 that a proper smoothing function, different from the one used for the gradient, can alleviate the mentioned problems.

3.3.2.2. Sum Version

The naive version of an approximate Laplacian is derived from equation (3.26) with the assumption that the discrete gradient operator described by equation (3.10) yields results with inconsequential discrepancy when compared to the continuous gradient operator. That assumption holds for nearly linear function areas encompassed by the smoothing sphere, but high error would occur otherwise. If the discrete gradient described by equation (3.10) is substituted back into equation (3.26) taking its error into account, the following expression of the discrete Taylor series expansion is obtained:

$$f_j = f_i + \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \sum_{j \in \mathcal{N}} \psi_{ij} \boldsymbol{x}_{ij} \left(f_{ij} - \frac{1}{2} \boldsymbol{l}_i \cdot \boldsymbol{s}_{ij} - R_{2,ij} \right) + \frac{1}{2} \boldsymbol{l}_i \cdot \boldsymbol{s}_{ij} + R_{2,ij}. \tag{3.32}$$

In order to leave terms coupled with the Laplacian on the left-hand-side, equation (3.32) is reorganised into the following form:

$$\boldsymbol{l}_i \cdot \boldsymbol{s}_{ij} - \boldsymbol{x}_{ij} \mathbf{B}_i \sum_{j \in \mathcal{N}} \psi_{ij} \boldsymbol{x}_{ij} \left(\boldsymbol{l}_i \cdot \boldsymbol{s}_{ij} \right) = 2 \left[f_{ij} - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \sum_{j \in N} \psi_{ij} \boldsymbol{x}_{ij} \left(f_{ij} - R_{2,ij} \right) - R_{2,ij} \right]. \tag{3.33}$$

Equation (3.33) is multiplied by $\boldsymbol{I} \cdot \boldsymbol{I}$ in order to obtain the Laplacian term as a scalar. Considering that it holds $\boldsymbol{s}_{ij} \cdot \boldsymbol{I} = \|\boldsymbol{x}_{ij}\|^2$ and equation (3.27), the following expression is obtained:

$$\nabla^2 f_i \left(\|\boldsymbol{x}_{ij}\|^2 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \sum_{j \in N} \psi_{ij} \boldsymbol{x}_{ij} \|\boldsymbol{x}_{ij}\|^2 \right) \approx 2d \left(f_{ij} - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \sum_{j \in N} \psi_{ij} \boldsymbol{x}_{ij} f_{ij} \right), \tag{3.34}$$

where the error terms are left out for the sake of simplicity. Both sides of equation (3.34) can be summed, with the weights of neighbour interpolation points around i taken into account. In that case, the Laplacian will remain unmodified due to equation (3.3), and the sum applied to the parentheses of each side of the equation can be reorganised as follows:

$$\sum_{j \in \mathcal{N}} \psi_{ij} A_{ij} - \left(\sum_{j \in N} \psi_{ij} \boldsymbol{x}_{ij} \right) \cdot \mathbf{B}_i \sum_{k \in N} \psi_{ik} \boldsymbol{x}_{ik} A_{ik} \approx \sum_{j \in N} \psi_{ij} A_{ij} \left(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i \right), \tag{3.35}$$

where A_{ij} is $\|\boldsymbol{x}_{ij}\|^2$ or f_{ij}, which yields the formulation of an approximate Laplacian operator applied to a scalar function at the location of a point i:

$$\left\langle \nabla^2 f \right\rangle_i = 2d \frac{\sum_{j\in\mathcal{N}} \psi_{ij} f_{ij} \left(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i\right)}{\sum_{j\in\mathcal{N}} \psi_{ij} \left\|\boldsymbol{x}_{ij}\right\|^2 \left(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i\right)}, \tag{3.36}$$

where the $\mathbf{B}_i \boldsymbol{o}_i$ term can be pre-calculated for each point i, by equations (3.9) and (3.12). If the offset vector $\boldsymbol{o}_i$ is a null vector (e.g. which holds for a regular distribution of neighbour points), then equation (3.36) reduces to the central difference expression when the nearest points are taken in $\mathcal{N}$. On the other hand, when the offset vector has finite length, then the difference f_{ij} and the squared distance $\|\boldsymbol{x}_{ij}\|^2$ are corrected inside the sum. Hence, the formulation of an approximate Laplacian described by equation (3.36) is called the "sum version" of the discrete operator. It resembles the KGF–SPH form introduced in [142], but it is improved with renormalisation. The scalar product within the sum virtually "shifts" each neighbour relative location, based on the irregularity correction and the weights offset correction of the neighbour point locations. This correction can also be rendered as a cancelling of the second–order gradient error due to the offset vector, for an investigated quantity A_i. Corrected values are summed in Shepard's interpolation manner, and finally the sums are divided. The absolute error of equation (3.36) includes the truncated higher–order Taylor series terms error:

$$\varepsilon_{\langle\nabla^2\rangle} = 2d \frac{\sum_{j\in\mathcal{N}} \psi_{ij} R_{2,ij} \left(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i\right)}{\sum_{j\in\mathcal{N}} \psi_{ij} \left\|\boldsymbol{x}_{ij}\right\|^2 \left(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i\right)}. \tag{3.37}$$

Evidently, equation (3.37) does not contain lower–order error $R_{1,ij}$ as equation (3.31) does, but depends on averaged error collected from the neighbour points. The presented version of the discrete operator is attractive since it is first–order accurate, and does not require the gradient value to be precalculated before evaluating equation (3.36), but only the offset vector $\boldsymbol{o}_i$ that can be done in the same loop as for the renormalisation tensor $\mathbf{B}_i$.

3.3.2.3. Least–Squares Version

In section §3.3.2.2 it was shown how the de-vectorisation of equation (3.33) provided that the final result could be obtained simply by a division of summations. Alternatively, an LS procedure similar to the one described in section §3.2.1 can be applied to elicit more accurate results. The term $\boldsymbol{l}_i$ on the left-hand-side of equation (3.33) can be taken out of the summation and thus the left-hand-side can be written as $\boldsymbol{l}_i \cdot \boldsymbol{q}_{ij}$, where $\boldsymbol{q}_{ij}$ is the vector defined as:

$$\boldsymbol{q}_{ij} = \boldsymbol{s}_{ij} - \boldsymbol{x}_{ij}^T \sum_{j\in\mathcal{N}} \psi_{ij} \mathbf{B}_i \boldsymbol{x}_{ij} \boldsymbol{s}_{ij}^T. \tag{3.38}$$

If equation (3.33) is multiplied by $\boldsymbol{q}_{ij}$ and afterwards, taking summation with the weights of neighbour points into account, a problem similar to equation (3.8) needs to be solved. The new renormalisation tensor is defined as:

$$\widehat{\mathbf{B}}_i = \left(\sum_{j \in \mathcal{N}} \psi_{ij} \boldsymbol{q}_{ij} \boldsymbol{q}_{ij}^T \right)^{-1}, \tag{3.39}$$

which is used to yield the formulation of an approximate Laplacian operator:

$$\left\langle \nabla^2 f \right\rangle_i = \boldsymbol{I} \cdot \widehat{\mathbf{B}}_i \sum_{j \in \mathcal{N}} 2\psi_{ij} \boldsymbol{q}_{ij} \left(f_{ij} - \boldsymbol{x}_{ij} \cdot \langle \nabla f \rangle_i \right). \tag{3.40}$$

and which will be referred to as the "full inversion version" of the Laplacian. The absolute error of equation (3.40) is:

$$\varepsilon_{\langle \nabla^2 \rangle} = \boldsymbol{I} \cdot \widehat{\mathbf{B}}_i \sum_{j \in \mathcal{N}} 2\psi_{ij} \boldsymbol{q}_{ij} \left(\boldsymbol{x}_{ij} \cdot \mathbf{B}_i \sum_{j \in N} \psi_{ij} \boldsymbol{x}_{ij} R_{2,ij} - R_{2,ij} \right). \tag{3.41}$$

The derivation of equation (3.40) presented here is analogous to one in [126], and similar to [125] in the context of the SPH method. The disadvantage of the presented scheme is its computation inefficiency for dynamically deforming point clouds. Besides an iteration loop over the neighbour points that is needed to compute the renormalisation tensor $\mathbf{B}_i$, an additional loop and memory space is needed to compute and store the summation tensor evaluated in equation (3.38), and finally a loop is needed to evaluate equation (3.39). Instead of manipulating equation (3.33) with the vector $\boldsymbol{q}_{ij}$, the expression is multiplied by the vector $\boldsymbol{s}_{ij}$. By classically doing the summation, the following is obtained:

$$\boldsymbol{l}_i \cdot \left(\sum_{j \in \mathcal{N}} \psi_{ij} \boldsymbol{s}_{ij} \boldsymbol{s}_{ij}^T - \sum_{j \in \mathcal{N}} \psi_{ij} \boldsymbol{x}_{ij} \boldsymbol{s}_{ij}^T \sum_{k \in N} \psi_{ik} \mathbf{B}_i \boldsymbol{x}_{ik} \boldsymbol{s}_{ik}^T \right) \approx \sum_{j \in \mathcal{N}} 2\psi_{ij} \boldsymbol{s}_{ij} \left(f_{ij} - \boldsymbol{x}_{ij} \cdot \langle \nabla f \rangle_i \right), \tag{3.42}$$

where the same tensors are found in the second product of summations on the left-hand-side. If the tensor elements of cubic order are taken as zero due to the symmetrical property of the smoothing function, the renormalisation tensor can be defined as:

$$\widetilde{\mathbf{B}}_i = \left(\sum_{j \in \mathcal{N}} \psi_{ij} \boldsymbol{s}_{ij} \boldsymbol{s}_{ij}^T \right)^{-1}, \tag{3.43}$$

which can be evaluated in the same loop along with $\mathbf{B}_i$. Equation (3.43) is used to yield a more approximate, but also a more computationally efficient, formulation of the Laplacian

operator than equation (3.40) in the following way:

$$\left\langle \nabla^2 f \right\rangle_i = \boldsymbol{I} \cdot \widetilde{\mathbf{B}}_i \sum_{j \in \mathcal{N}} 2\psi_{ij} \boldsymbol{s}_{ij} \left(f_{ij} - \boldsymbol{x}_{ij} \cdot \left\langle \nabla f \right\rangle_i \right), \tag{3.44}$$

which will be referred to as the "basic inversion version" of the Laplacian. Equation (3.44) still requires the gradient of the relevant field to be evaluated before the evaluation of the Laplacian, unlike the sum version of the Laplacian described by equation (3.36). One potential drawback of equations (3.40) and (3.44) is related to the fact that vectors $\boldsymbol{q}_{ij}$ and $\boldsymbol{s}_{ij}$ can contain elements close to zero; floating point constraints can lead to errors when calculating inverse tensors by equations (3.39) and (3.43), and using them within the Laplacian expressions.

3.4. Solid Boundaries

Lagrangian mesh-free set of points deforms freely, it is not topologically bound to any surrounding geometry. Therefore, the geometry can simply be moved in time, and the fluid adjusts to moving boundaries by respecting the imposed boundary conditions, as each point on the moving body also performs Lagrangian motion.

3.4.1. State of the Art

The researchers working with the SPH methods have not found a simple and efficient scheme for boundary conditions, and that problem is listed among the 'Grand Challenges' by the SPH European Research Interest Community [149]. Meshless methods that are derived from the SPH and MPS theories cannot handle points with truncated compact spheres. Hence multiple rows of boundary points are placed on walls and behind them to fill the compact spheres of fluid points near walls.

Some methods transform the geometry into discrete points at the start of the simulation [126], which can be classified as a kind of meshing preprocess. Uniform distributions of points on walls and behind walls (dummy points) on complex surfaces is hard to obtain. These layers of dummy particles are linked to their corresponding wall particles and carrying equal pressure and velocity values. Other mesh-free methods mirror fluid points each time step against the boundary normal until their compact spheres are filled [128, 127]. Mirrored particles have velocities extrapolated from the fluid and wall velocities. There are a few attempts to solve semi-analytically for the non-existent points of the truncated kernel [150, 151, 152], but the method mixes Eulerian and Lagrangian description of the flow and the boundary has to be meshed in an appropriate way.

In contrary to the described problems, GFDMs are based on generalised FD operators, and thus do not require multiple rows of points to be placed behind boundaries [83, 81, 84] as they are immune to incompletely filled compact spheres.

3.4.2. Boundary Points

In this thesis, a scheme for treating wall boundaries is introduced, which is consistent with the other mesh-free Lagrangian parts of the method. Most importantly, the introduced spatial operators work adequately if the point under consideration is encircled by one ring of neighbour points, as explained in section §3.2. It can be deduced that only one array of points is needed on the boundary to impose the boundary conditions, because the nearby fluid points will adequately calculate spatial operators at their locations.

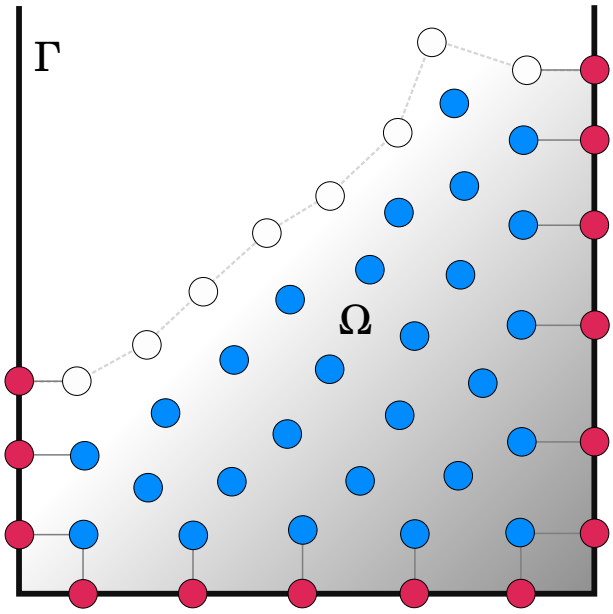

Figure 3.5.: Schematic of points that are used for no–slip (red) and free–surface (white) boundary conditions, and corresponding interior fluid points (blue). The grey gradient denotes an interpolated pressure field.

Secondly, the incompressibility constrains the fluid points to be equidistant, so that each point carries approximately same fluid volume. Consequently, points close to walls can be projected to generate equidistant points as well, where the boundary conditions are going to be imposed. An example of preparing boundary points for a time step is shown in figure 3.5.

Boundaries can be represented in any form for two and three dimensional domains: analytic curves and surfaces, polygonal surface meshes, point clouds, etc. In terms of computational performance, discrete edges and triangles for two- and three-dimensional problems, respectively, are the fastest elementary shapes for solving the projection problem. Since the shapes can describe complex geometries, there is no need for any kind of mesh

preparation before simulating, the only input for a simulation are those discrete boundary surfaces defining the domain and bodies of the simulation.

At the start of a time step, all boundary points from the previous time step are discarded. Fluid points which lie on the edge of the point cloud, and are located near the boundaries, are projected onto those boundaries. New points for the current time step are then generated on the projected locations. Fluid points normally take into account generated boundary points when evaluating spatial operators, i.e. each point i in the point cloud can have neighbours of different types, written as:

$$\mathcal{N} = \mathcal{F} \cup \mathcal{B}, \tag{3.45}$$

where $\mathcal{N}$ is the set of all neighbour points, $\mathcal{F}$ is the subset of fluid neighbour points, $\mathcal{B}$ is the subset of neighbour boundary points.

Instead of generating the boundary point at the projected location, it can be generated by mirroring about the projection location, $\boldsymbol{x}_b$, using the surface normal, $\boldsymbol{n}$. The directional derivative at the wall would be adequately defined by the central FD expression (3.17), because the wall is positioned at the middle of the line segment connecting fluid and generated points. On the other hand, the hydrodynamic values at the generated location (behind the wall) do not correspond to the situation at the wall. To avoid the complications, mirroring of points is not considered in this thesis.

The generated boundary point is assigned the velocity and acceleration of the wall at the corresponding location, which is explained in section 2.8.2. For the mirrored points, those values would have to be extrapolated from the fluid and wall to locations within the solid body.

The significance of solid boundary points is manifested through the PPE defined in section 2.3.1. The pressure gradient that results by imposing the boundary conditions enforces the fluid to align to movement of boundaries. The details on imposing the boundary conditions in the PPE are given in section §3.6.

3.5. Free Surface

A free surface is a boundary between two homogeneous fluids, e.g. liquid water and the air. Unlike liquids, gases cannot form a free surface on their own. In this thesis, the air has not been modelled by meshless points, in order to speed up the computation time of transient simulations. At a specific time instant, the meshless point cloud describes the fluid and boundaries: wall and inflow/open boundary points. Therefore, fluid points that lie on the edge of the point cloud, which are not boundary points, are certainly located on

the free surface, i.e. they discretely form the free surface by imposing suitable boundary conditions along the free surface.

Usually the process classifying such points in meshless methods is named *free surface detection*. The detection of free surface has an important role in methods for free surface flow, because the pressure values on free surface points should be equal to atmospheric pressure by imposing the Dirichlet boundary condition. The solvability of a PPE and the accuracy of its solution greatly depends on the boundary conditions, as described in section §2.4. Specifically, the points on the edge of a fluid point cloud lie on the free surface, where the Dirichlet boundary condition must be imposed so that the PPE has a unique solution.

The free surface detection algorithm consists of two steps, similar to a verification and validation process:

1. the first step classifies points that are potentially on the edge of a point cloud,
2. the second step additionally evaluates those classified points in order to corroborate that they really are located on the free surface.

3.5.1. Point Cloud Edge

The first step of the free–surface detection evaluates each point in the point cloud with a test that proves whether it is inside the point cloud. Inner points are those points which are sufficiently encircled by neighbour points in their compact domains. The test usually takes into account a combination of the following factors: the number of neighbours in the compact sphere ($\sum_{j\in\mathcal{N}} 1$), the sum of neighbour weights ($\sum_{j\in\mathcal{N}} W_{ij}$), and renormalisation properties. Properties of the renormalisation tensor can ease the identification of the points which lie on free surface, e.g. by inspecting its eigenvalues [153]. In this thesis, the properties of the renormalisation tensor are used in a more simple, but robust manner. The first step of the free surface detection makes use of the neighbourhood irregularity correction from the Laplacian expression, i.e. the deficiency of equation (3.36) described in section §3.3, which will be described in the following text.

A free–surface normal vector $\boldsymbol{n}_i$ is pointing away from the fluid, towards the void. At a discrete meshless point, it is approximately turned to the negative direction of the offset vector $\boldsymbol{o}_i$, defined by equation (3.12), which points to where the neighbourhood point distribution dominates. In other words, an edge point has all or most of its neighbours $j \in \mathcal{N}$ located on one side (mostly it holds $\boldsymbol{x}_{ij} \cdot \boldsymbol{o}_i > 0$). Consequently, the sum of the neighbourhood irregularity correction from equation (3.36), $\sum_{j\in\mathcal{N}} (1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i)$, yields a non-dimensional value that is much smaller for an edge point then for an inner point. This value can be used to identify edge points if a proper limiting condition is used.

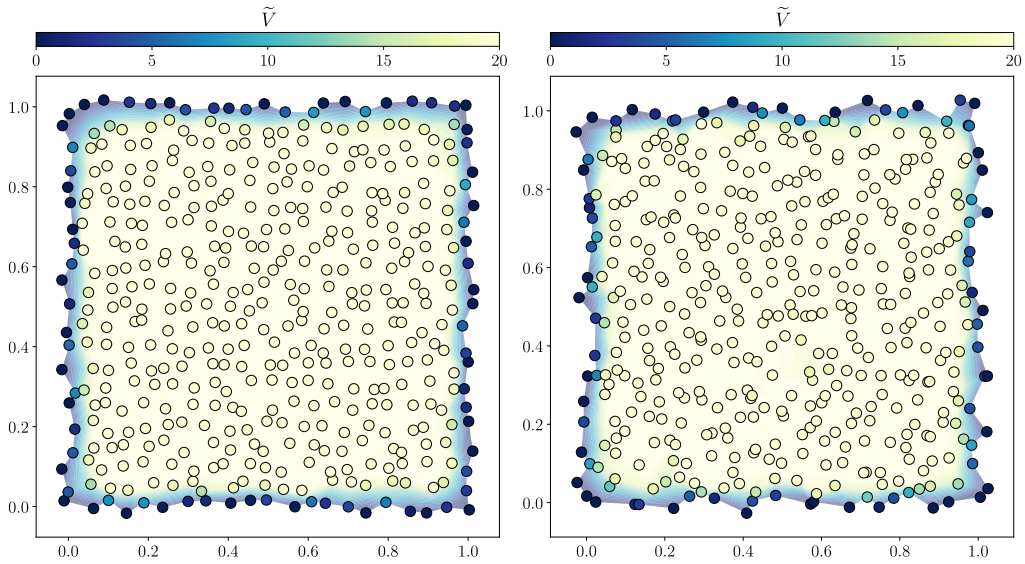

Figure 3.6.: Scatter and contour plots of the non-dimensional neighbourhood volume for each point in square point clouds defined by $c = 70\%$ (left image) and $c = 120\%$ (right image).

It should be noted that $\mathbf{B}_i\boldsymbol{o}_i$ depends on the choice of the weighting function. Let $\widetilde{V}_i$ be some non-dimensional volume of the neighbourhood defined through the sum of the neighbourhood irregularity correction:

$$\widetilde{V}_i = \sum_{j \in \mathcal{N}} \left(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i\boldsymbol{o}_i\right). \tag{3.46}$$

Then the following expression holds for an edge point i:

$$\widetilde{V}_i < \lambda_{FS} V_d, \tag{3.47}$$

where V_d is the non-dimensional compact sphere volume:

$$V_d = \pi \left(\frac{d+1}{3}\right) \left(\frac{h}{\Delta}\right)^d, \tag{3.48}$$

and λ_{FS} is the arbitrary chosen constant used to denote the portion of V_d, which defines minimum non-dimensional volume of the neighbourhood around an inner point. Equation (3.46) is quickly evaluated, since it requires only a scalar product operation of neighbours relative locations and the vector $\mathbf{B}_i\boldsymbol{o}_i$, which was noted to be precomputed for each point in the renormalisation stage.

Figure 3.6 shows how the test performs on highly distorted point clouds. After spawning points on a regular grid, they are moved by a vector $\boldsymbol{e}_i\, c\, \Delta/2$, where c is the measure of irregularity or randomness, and $\boldsymbol{e}_i$ is the d-dimensional vector with random components values between –1.0 and 1.0. Figure 3.6 shows rectangular point clouds that are distorted

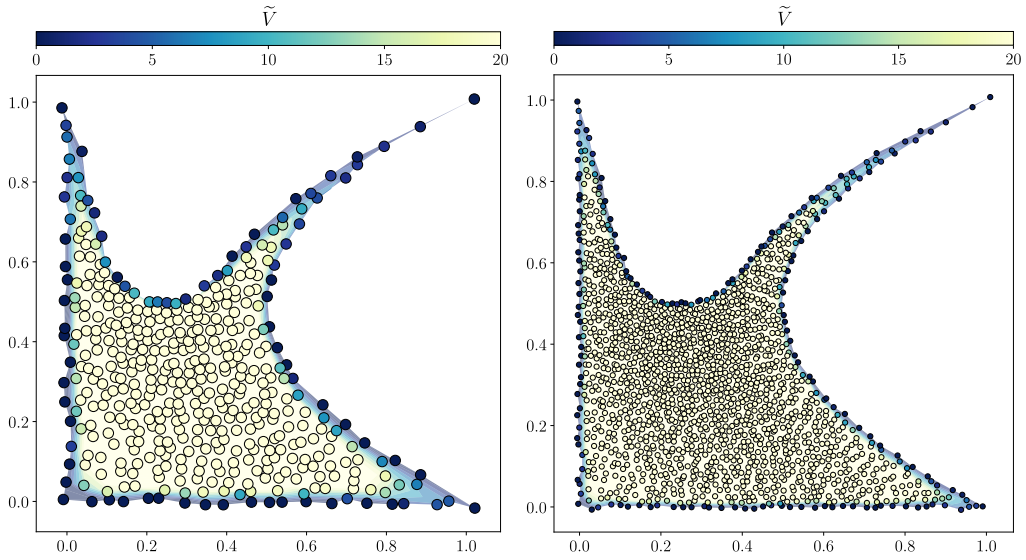

Figure 3.7.: Scatter and contour plots of the non-dimensional neighbourhood volume for each point in a complex-shaped point cloud defined by $c = 70\%$, and $\Delta = 0.05$ (left image) and $\Delta = 0.025$ (right image).

by randomness levels of $c = 70\%$ and $c = 120\%$. Equation (3.46) yields smooth results for extremely distorted point clouds, validating that candidates for boundary points may be found by evaluating equation (3.47). Equation (3.48) is used as a relevant measure for the default volume, which equals to 24.6 for an interior point with $h = 2.8\Delta$. Approximately $\lambda_{FS} = 30\%$ may be used as a good starting point, which means that the volume below 7.5 indicates a boundary point, also proved by figure 3.6.

Furthermore, equation (3.46) is evaluated on a point cloud with curved boundaries. The complex shape was obtained by parabolically compressing the top and right sides of the square point cloud, which resulted in a variable-spaced or compressed point distribution. The distribution of points is then distorted to achieve a randomness level of $c = 70\%$. Figure 3.7 shows how the Laplacian-based test performs on point clouds of complex shapes and variable point spacing, for $\Delta = 0.05$ and $\Delta = 0.025$. It can be observed that equation (3.46) yields smooth results with a distinct transition between certainly interior and certainly boundary points, regardless of the initial point spacing.

Indeed, tested distorted conditions rarely arise in numerical simulations of incompressible flows, where the spacing between adjacent points should remain roughly uniform at all time. One may wonder what should be done with transitional values of the volume. If in some cases the test gives false–positive results, those points are later rejected as free surface points in the second step of the algorithm. Therefore, one can choose relatively high value for λ_{FS} in order to identify more possible candidates, since the second step will take care of the false–positives.

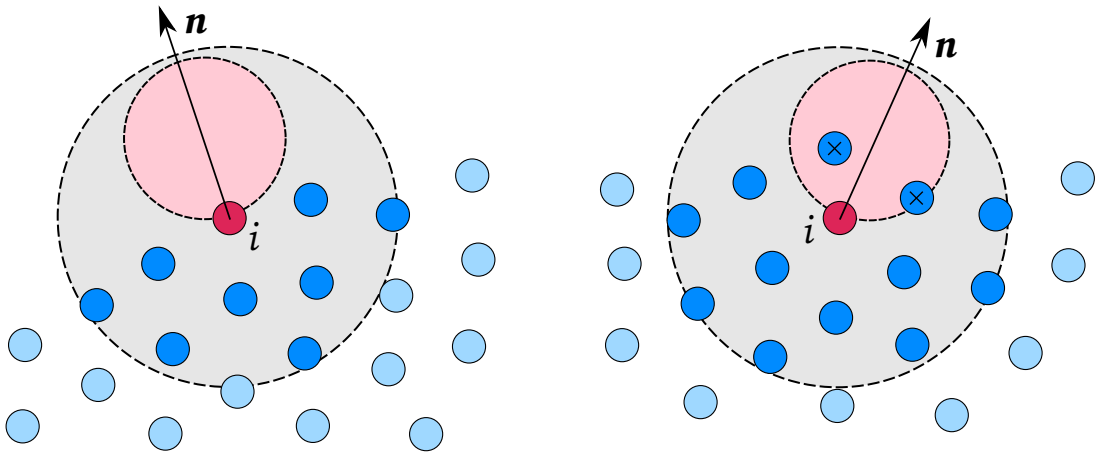

Figure 3.8.: Two examples of a scan–sphere test in 2D, one that passed (left image) and one that failed (right image). The neighbour points inside the grey compact–sphere are coloured dark-blue, and the points inside the red test–sphere are marked with $\times$.

3.5.2. Validating Found Points

The second detection step is used to *validate* if points are lying on the edge of a point cloud. It is usually performed by employing various geometrical overlapping tests, e.g. Marrone *et al.* [153] introduced an idea of using a scan cone around the expected normal vector of the fluid surface to make a further check if there is any particle covering the test particle. Barecasco *et al.* [154] added into account an overlapping spheres test to improve the methods and geometrically detect free surface particles in a robust way. The scan–cone test for each j point in the compact sphere of point i is given as:

$$\frac{\boldsymbol{x}_{ij}}{\|\boldsymbol{x}_{ij}\|} \cdot \boldsymbol{n}_i > \cos\left(\frac{\theta_i}{2}\right), \tag{3.49}$$

where θ_i is the cone threshold angle, e.g. taken as constant $\theta = \pi/3$. Equation (3.49) is true for a free surface point, i.e. there exists no point j that neighbours the point i in the area of expected void where $\boldsymbol{n}_i$ points. Otherwise, a point is classified as inner.

Here a more computationally efficient geometrical test is considered, by employing a "scan sphere". The test holds true for a free surface point if a sphere, which is of radius $R < \Delta$ and centred Δ away from the point i in the normal direction $\boldsymbol{n}_i$, overlaps with no points. Otherwise, a point is classified as an inner point of a fluid, since the sphere overlaps with some neighbouring points where void was expected. The scan–sphere concept is schematically drawn in figure 3.8. The scan–sphere test only uses a small compact sphere ($R < \Delta < h$), as compared to the scan–cone test that uses the full neighbourhood compact sphere ($h > \Delta$), in addition to evaluating equation (3.49) for each neighbouring point.

The second step that successfully eliminates false–positives adds to the robustness of the detection algorithm that must be applied at each time step of a simulation in order to

correctly solve the NSE. Figure 3.9 renders a situation where a complex free surface profile is adequately captured.

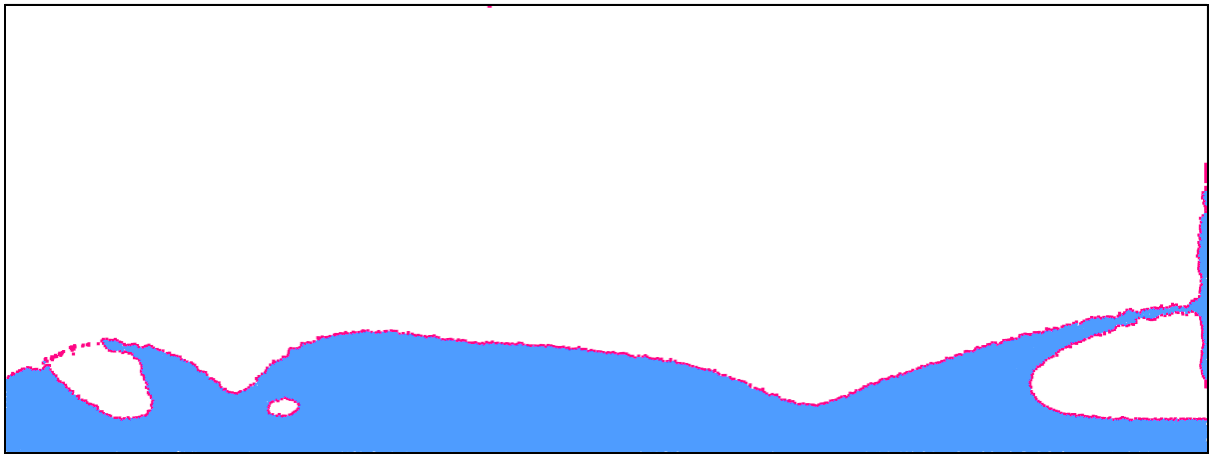

Figure 3.9.: An example of a successful detection of the free surface with a complex shape. Red–coloured points are classified as free surface points.

3.6. Pressure Equation

For numerical reasons, it is easier to solve a set of equations different from the original NSE given in section §2.1. It was shown in section §2.3 how the original divergence-free constraint was replaced by an Poisson equation for the pressure. The introduced set of equations is proven to be equivalent to the original formulation of the NSE. A numerical scheme built on the velocity–pressure formulation is efficient, with the primary computational cost per time step consisting of solving one Poisson equation, irrespective of a 2D or 3D type of the problem. This is, in fact, the best that one can achieve for incompressible flows [155]. By virtue of the advantages listed in section §2.3 and the computational efficiency equivalent to projection methods, the velocity–pressure formulation of the NSE extended to Lagrangian specification of the incompressible flow is used. Incompressibility is enforced at all times, i.e. if the flow contains divergence errors, they will be damped in time both inside the fluid and on the boundaries.

3.6.1. Linear System of Equations

For inner fluid points, the discrete version of the Poisson equation (2.20) writes:

$$\left\langle \nabla^2 p \right\rangle_i = b_i, \qquad \boldsymbol{x}_i \in \Omega, \tag{3.50}$$

where b_i denotes the right-hand-side, which is formally written as:

$$b_i = \rho \left(\nabla \cdot \boldsymbol{f} - \left\langle \frac{\partial \langle \nabla \cdot \boldsymbol{u} \rangle_i}{\partial t} \right\rangle \right), \qquad \boldsymbol{x}_i \in \Omega. \tag{3.51}$$

In this thesis, solenoidal external acceleration vector field is always assumed, i.e. $\nabla \cdot \boldsymbol{f} = 0$ holds everywhere at all time for the gravitational acceleration or some imposed acceleration. The second term in equation (3.51) is also zero in continuous context due to equation (2.2), but never in the discrete and computational context. Since absolutely solenoidal velocity field cannot be computationally achieved, the divergence must be damped as much and as fast as possible. For this reason, the term was formally preserved in (2.20), and the inclusion of the discrete time derivative of the discrete velocity divergence in equation (3.51) is described in section 3.6.3.

In addition to inner fluid points included in the PPE through (2.20), boundary points give closure to the system of equations. The boundary points imposing the Neumann boundary condition are included in the PPE as:

$$\langle \boldsymbol{n} \cdot \nabla p \rangle_i = b_i, \qquad \boldsymbol{x}_i \in \Gamma, \tag{3.52}$$

where the right-hand-side is evaluated based on the required boundary condition, from those listed and described in section §2.4. The right-hand-side is calculated using discrete versions of the spatial derivatives introduced in section §3.2 and section §3.3.

Finally, boundary points imposing the Dirichlet boundary condition are included in the PPE as:

$$p_i = b_i, \qquad \boldsymbol{x}_i \in \Gamma, \tag{3.53}$$

where the right-hand-side is simply the known value. The Dirichlet boundary condition assures that the Poisson equation has a unique solution. Free surface points impose the Dirichlet boundary condition by equalising their pressure values to the atmospheric pressure. Therefore, a robust free surface detection scheme is crucial for the success of the numerical method. The robustness of the detection scheme is explained in section §3.5.

It is important to recall that all spatial derivatives are written in the FD context, i.e. as weighted summation of finite differences between interacting points, see section §3.2 and section §3.3. It is straightforward to merge the equations completely forming the discrete PPE, i.e. equations (3.50), (3.52), and (3.53), into a linear system of equations written in the matrix form as:

$$\mathbf{A}\mathbf{p} = \mathbf{b}, \tag{3.54}$$

where $\mathbf{A}$ is the coefficients matrix, $\mathbf{p}$ is the vector containing pressure solutions for all points forming the system, and $\mathbf{b}$ is the vector containing the right-hand-side values.

Obviously, the number of elements in vectors $\mathbf{p}$ and $\mathbf{b}$ equals to the number of active points in the simulation at the current time step. The coefficients matrix $\mathbf{A}$ is a square matrix with the number of rows and columns equal to the number of active points in the simulation at the current time step. Since the discrete spatial operators are non-global, the matrix $\mathbf{A}$ is sparse. The amount of its sparsity depends on the size of compact radius, i.e. the size of the neighbourhood around points. Sparsity of the system matrix benefits memory consumption and Krylov–iteration performance, as the total number of neighbouring points is proportional to the effort of multiplying the matrix by a vector [156].

In order to define the matrix, the Laplacian is analysed. The sum version of the novel Laplacians, equation (3.36), is the fastest among formulations derived in this thesis, because the gradient is not required for its calculation in contrast to the naive and inverse versions. The renormalisation of the offset vector $\mathbf{B}_i\boldsymbol{o}_i$ is precalculated, so the sum version can be evaluated about as fast as the classical SPH Laplacian. Furthermore, the tests in chapter 5 show that the sum version provides the best compromise between efficiency and accuracy. Finally, it has characteristics that help to detect which points represent free surface, as explained in section 3.5.1. For these reasons, the following mathematical manipulations are based on the sum version of the Laplacian, but the analogous techniques can be used on other versions of the Laplacian to prepare equation (3.54) for solving. According to equation (3.36), the coefficient A_{ij} of the matrix $\mathbf{A}$ for a i-th (row) inner fluid point and a j-th (column) neighbour point is defined as:

$$\mathrm{A}_{ij} = \frac{2d\psi_{ij}\left(1 - \boldsymbol{x}_{ij}\cdot\mathbf{B}_i\boldsymbol{o}_i\right)}{\sum_{j\in\mathcal{N}}\psi_{ij}\left\|\boldsymbol{x}_{ij}\right\|^2\left(1 - \boldsymbol{x}_{ij}\cdot\mathbf{B}_i\boldsymbol{o}_i\right)}, \tag{3.55}$$

and is non-zero if and only if $j \in \mathcal{N}$. From the equation (3.36), it follows that the diagonal coefficient A_{ii} equals to the negative sum of other row coefficients. The dimensionality number and the denominator are constant within the i-th row of the matrix and they can be transferred to the right-hand-side:

$$\mathrm{b}_i = \frac{\rho}{2d}\left\langle\frac{\partial\left\langle\nabla\cdot\boldsymbol{u}\right\rangle_i}{\partial t}\right\rangle\sum_{j\in\mathcal{N}}\psi_{ij}\left\|\boldsymbol{x}_{ij}\right\|^2\left(1 - \boldsymbol{x}_{ij}\cdot\mathbf{B}_i\boldsymbol{o}_i\right), \qquad \boldsymbol{x}_i \in \Omega. \tag{3.56}$$

The expression for the coefficient A_{ij} now simplifies to:

$$\begin{cases}\mathrm{A}_{ij} = \psi_{ij}\left(1 - \boldsymbol{x}_{ij}\cdot\mathbf{B}_i\boldsymbol{o}_i\right) & i \neq j,\\ \mathrm{A}_{ii} = -\sum_{k\in\mathcal{N}}\psi_{ik}\left(1 - \boldsymbol{x}_{ik}\cdot\mathbf{B}_i\boldsymbol{o}_i\right) & i = j.\end{cases} \qquad \boldsymbol{x}_i \in \Omega,\ \boldsymbol{x}_j \in \Gamma\cup\Omega, \tag{3.57}$$

Matrix coefficients for the Neumann boundary condition (3.52) are defined according to the chosen discrete representation of the directional derivative explained in section 3.2.2,

while the right-hand-side is imposed according to the chosen boundary condition listed in section §2.4. The simplest approach is to use the one–sided FD equation (3.17) between the boundary point i and the parent fluid point j located some distance away from the boundary in the boundary normal direction, $\boldsymbol{x}_j = \boldsymbol{x}_i + \|\boldsymbol{x}_{ij}\|\,\boldsymbol{n}$, as explained in section §3.4. Furthermore, the rows can be multiplied by $\|\boldsymbol{x}_{ij}\|$ to remove the order-of-magnitude dependency from the left-hand-side of the system. Therefore, the left-hand-side of equation (3.52) translates to:

$$\begin{cases} \mathrm{A}_{ij} = 1, \\ \mathrm{A}_{ii} = -1. \end{cases} \qquad \boldsymbol{x}_i \in \Gamma,\, \boldsymbol{x}_j \in \Omega, \tag{3.58}$$

Taylor series–based correction of the FD equation (3.17) using the Laplacian term yields equation (3.19) that is more fitting for the PPE. The second term in equation (3.19) is a known quantity taken from equation (3.50), and is transferred to the right-hand-side. The left-hand-side of equation (3.58) stays unmodified, while the right-hand-side for the point i imposing the Neumann boundary condition writes:

$$\mathrm{b}_i = \|\boldsymbol{x}_{ij}\|\,(q_w + \lambda_L\, r_w\, \mathrm{b}_j)\,, \qquad \boldsymbol{x}_i \in \Gamma, \tag{3.59}$$

where q_w is the imposed directional derivative value at the boundary, and b_j is the right-hand-side of equation (3.50) evaluated by equation (3.51). The directional derivative equation (3.18) is more stable than equation (3.19) when explicitly approximating, but not when solving an system of linear equations. The typical gradient helping to describe the directional derivative may convolute the system matrix, due to destroying the favourable properties described in the next section. Particularly, $\mathrm{A}_{ij} = \psi_{ij}\,\boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{n}_i$ does not have a consistent sign. In future work, this will be further investigated.

The Dirichlet boundary condition equation (3.53) written according to equation (3.54) is written as $\mathrm{A}_{ii}\, p_i = \mathrm{b}_i$, i.e.:

$$\begin{cases} \mathrm{A}_{ij} = 0 \quad i \neq j, \\ \mathrm{A}_{ii} = 1 \quad i = j. \end{cases} \qquad \boldsymbol{x}_i \in \Gamma, \tag{3.60}$$

To keep the same sign of the matrix diagonal entries, the rows describing Dirichlet boundary conditions are multiplied by –1. The following text analyses the diagonal and off–diagonal entries, and matrix properties in detail.

3.6.2. Solvability

A drawback of mesh-free finite difference approaches is that the resulting linear systems are in general non-symmetric, i.e. the matrix $\mathbf{A}$ is a *non-symmetric square matrix*. Nevertheless, the sparsity of the system matrix promises a fast solution of system (3.54) with (semi-)iterative linear solvers, assuming that the convergence properties are reasonable [156]. Analysing equation (3.57), the renormalised offset vector $\mathbf{B}_i\boldsymbol{o}_i$ does not participate in keeping the FD form, and therefore is the culprit for making most of the matrix non-symmetric. The other culprits are Neumann boundary conditions, i.e. discrete directional derivatives, which are non-symmetric by their definition. Therefore, an appropriate Krylov subspace iterative solvers that can handle non-symmetric matrices should be used to solve equation (3.54) achieving adequate convergence [157], such as GMRES, BiCGStab, QMR, QMRCGStab, etc.

Analysing the definitions of spatial derivatives or coefficients they form in the matrix by equations (3.57) and (3.58), it is apparent that diagonal entries equal to the negative sum of other entries within the row:

$$\mathrm{A}_{ii} = -\sum_{j\in\mathcal{N}} \mathrm{A}_{ij}, \tag{3.61}$$

while each row corresponding to a Dirichlet boundary condition, equation (3.60), is *strictly diagonally dominant*:

$$|\mathrm{A}_{ii}| > \left|\sum_{j\in\mathcal{N}} \mathrm{A}_{ij}\right| = 0, \tag{3.62}$$

which means that the *diagonal dominance* property of the matrix is satisfied:

$$|\mathrm{A}_{ii}| \geq \left|\sum_{j\in\mathcal{N}} \mathrm{A}_{ij}\right|. \tag{3.63}$$

A strictly diagonally dominant matrix is non-singular. The matrix $\mathbf{A}$ is mostly weakly diagonally dominant except for the rows imposing Dirichlet boundary conditions. Resolving equation (3.61) computationally may accumulate impurities of machine floating number operations, and miss the diagonal dominance by some small value. In order to provide unconditional diagonal dominance for such numerical problems, the diagonal can be slightly scaled. To implement this scaling, left-hand-sides of the equations that form the PPE system are modified by adding the term $-\varepsilon_{ii}p_i$, where ε_{ii} is a small number that helps the diagonal dominance property from a computational point of view, e.g. $\varepsilon_{ii} = 10^{-8}$. The diagonal term is known to smooth the pressure field for methods that use Laplacian of low accuracy. Inaccurate discrete operators forming the linear system sometimes require high values of the term ε_{ii} for the PPE to converge, which results in compressible solutions [158]. Appropriate iterative solvers applied to equation (3.54), which is

described by the introduced spatial derivatives, have been shown to always converge. Indeed, small values of ε_{ii} have been shown to help the convergence. With assistance of the technique that damps the velocity divergence and the technique that conserves the fluid volume on advection, the incompressibility is soon recovered after any non-fatal deviations from equation (2.2) are obtained. These techniques are explained in section 3.6.3 and section 3.8.3, respectively.

The sum version of the Laplacian, equation (3.36), for a highly irregular point cloud can have a small value of the denominator. Imbalance of matrix coefficients due to small values, in combination with machine precision, yield sensitive linear system whose results oscillate locally around the problematic points. The dominator is transferred to the right-hand-side, thus yielding equation (3.57) for the coefficients of inner points. This grants two favourable attributes of the system: the Laplacian coefficients A_{ij} of the matrix have comparable order of magnitude, and problematic low value has lowered the source term for that row alleviating the possible damage.

This paragraph describes the positivity of matrix coefficients. Usually matrices that describe Laplacian graphs or regular grids are symmetric with negative and weakly dominant diagonal entries, while off-diagonal entries are positive. Looking at equation (3.57), it holds that $\psi_{ij} > 0$, but $(1 - \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i)$ may be negative in which case their product is negative. To keep A_{ij} always positive, one may use the constrained value of the neighbour irregularity correction $\min\{1.0, \boldsymbol{x}_{ij} \cdot \mathbf{B}_i \boldsymbol{o}_i\}$ when evaluating equations (3.56) and (3.57), which results in a so-called positive stencil. The desirability of positive stencils has been pointed out in [159]. While optimal stencil selection [160] makes positive stencils *likely*, the introduced constraining technique *guarantees* it. These modifications to the system are justifiable to help the convergence, because unfavourable irregularity correction occurs only for edge cases where all or most of the neighbour points are located on one side. Positive stencils imply that the system matrix $\mathbf{A}$ is an L–matrix. The class of L–matrices are those matrices whose off-diagonal entries are less than or equal to zero (Z–matrices), but whose diagonal entries are constrained as positive. The matrix $\mathbf{A}$ is therefore a L^{-}–matrix, and is made as L–matrix by negation.

As already mentioned in section §3.5, the scheme that detects the points on the free surface is very important for the solvability of the linear system. Points that are marked on the free surface impose Dirichlet boundary condition which provides that the system has a unique solution. A FD matrix is called essentially irreducible if every point is connected to a Dirichlet boundary point. A FD matrix that is not essentially irreducible is singular, as the points that are not connected to a Dirichlet point form a singular sub-matrix. Consequently, if the simulation handles fragmented fluid parts, each fluid clump must have at least one point that imposes the Dirichlet boundary condition (3.53). The matrix $\mathbf{A}$ is *essentially diagonally dominant*, because it is weakly diagonally dominant (equation (3.61))

and each point is through the graph connected to some Dirichlet point satisfying the strict diagonal dominance (equation (3.60)). An essentially diagonally dominant L–matrix is an M–matrix, i.e. $\mathbf{A}$ is an M–matrix because it is discretised by positive stencils and every point is connected to a Dirichlet point [156]. The benefits of an M–matrix structure with respect to linear solvers can be found in [161].

Finally, it should be noted that future work will include making the system matrix symmetric. This is possible if the interaction between neighbours is not forced to be evaluated from a specific point using the renormalisation property of the i-th point, but using the average of neighbours renormalisation properties. In other words, within summations $\sum_{j\in\mathcal{N}}$ forming the system of equations, one would write $(\mathbf{B}_i + \mathbf{B}_j)/2$ instead of $\mathbf{B}_i$, and $(\mathbf{B}_i\boldsymbol{o}_i + \mathbf{B}_j\boldsymbol{o}_j)/2$ instead of $\mathbf{B}_i\boldsymbol{o}_i$. Similar averaging technique was employed in [122, 123] with respect to simulating points of variable spacings, i.e. volumes.

3.6.3. Source Term

Due to initial conditions, truncation errors, machine precision and other sources of errors, the discretely calculated divergence of the velocity will not be identically zero in the numerical computation:

$$\langle \nabla \cdot \boldsymbol{u} \rangle \neq \nabla \cdot \boldsymbol{u} = 0. \tag{3.64}$$

Therefore, in numerical computations one often modifies equation (2.15) by introducing the source term on the right-hand-side that accounts for the discrete time derivative of the discrete velocity divergence. Henshaw [162] includes some amount of the divergence error to the equation:

$$\left\langle \nabla^2 p \right\rangle = \rho\, \lambda_F \left\langle \nabla \cdot \boldsymbol{u} \right\rangle, \tag{3.65}$$

in which case it holds:

$$\left\langle \frac{\partial \left\langle \nabla \cdot \boldsymbol{u} \right\rangle}{\partial t} \right\rangle = \nabla^2 \left\langle \nabla \cdot \boldsymbol{u} \right\rangle - \lambda_F \left\langle \nabla \cdot \boldsymbol{u} \right\rangle. \tag{3.66}$$

The term $\lambda_F \langle \nabla \cdot \boldsymbol{u} \rangle$ relaxes the source of divergence errors, provided that the coefficient is a positive scalar, $\lambda_F > 0$ [162]. Hence, this term, sometimes named the "discrete divergence–damping term", is used on the right-hand-side of the pressure equation. The incompressibility condition is enforced "exponentially", i.e. any errors in satisfying the incompressibility condition are rapidly damped in time. Thus this condition is enforced in a robust way, without expected troubles for "reasonable" numerical discretisations of the equations [97, 96]. This technique has been used previously by a number of researchers in the field of incompressible flows. A detailed description of the damping coefficient, with an analysis showing that the term does not degrade the accuracy of the numerical method, is given by Henshaw *et al.* [162, 163, 99, 164]. Within their implicit method, the authors

relate the coefficient to the viscosity and grid size,

$$\lambda_F = \frac{C\,\nu}{\Delta^2}. \tag{3.67}$$

In comparison to fully implicit methods, explicit methods in time require an explicit relation of the divergence-damping coefficient. Since it is required, or better said desired, that the divergence is zero in the next time step $\langle \nabla \cdot \boldsymbol{u} \rangle_{n+1} = 0$, first order backward differencing (BDF1) yields:

$$\left\langle \frac{\partial \langle \nabla \cdot \boldsymbol{u} \rangle}{\partial t} \right\rangle = -\frac{1}{\delta t} \langle \nabla \cdot \boldsymbol{u} \rangle_n \,, \tag{3.68}$$

where $\lambda_F = \delta t^{-1}$. This assumption leads to oscillations in the pressure fields.

Impulsive impacts need to be carefully assessed, because the change of the situation is processed at discrete time intervals δt. It should be reminded that pressure is transmitted through the fluid at a rate that depends on the speed of sound in the medium and the shape of the container due to the refraction and reflection. The progress of an impact in continuum that is discretely modelled by spatial resolution Δ depends on physical time intervals in which the pressure travels between the points:

$$\delta\tau = \frac{\Delta}{c_S}, \tag{3.69}$$

where c_S is the pressure propagation velocity. Simulations based on explicit schemes mostly run with numerical time steps smaller than physical time steps, $\delta t < \delta\tau$, i.e. a pressure wave rarely propagates further than one cell in a numerical time step.

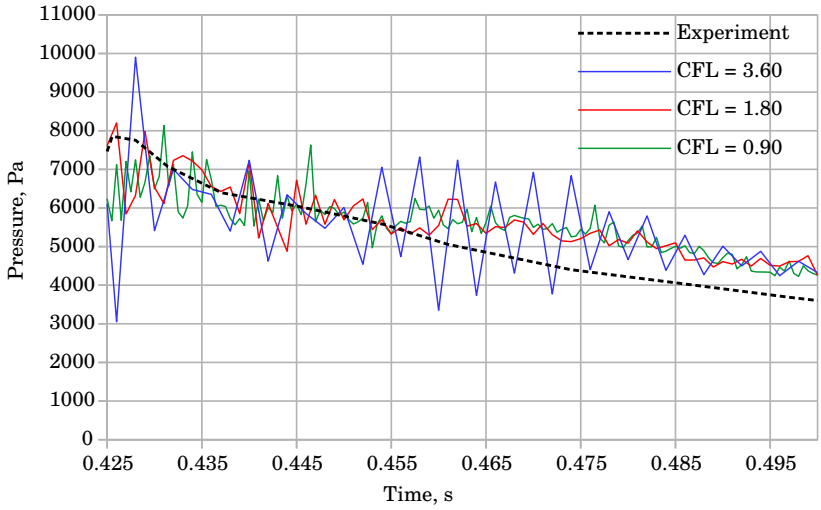

Figure 3.10.: An example of an impact with induced oscillations in the pressure by using the discrete instead of the physical time-step within the pressure source term.

This implies that successive steps of an impact physically described with $\delta\tau$ intervals are numerically shortened to δt, which leads to high frequency oscillations in the pressure field with oscillation amplitudes increasing as $\delta\tau \to 0$. By taking the PPE source term described in section 3.6.3 inversely proportional to the discrete time step δt, a divergence-free solution will try to be forced immediately, which leads to an overshoot of predicting values for the next step, and that iteratively leads to the oscillations. An example of such deliberately induced pressure oscillations after an impact of water on a wall is shown in figure 3.10, for three simulations performed with different CFL numbers.

Cheng *et al.* [158] note that an impulse computed numerically should correspond to the physical value of the impulse. They note that the ratio of the numerical and physical time steps, $\delta t/\delta\tau$, should be used to stabilise high numerical pulses in the pressure field. Namely, the discrete point cloud or any discretisation should respect the physical pressure propagation velocity c_S and the distance between points/cells Δ when exchanging the pressure information between adjacent points/cells.

The calculation of pressure is decoupled from velocity. Therefore, the remedy for the oscillations in the pressure field should be applied in the source term of the PPE. Without a doubt, higher order schemes for the momentum equation will also benefit the smoothness of velocity and pressure fields. Following [158], the divergence damping coefficient should be defined as $\lambda_F = c_S/\Delta$. In their numerical simulation, c_S is set to a very low value to stabilise the MPS method. This thesis acknowledges that physical time step should be used in order to relax the amount of divergence, but the velocity to relax the divergence should not be a global value as suggested in [158]. In this thesis, the damping coefficient is defined for each point as:

$$\lambda_{F,i} = \max\left\{\lambda_C \frac{c_{\max,i}}{\Delta},\ \frac{c_{\min}}{\Delta}\right\}, \tag{3.70}$$

where $c_{\max,i}$ is the maximum relative speed between the point i and its neighbours, $c_i = \|\boldsymbol{u}_i - \boldsymbol{u}_j\|_{\max}$, λ_C is the multiplier term, and $c_{\min}$ may be used to always enforce the source term by some amount of virtual speed of sound.

3.6.4. Divergence Damping at Boundaries

In the following text, it will be shown how the discrete divergence damping is handled at the boundary, which is derived based on the approach introduced by Shirokoff and Rosales [96].

If the momentum equation (2.14) is dotted with the normal $\boldsymbol{n}$ and evaluated at the boundary Γ, the pressure derivative along the normal, equation (2.21) is obtained. However, if

after dotting it is evaluated in the fluid near the boundary, it follows:

$$\boldsymbol{n} \cdot \nabla p = \rho\, \boldsymbol{n} \cdot \left(\boldsymbol{f} - \frac{\partial\, \boldsymbol{u}|_\Gamma}{\partial t} + \nu \nabla^2 \boldsymbol{u} \right), \qquad \boldsymbol{x} \in \Gamma. \tag{3.71}$$

Almost all terms are eliminated from the equation by subtracting by the boundary condition equation (2.21). Then the following Ordinary Differential Equation (ODE) for the normal component of the velocity at the boundary is obtained:

$$\frac{\partial}{\partial t} \left[\boldsymbol{n} \cdot \left(\boldsymbol{u}|_\Gamma - \boldsymbol{g} \right) \right] = 0, \qquad \boldsymbol{x} \in \Gamma. \tag{3.72}$$

Therefore, provided that $\boldsymbol{n} \cdot \left(\boldsymbol{u}|_\Gamma - \boldsymbol{g} \right) = 0$ initially holds, equation (3.72) holds for all time. However, this is true in the continuous context, but discretely the normal boundary condition for the flow velocity is enforced in a rather weak fashion, i.e. occurring errors in satisfying the condition are not damped. The condition lacks the inherent stability provided by the heat equation. Through the pressure derivative, the velocity at the boundary tries to be enforced for the following time step. In practice, this results in a drift of the normal velocity component, which can have destabilising effects on a numerical scheme and must be corrected. In order to resolve the issue, Shirokoff and Rosales [96] simply substitute the ODE by a FD:

$$\frac{\partial}{\partial t} \left[\boldsymbol{n} \cdot \left(\boldsymbol{u}|_\Gamma - \boldsymbol{g} \right) \right] = -\lambda_B\, \boldsymbol{n} \cdot \left(\boldsymbol{u}|_\Gamma - \boldsymbol{g} \right), \qquad \boldsymbol{x} \in \Gamma, \tag{3.73}$$

and use it as feedback term in the pressure boundary condition:

$$\boldsymbol{n} \cdot \nabla p = \rho\, \boldsymbol{n} \cdot \left[\boldsymbol{f} - \frac{\partial \boldsymbol{g}}{\partial t} + \nu \nabla^2\, \boldsymbol{u}|_\Gamma + \lambda_B \left(\boldsymbol{u}|_\Gamma - \boldsymbol{g} \right) \right], \qquad \boldsymbol{x} \in \Gamma, \tag{3.74}$$

where λ_B is the divergence–damping coefficient, which is a positive scalar $\lambda_B > 0$. In equation (3.74), the terms refer to the next time step, $n + 1$. The system of the NSE with the modified pressure boundary condition equation (3.74) is still equivalent to the original formulation of the NSE given in section §2.1 for smooth enough solutions of the velocity and pressure.

The feedback term is responsible for relaxing divergence errors at the boundary in time, if the value of λ_B is appropriately chosen. In [96] details on the choice of the coefficient are not given, but the authors remark that only the order of magnitude must be guessed in order to relax the errors in time. While this is true for fully implicit numerical schemes, semi-implicit and explicit numerical schemes are more sensitive. As it was the case with damping the divergence errors within fluid, a more explicit description of λ_B is needed for the robust stabilisation. One could naively argue that $\boldsymbol{n} \cdot \left(\boldsymbol{u}|_\Gamma - \boldsymbol{g} \right) = 0$ should be zero in the next time step, so the λ_B can be taken inversely proportional to δt. However,

this has shown to be too rigorous for the semi-explicit method described in this thesis. Alternatively, the feedback term in equation (3.74) can be interpreted as a relaxation factor assisting the boundary acceleration, $\partial \boldsymbol{g}/\partial t$, that is desired to be reached in the next time step. Therefore, equation (3.72) evaluated directly using the known accelerations of the boundary itself and fluid at the boundary is added to the pressure boundary condition, and the following expression is obtained:

$$\boldsymbol{n} \cdot \nabla p = \rho\, \boldsymbol{n} \cdot \left[\boldsymbol{f} - \nu \nabla \times \nabla \times \boldsymbol{u} - (1 + \lambda_B) \frac{\partial \boldsymbol{g}}{\partial t} + \lambda_B\, \boldsymbol{a}|_\Gamma \right], \qquad \boldsymbol{x} \in \Gamma, \tag{3.75}$$

where the curl–curl viscous term is used according to equation (2.23). The fluid acceleration near the boundary $\boldsymbol{a}|_\Gamma$ is assumed to be constant in the next time step, $n+1$, so the acceleration from the current time step n may be used.

3.7. Momentum Equation

Since the pressure is considered to be a function of the velocity, equation (2.19), the pressure field is obtained by solving equation (3.54). All terms from the right-hand-side of the momentum equation (2.14) are ready to be approximated.

By using the method of lines (MOL) approach, the discretised equations are solved in time. This technique converts PDEs into an ODE initial value problem, discretising all but one dimension. In this thesis, the considered spatial derivatives are discretised using mesh-free FDs, i.e. they are replaced with algebraic approximations. In effect, only one independent variable remains, the time variable t. This means that any integration algorithm solving initial value ODEs can be used to compute the numerical solution of the PDE. Discretising the momentum equation (2.14) in space yields the resulting system of ODEs:

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = f\left(t, \boldsymbol{u}, p\right). \tag{3.76}$$

The following text describes numerical solutions to the ODE (3.76).

3.7.1. Time Integration

The discrete version of the momentum equation is written as:

$$\left\langle \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \right\rangle = -\frac{1}{\rho} \left\langle \nabla p \right\rangle + \nu \left\langle \nabla^2 \boldsymbol{u} \right\rangle + \boldsymbol{f}, \tag{3.77}$$

where $\langle\rangle$ is implied for the solving time step $(n+1)$, and the right-hand-side of the equation denotes the acceleration, $\boldsymbol{a}_{n+1}$. The explicit or direct computation of the velocity is

done in terms of known quantities of the pressure and diffusion, as implicit computation would add some amount of numerical damping without actually stabilising the convection of violent flows. The pressure, and therefore its gradient, is known after solving the PPE. Moreover, flows under consideration are characterised as impulsive and inertia–dominated, so the diffusion term can be approximated as constant throughout the time step, $\langle \nabla^2 \boldsymbol{u} \rangle_{n+1} \approx \langle \nabla^2 \boldsymbol{u} \rangle_n$, or extrapolated from previous values of the velocity, $\langle \nabla^2 \boldsymbol{u} \rangle_{n+1} \approx \langle \nabla^2 \boldsymbol{u}_{ext} \rangle$. In this thesis, the Laplacian of the current velocity field is used to explicitly calculate the diffusion term, since the extrapolation technique did not show difference when estimating impacts loads. The implicit treatment of the diffusion term will be tested in future work.

In an survey [165], 105 out of 124 studies (85%) on DNS used second order scheme, while 125 out of 132 studies (95%) on LES used second order scheme for the time integration. Most engineers and researchers use second order discretisation schemes both in space and time. In particular, there are no important differences with results with low order in time and high order in space. In this thesis, the first and second order differentiation for the time integration of the velocity are considered.

The FD approximation of the Lagrangian acceleration, using the first–order backward Euler approximation (BDF1) is given as:

$$\left\langle \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \right\rangle = \frac{\boldsymbol{u}_{n+1} - \boldsymbol{u}_n}{\delta t} + \mathcal{O}\left(\delta t\right), \tag{3.78}$$

where δt is the current time step. Evidently, equation (3.78) assumes the acceleration to be constant throughout the time step. The most significant errors arise when capturing rapidly changing flows, i.e. impacts. By applying the Taylor series expansion one more time for the time step n, the second–order backward Euler approximation (BDF2) that holds for variable time steps is given as:

$$\left\langle \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \right\rangle = \frac{1}{\delta t}\left[\left(\frac{1+2r_t}{1+r_t}\right)\boldsymbol{u}_{n+1} - \left(1+r_t\right)\boldsymbol{u}_n + \left(\frac{r_t^2}{1+r_t}\right)\boldsymbol{u}_{n-1}\right] + \mathcal{O}\left(\delta t^2\right), \tag{3.79}$$

where r_t is the ratio of the current and previous time step values. For constant time stepping, equation (3.79) reduces to:

$$\left\langle \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \right\rangle = \frac{3\,\boldsymbol{u}_{n+1} - 4\,\boldsymbol{u}_n + \boldsymbol{u}_{n-1}}{2\,\delta t} + \mathcal{O}\left(\delta t^2\right). \tag{3.80}$$

equations (3.79) and (3.80) incorporate the change in acceleration, i.e. the jerk term, through the discrete velocity derivative between the time steps. In contrast to equation (3.79), the Taylor series expansion may be written by directly using the discrete

change in the acceleration term between two time steps:

$$\boldsymbol{u}_n = \boldsymbol{u}_{n+1} - \delta t \left\langle \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \right\rangle - \frac{1}{2}\delta t^2 \left\langle \frac{\mathrm{D}^2\boldsymbol{u}}{\mathrm{D}t^2} \right\rangle + \mathcal{O}\left(\delta t^2\right), \tag{3.81}$$

which leads to acceleration–based backward Euler approximation (BDFA):

$$\left\langle \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \right\rangle = \frac{\boldsymbol{u}_{n+1} - \boldsymbol{u}_n}{\delta t} - \frac{1}{2}\left(\boldsymbol{a}_{n+1} - \boldsymbol{a}_n\right) + \mathcal{O}\left(\delta t^2\right), \tag{3.82}$$

which one may find convenient since $\boldsymbol{a}_{n+1}$ is known by evaluating the right-hand-side of equation (3.77).

Table 3.1.: Solution to the Lagrangian velocity of fluid points for the next time step.

Scheme	Time stepping	Solution to $\boldsymbol{u}_{n+1}$
BDF1	Variable	$\boldsymbol{u}_n + \delta t \boldsymbol{a}_{n+1}$
BDF2	Constant	$\frac{2}{3}\left(\delta t \boldsymbol{a}_{n+1} + 2\boldsymbol{u}_n - \frac{1}{2}\boldsymbol{u}_{n-1}\right)$
BDF2	Variable	$\left(\frac{1+r_t}{1+2r_t}\right)\left[\delta t \boldsymbol{a}_{n+1} + (1+r_t)\,\boldsymbol{u}_n - \left(\frac{r_t^2}{1+r_t}\right)\boldsymbol{u}_{n-1}\right]$
BDFA	Variable	$\boldsymbol{u}_n + \delta t\left(\frac{3}{2}\boldsymbol{a}_{n+1} - \frac{1}{2}\boldsymbol{a}_n\right)$

The velocity vector at the time step $n+1$, $\boldsymbol{u}_{n+1}$, is the only unknown in equation (3.77) independently of the chosen BDF approximation. Therefore, it is directly calculated from the chosen FD approximation. The solutions are listed in table 3.1 for the described backward–differencing schemes.

Due to the explicit calculation of the velocity used for the advection, this part of the method is formally conditionally stable, indicating that there must be some limit on the size of the time step for there to be a proper solution. Conveniently, the Lagrangian advection described in section §3.8 is well behaved for arbitrarily large values of the time step, i.e. it is unconditionally stable [166, 167]. Consequently, larger time steps applied to solve equation (3.77) will accumulate error during the simulation, but the advection will sustain. In fact, if large time steps are used in areas of high pressure gradients, then the point cloud representing fluid might advect too much and burst.

3.7.2. Time Step Limitations

The diffusive stability constraint is generally defined as:

$$\frac{\nu\,\delta t}{\Delta_{\min}} < \left(\frac{1}{2}\right)^d, \tag{3.83}$$

where $\Delta_{\min}$ is the smallest grid resolution or in the meshless context the point spacing, and δt is the duration of the time step. According to the constraint (3.83), the time step is limited by the ratio of the grid resolution ($\Delta_{\min} \equiv \Delta$ since the spacing is constant) and viscosity:

$$\delta t_{\nu,\,\max} = \varepsilon_\nu \left(\frac{1}{2}\right)^d \left(\frac{\Delta}{\nu}\right), \tag{3.84}$$

where ε_ν is the diffusive time–step limiter, i.e. a positive scalar smaller than 1.0. The convective stability constraint of the usual CFL type is defined as:

$$\frac{u_{\max}\,\delta t}{\Delta_{\min}} < \varepsilon_c, \tag{3.85}$$

where $u_{\max}$ is the maximum magnitude of the velocity found in the velocity field, and ε_c is the value of CFL limit, often taken as 1.0 or lower. According to equation (3.85), the convective stability constraint limits the time step by the ratio of the grid resolution and viscosity:

$$\delta t_{c,\,\max} = \varepsilon_c \left(\frac{\Delta}{u}\right)_{\min}. \tag{3.86}$$

If the point spacing is constant, i.e. adaptive refinement is not used, the term $(\Delta/u)_{\min}$ is evaluated for the point with the maximum speed in the point cloud.

If the viscous term is treated explicitly, then the stability is determined by both constraints (3.83) and (3.85). If the viscous term is treated implicitly, then the standard CFL constraint (3.85) is sufficient for stability for a first or second order semi-implicit time discretisation [97]. Fine point spacing resolution and large kinematic viscosity of the fluid limits the maximum allowed time step to small values. In this thesis, the problems include fresh and sea water with the kinematic viscosity of approximately $\nu = 10^{-6}\,\mathrm{m}^2/\mathrm{s}$. If one takes that very fine point spacing resolution comes down to 0.5 mm for model–scaled problems, equation (3.84) limits the time step to large values, $\delta t_{\nu,\,\max} \gg 1$, due to low viscosity of water. Therefore, the diffusive stability constraint is not relevant for the problems introduced in this thesis.

The introduced convective constraint holds for the Eulerian description of the flow. In the Lagrangian context, the convective derivative is absent. Lagrangian advection is not constrained to classical CFL limiting. It can intrinsically be stable for time stepping with high classical CFL numbers, $\varepsilon_c > 1.0$, because the continuity constraint (2.2) forces neighbouring points to travel with relatively similar velocities, assuming fine discretisation that captures the flow well. In other words, fluid in a static mesh cell always impacts cell boundaries, while in Lagrangian movement the boundaries move with the flow. Due to the connection of the velocity and position integration, the discussion of limiting the time step is continued in section 3.8.4 after introducing the position integration technique.

3.8. Lagrangian Advection

A moving Lagrangian framework often provides better approximations of unsteady flows with free surface and multiphase flows with moving interfaces than a fixed, i.e. Eulerian framework. Moreover, a Lagrangian framework avoids dealing with the non-linear advection term, which often leads to a more accurate representation of transport phenomena. Mesh-free methods intrinsically cohere with Lagrangian movement, since points are free of topology. This flexibility of free–form movement can hamper the conservation laws, which must be assessed. Specifically, inaccurate movement of mesh-free points in incompressible flows produce errors in volume conservation that results in numerical compressibility.

3.8.1. Explicit Movement of the Point Cloud

After solving the momentum equation without implicitly advecting meshless points, Lagrangian movement is usually done explicitly along the solved streamlines. It leads to the distortion of the particle distribution and thus introduces volume and mass conservation errors, especially in flows with higher Reynolds numbers, e.g. this can be illustrated by a rotating disc which boundary points move in the tangential direction and increase the volume of the disc [168]. Figure 3.11 shows an example of such distorted particle distribution, where the volume conservation is clearly violated. The volume beneath falling water column has compressed, while the mixing area got unstable.

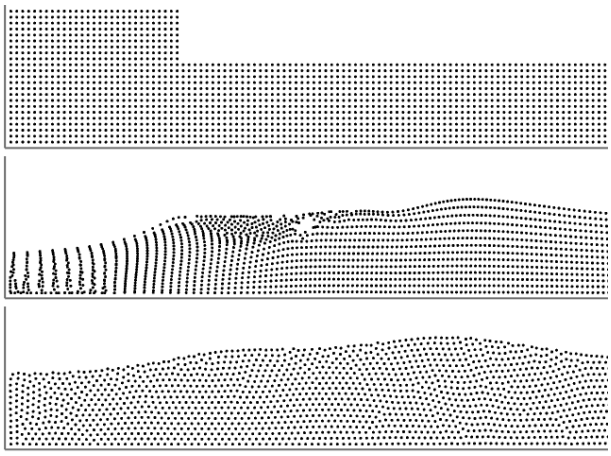

Figure 3.11.: Gravity–induced test (top image) that results in a distorted particle distribution without explicit volume conservation (middle image), compared to an ordered particle distribution that conserves volume (bottom image).

The first order movement of points at n-th time step is described as:

$$\delta \boldsymbol{x}_{i,n} = \boldsymbol{u}_n \left(\boldsymbol{x}_i \right) \, \delta t, \tag{3.87}$$

where the velocity $\boldsymbol{u}$ at the location of the point i may be taken as the point's own velocity $\boldsymbol{u}_i$, or as the "smoothed" velocity of the neighbourhood calculated by the interpolation equation (3.4), etc. Despite the significant inaccuracies that are accumulated when using equation (3.87), the first order movement is still used in most of Lagrangian SPH methods, e.g. [88]. Instead of assuming that the velocity is constant between two time steps, the velocity derivative can be used and assumed to be constant between two time steps. From the Taylor's series expansion, it follows:

$$\begin{aligned} \delta\boldsymbol{x}_{i,n} &= \boldsymbol{u}_n\,\delta t + \frac{1}{2}\frac{\boldsymbol{u}_n - \boldsymbol{u}_{n-1}}{\delta t}\delta t^2, \\ &= \left(\frac{3}{2}\boldsymbol{u}_n - \frac{1}{2}\boldsymbol{u}_{n-1}\right)\delta t, \end{aligned} \tag{3.88}$$

which is called the second order time integration method. Higher order time integration methods, which use multiple time levels before the current one, should be used with caution in violent flows, e.g. on impacts, due to accounting for velocities before the abrupt flow change happening at the current time instant. As a compromise, the higher order term of the Taylor's series, i.e. the acceleration rate of change, can be introduced using the current and previous time step and assuming that the jerk is constant between two time steps:

$$\begin{aligned} \delta\boldsymbol{x}_{i,n} &= \boldsymbol{u}_n\,\delta t + \frac{1}{2}\frac{\boldsymbol{u}_n - \boldsymbol{u}_{n-1}}{\delta t}\delta t^2 + \frac{1}{6}\frac{\boldsymbol{a}_n - \boldsymbol{a}_{n-1}}{\delta t}\delta t^3, \\ &= \left[\frac{3}{2}\boldsymbol{u}_n - \frac{1}{2}\boldsymbol{u}_{n-1} + \frac{1}{6}\left(\boldsymbol{a}_n - \boldsymbol{a}_{n-1}\right)\delta t\right]\delta t, \end{aligned} \tag{3.89}$$

where $\boldsymbol{a}$ is the acceleration at the location of the point i, taken as the point's own acceleration obtained from the Navier–Stokes momentum equation. Taylor's series expansion holds for continuous and smooth functions. Therefore, the explicit movements of points by equations (3.88) or (3.89) that experience violent impacts on solids should use time step small enough to smoothly represent the functions of the position and velocity in time.

A treatment to avoid the highly anisotropic particle distribution that usually triggers instability, as seen in figure 3.11, is slightly shifting particles away from their streamlines, which was firstly introduced by Xu *et al.* [169]. The shifting technique was later improved by Lind *et al.* [170] by using Fick's law of diffusion, in order to control the particle distribution. Khorasanizade and Sousa [149] adjusted the technique for flows at high Reynolds numbers. Suchde and Kuhnert [168] tried to improve the Lagrangian movement without the introduction of any additional artificial movements. They approximated streamline velocities by an ODE, and moved points along these approximated streamlines. It can be argued that the ODE approach in lieu of additional shifting yields more accurate results for rapidly changing flow profiles. However, the sloshing test case in [168] suggests that the improved movement methods even with small time steps yield volume conservation

errors in the order of magnitude of 10%. Furthermore, it was shown, and it will be also validated in this thesis, that the artificial movements do not contaminate simulations if the time–step and movement distance is restricted to adequate values, i.e. Lagrangian characteristics are preserved [170]. Consequently, a shifting technique that preserved the Lagrangian property of the system keeps a simulation stable for any amount of time due to explicit reordering of points in the point cloud. Finally, a shifting technique without time dependency and parameters tweaking, that can also naturally handle solid boundaries, is desirable. Therefore, Position Based Dynamics (PBD) technique is investigated, as an alternative to the shifting introduced in [169], due to its unconditionally stable time integration and robustness [166]. In the following text, the "classical" shifting and PBD techniques are described.

3.8.2. The Shifting Technique

The particle or point shifting is also known as corrective displacement and particle regularization. The particle flux is defined by:

$$\boldsymbol{J} = -D\,\nabla C, \tag{3.90}$$

where C is the particle concentration and D is the diffusion coefficient. Assuming that the flux is proportional to the velocity of the particles, a particle shifting velocity $\boldsymbol{u}_s$, and subsequently a particle shifting distance, $\delta\boldsymbol{r}_s = \boldsymbol{u}_s\,\Delta t$, can be found. Consequently, the particle shifting distance is proportional to $\boldsymbol{J}\Delta t$. Special attention needs to be given to the value of diffusion coefficient D , which should be large enough to provide effective particle shifting and redistribution, but small enough to avoid error by excessive diffusion. An upper limit on the diffusion coefficient can be found through a Von Neumann stability analysis of the advection–diffusion equation. Skillen *et al.* [128] suggest $D \leq h\,\|\boldsymbol{u}\|_i$. Therefore, the proportionality mentioned before leads to:

$$\delta\boldsymbol{r}_s = -A\,h\,\|\boldsymbol{u}\|_i\,\Delta t\,\nabla C, \tag{3.91}$$

where A is the problem–dependent dimensionless constant, with a value within an order of magnitude of unity.

3.8.3. Position Based Dynamics

A set of nonlinear constraints that enforce constant density is solved using Jacobi iterations by updating particle positions each iteration [167], which is described in the following text. The PBD applied to constraining the fluid density is referred to as Position Based

Fluids (PBF). The constraint is defined as:

$$C\left(\boldsymbol{x}_i\right) \equiv C_i = \frac{\rho_i}{\rho_{rest}} - 1, \tag{3.92}$$

where ρ_i is the numerical density estimated in a SPH fashion as $\rho_i = \sum_{\mathcal{N}} W_{ij}$, and ρ_{rest} is the numerical density estimated for the chosen regular point distribution: uniform Cartesian grid or hexagonal point distribution. When reorganising points via the PBD, the compact radius h_P does not need to have the same value as the compact radius h used for calculating gradients and Laplacians.

The idea is to find a correction to the current point location, $\delta\boldsymbol{x}_i$, so that $C\left(\boldsymbol{x}_i + \delta\boldsymbol{x}_i\right) = 0$ holds. The change in constraint (3.92) can generally be represented through equation (3.5) in the following way:

$$C\left(\boldsymbol{x}_i + \delta\boldsymbol{x}_i\right) \approx C_i + \delta\boldsymbol{x}_i \cdot \nabla C_i = 0. \tag{3.93}$$

Assume that the location correction $\delta\boldsymbol{x}_i$ should be in the direction of ∇C_i, i.e. the vectors are co-linear with each other:

$$\delta\boldsymbol{x}_i = \lambda \nabla C_i, \tag{3.94}$$

which substituted back in equation (3.93) yields:

$$C_i + \lambda_{P,i} \nabla C_i \cdot \nabla C_i = 0. \tag{3.95}$$

Now the scaling factor $\lambda_{P,i}$ can be obtained:

$$\lambda_{P,i} = -\frac{C_i}{\sum_k \left\|\nabla_k C_i\right\|^2 + \varepsilon_P}, \tag{3.96}$$

where ε_P is the relaxation parameter taken as constant, and $\nabla_k C_i$ is defined as:

$$\nabla_k C_i = \frac{1}{\rho_{rest}} \begin{cases} \sum_j \nabla W_{ij}, & k = i, \\ -\nabla W_{ij}, & k = j, \end{cases} \tag{3.97}$$

in order to solve in a parallel Jacobi fashion [167], instead of the sequential Gauss–Seidel iterations [166], and ∇W_{ij} is estimated in an SPH fashion. Equation (3.96) is used for the position update by including corrections from neighbours:

$$\delta\boldsymbol{x}_i = \frac{1}{\rho_{rest}} \sum_j \left(\lambda_{P,i} + \lambda_{P,j}\right) \nabla W_{ij}, \tag{3.98}$$

By empirical evaluation, it was found that a small compact radius is efficient, $h_P < 2\Delta$, and that $\varepsilon_P = 10^{-5}$ is adequate. The point cloud is firstly advected as described in section 3.8.1 and then those new positions are corrected by iteratively applying corrections

described by equation (3.98). After the points are corrected, velocities at the new locations are updated by meshless interpolation at the uncorrected locations by equation (3.4).

3.8.4. Time Step Limitations

PBD techniques are said to be unconditionally stable [166]. Equation (3.98) that updates the position does not extrapolate into the future as explicit schemes do, but move the points to a physically valid configuration computed by the constraint solver. It is parameter-free and thus problem–independent, unlike the shifting technique described in section 3.8.2 that relies on the point velocity and problem–dependent coefficients. In other words, the PBF technique can recover point cloud regularity from highly irregular distributions that may have arisen from inadequate or defective point cloud movement, assuming enough iterations of equation (3.98) are performed. In conclusion, the PBF technique does not append time step restrictions.

On the other hand, one should be careful with the fact that the point cloud movement described in section 3.8.1 in combination with the PBF technique can handle large time steps, because stepping restrictions for the velocity still apply. Large time steps and point movements may accumulate velocity divergence errors that cannot be damped faster than they are generated. In such cases, abrupt and immense pressure gradients produced after solving the PPE blow up problematic areas. The time step should be chosen to ensure sufficient accuracy of the solution, but in any case can be equal or greater than for an Eulerian scheme. For example, for a situation in which adjacent fluid parcels mix, time step should be smaller than for a situation in which fluid parcels advect in the same direction. Due to the consideration of violent flows, it follows that in addition to the mixing sensitivity, the choice of the time step should also be sensitive to impacts of the free surface on the boundary, and impacts of fluid fragments between themselves.

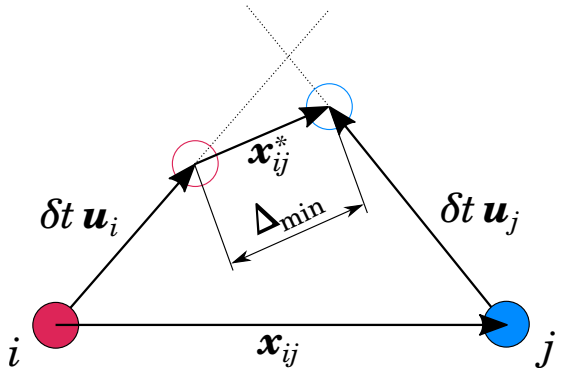

Figure 3.12.: Extrapolated paths of two neighbour points and their predicted distance from each other.

The question of adaptively choosing the value of the next time step can be formed as follows: how long does it take for two neighbour points to collide, i.e. their paths to

become too close to each other? The problem is schematically drawn in figure 3.12, where the first order integration is assumed, the predicted distance vector of two neighbour points is obtained by equation (3.87):

$$\boldsymbol{x}_{ij}^{\star} = \boldsymbol{x}_{ij} + \boldsymbol{u}_{ij}\,\delta t, \tag{3.99}$$

which is approximate, because the velocities of the current time step are used. Now we are interested into finding the maximum time step, $\delta t_{\max}$, for which the two points are going to be too close to each other, $\left\|\boldsymbol{x}_{ij}^{\star}\right\| \leq \Delta_{\min}$. The problem is posed by squaring the above equation, i.e. by dotting it with itself, which yields a quadratic equation. The solution to the equation is:

$$\delta t_{\max} = \frac{1}{\|\boldsymbol{u}_{ij}\|^2}\left\{\pm\sqrt{\left(\boldsymbol{x}_{ij}\cdot\boldsymbol{u}_{ij}\right)^2 + \|\boldsymbol{u}_{ij}\|^2\left(\Delta_{\min}^2 - \|\boldsymbol{x}_{ij}\|^2\right)} - \boldsymbol{x}_{ij}\cdot\boldsymbol{u}_{ij}\right\}. \tag{3.100}$$

If the sum under the radical sign is negative or if the result of $\delta t_{\max}$ is negative, then the two points are distancing away from each other. Otherwise, if the equation has two positive solutions the points are passing by each other like shown in figure 3.12, so the minimum of the two solutions is chosen. The minimum allowed distance is a fraction of the initial spacing, $\Delta_{\min} = 0.4\,\Delta$ was empirically found to be appropriate. Finally, the solutions of equation (3.100) may be limited by the CFL constraint (3.86), $\delta t_{\max} \leq \varepsilon_c\,\Delta / \|\boldsymbol{u}_i\|$, where ε_c limits the number of initial points spacings that the point can travel in a time step, e.g. $\varepsilon_c = 1.8$.

3.8.5. Volume Conservation

The PBF technique can be depicted as a re-meshing method, not as an integration method. Its objective is to conserve the volume of fluid in the domain after the integration step, based on virtual volumes of points by enforcing equidistant neighbour points.

For the sake of comparison, flux–based conservative methods conserve the volume in incompressible flows by controlling the flux through faces from both sides. A solenoidal velocity field has to exist, equation (2.2), in order to describe the incompressible flow. When integrating over the area, the continuity states that the net mass accumulation in the control volume is zero. When $\nabla\cdot\boldsymbol{u} \neq 0$, some amount of mass nonphysically increases or disappears from each cell. If a flux–based method fails to ideally satisfy the incompressibility constraint, it may accumulate the error through time, which is pronounced in Discontinuous Galerkin (DG) methods [171]. Moreover, when $\mathrm{d}\rho/\mathrm{d}t + \nabla\cdot(\rho\boldsymbol{u}) = 0$ is not ideally satisfied in multiphase free surface flows, a FVM may accumulate the error through time, yielding high L^{∞}–norm of the error. Alternatively, the existence of Lagrangian points and the constantness of distances between closest neighbour points in the

point cloud guarantee the required fluid volume at all time. Figure 3.13 shows the noted difference of conserving the volume in a ill-prepared simulation by a flux–based solver and a PBF solver that optimises the volume of a point cloud. Non-ideal convergence when constraining equidistances between points with the PBF leads to slight oscillations of global fluid volume in the domain, but it cannot result in uncontrollable loss or gain in volume. In conclusion, the PBF technique conserves the global volume of fluid at all time.

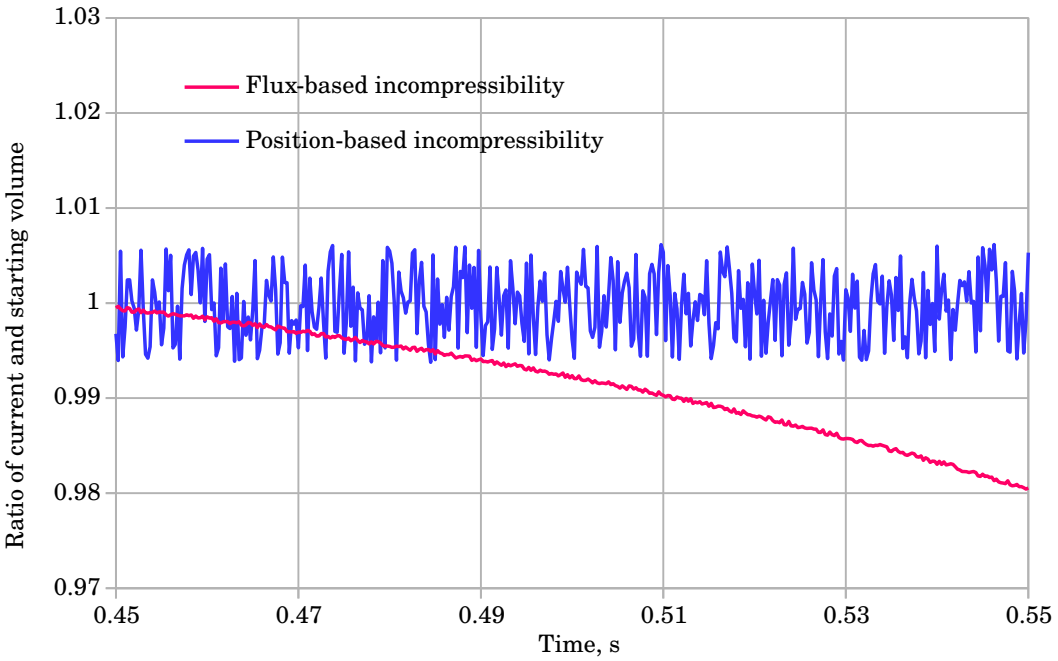

Figure 3.13.: An example of the conservation of total fluid volume through simulation time: flux–based accumulation of error versus the PBF technique trying to optimise the point cloud arrangement.

3.9. Coupling Between Domains

3.9.1. State of the Art

The numerical treatment of inflow and outflow boundary conditions in Lagrangian meshless methods deserves special attention. For the most popular meshless method (SPH), the problem is listed among the 'Grand Challenges' by the SPH European Research Interest Community [149].

Some researchers instead of dealing with open boundaries try to employ periodic boundaries, which obviously offer limited capabilities. Generally, when imposing an open boundary condition for incompressible flows, volume conservation is a primary factor that needs to

be appropriately accounted for. SPH and MPS particles need to be carefully created and removed in order to keep an adequate distribution near open boundaries. For the sake of comparison, it's worth noting that Eulerian methods that work with fluxes do not deal with such challenges connected to physical "cell" movement, but need to take care about cells intersected by the free surface.

Often the open boundary conditions in the SPH method are implemented by means of buffer regions whereby physical quantities are either imposed or extrapolated from the fluid region [172]. This is due to the fact that SPH operators cannot yield accurate results if the kernel is truncated. Khorasanizade and Sousa [149] proposed a buffer–based inflow/outflow boundary treatment for the semi-implicit SPH method for high Reynolds numbers. They modify particle shifting technique to limit possible volume conservation errors. Hirschler *et al.* [127] formulated the open boundary as an impermeable wall boundary by mirroring particles that move with the specified velocity of the boundary. This technique can lead to large errors if the velocity profile at the open boundary is very steep. As opposed to the buffer–based methods, Ferrand *et al.* [173] extended their semi-analytical boundaries for the SPH method that account for truncating the kernel [152]. Since it relies on mixing the Lagrangian and Eulerian concepts for the boundary segments, it uses Riemann invariants to impose compatible density and velocity fields.

The referenced schemes use buffers of particles due to the nature of the SPH spatial operators, which cannot handle partially truncated neighbourhoods. On the contrary, the renormalised operators used in this thesis work adequately when a point is encircled by a single ring of neighbour points. They are to a degree immune to the point cloud irregularity, and always exact for linear fields in the neighbourhood. Consequently, boundary conditions can be applied by using a single array of points placed on the boundary, because the boundary points will contribute to encircling fluid points located near the boundary surface when calculating renormalised spatial operators. In conclusion, the boundary–projection technique applied to solid walls is applicable to all boundaries, since buffers of points behind boundaries insignificantly affect the accuracy of solutions to the PPE.

3.9.2. Inlet Boundaries

For the class of problems this thesis investigates, the velocity profiles are available at inflow cross sections, Γ_{in}. Therefore, the velocity of points is prescribed at the inlet boundary, $\boldsymbol{x} \in \Gamma_{in}$, according to the known profile to impose the Dirichlet boundary condition (2.26) on the velocity. In this thesis, inlet boundaries are exclusively used for the propagation of waves generated by potential flow theory that is described in section §2.6. The potential flow solver provides the velocity field, $\boldsymbol{u}_{lo.fi.}(\boldsymbol{x}, t)$, and free surface elevation information,

$\eta_{lo.fi.}(x, y, t)$. As discussed in section 2.4.3, it can be assumed that the pressure does not change in the direction normal to the inlet boundary surface, so the zero–Neumann boundary condition (2.28) is imposed on the pressure when solving the PPE.

Now that imposing the boundary conditions is conclusive, the remaining issue to be dealt with is controlling the Lagrangian advection of points into the domain. After obtaining the solution to the pressure field, the velocity is marched in time. Points entering the domain need to be adequately arranged near the open boundaries, i.e. each newly generated point has to be assigned the same virtual volume. This issue is taken care of by the volume conservation technique illustrated in section 3.8.3. When considering the generation of waves at the boundary, positive and negative velocities relative to the boundary are periodically occurring, i.e. fluid flows inside and outside of the domain. This is schematically drawn in figure 2.2. The points that are generated to impose the boundary condition for waves, which also do not enter the domain in the next time step, do not pose any problems for the method as they are simply discarded.

The main pieces of the scheme are summarised below.

- Fluid points near the boundary generate temporary boundary points for imposing the inlet boundary conditions for the current time step. If the solution is known only at the boundary, then the boundary point is generated by projecting on the boundary surface. Else if the solution is also known behind the boundary surface, then the generated boundary point b is distanced by the value of point spacing Δ from their parent fluid point i, $\boldsymbol{x}_b = \boldsymbol{x}_i - \max\{\Delta, r_{in}\}\boldsymbol{n}$, where r_{in} is the distance from the fluid point to the boundary surface, and the normal $\boldsymbol{n}$ points into the fluid.
- The known value of the velocity $\boldsymbol{u}_b$ is assigned to the generated boundary point b.
- The zero–Neumann boundary condition (2.28) is imposed in the PPE. The simple central FD equation (3.17) is used for the directional derivative in equation (2.28), because $p_b \approx p_i$.
- The temporary boundary points advect with fluid points. Before advecting the points, the relaxation of the velocity field is done in the volume that stretches from the boundary surface in the normal direction of the surface. This stabilises the simulation, as described in section 3.9.4. Temporary points that enter the fluid domain after the movement are converted to fluid points, and those that stay out of the domain are discarded.

3.9.3. Open Boundaries

When analysing spatially decomposed external–flow problems with an inlet that forces the fluid to flow into the computational domain, the flow properties at the remaining

boundaries are usually unconstrained. Therefore, instead of using outlet boundaries that force the flow outside the domain, open boundaries are used. They let the fluid to freely enter and leave the computational domain while insignificantly affecting the development of the flow. The numerical scheme for open boundaries follows the principle of inlet boundaries illustrated in figure 5.13, with some differences. The main pieces of the scheme are listed as follows.

- Fluid points near the boundary generate temporary boundary points in the direction normal to the boundary surface. The generated boundary point b is distanced by point spacing Δ from their parent fluid point i, $\boldsymbol{x}_b = \boldsymbol{x}_i - \Delta \boldsymbol{n}$.
- An approximate value of the velocity $\boldsymbol{u}_b$ is assigned to the generated boundary point. More accurate approximation would be set using extrapolation, but as argumented in section §2.4, the assumption that the velocity does not change in space and time is a valid approximation for open boundaries far away from the body, $\boldsymbol{u}_b = \boldsymbol{u}_i$.
- The Neumann boundary condition (2.30) is imposed in the PPE as it is for the boundary points on the wall. The right-hand-side of equation (2.30) is calculated by assuming that the velocity does not change in time, $\partial \boldsymbol{u}\left(\boldsymbol{x}_b\right)/\partial t = 0$. The simple central FD equation (3.17) is used for the directional derivative in equation (2.30), which evaluates at $\left(\boldsymbol{x}_b + \boldsymbol{x}_i\right)/2$, as it does not make much difference where the boundary exactly is.
- The boundary points advect with fluid points. Those points that cross the boundary and enter the fluid domain are converted to fluid points, and those that leave the fluid domain are removed from the simulation.

In a simulation where an inlet boundary is the main source of flow development, e.g. by generating waves, the remaining open boundaries of the domain must not feed the domain with new energy. For whatever reason, the flow solution may be rotational or reverting near the open boundary, $\boldsymbol{u}|_\Gamma \cdot \boldsymbol{n} > 0$. In addition, it may be accelerating into the domain, $\boldsymbol{n} \cdot \partial\, \boldsymbol{u}|_\Gamma / \partial t > 0$, thus pushing new flow from unknown origin (unlike inlets do). This is avoided by combining the three following components:

- Open boundaries from the objects under consideration should be "far enough".
- The boundary condition should not extrapolate from the unknown; the velocity is assumed to be constant in time and space.
- Flow may be damped or relaxed before reaching the boundary. There are various techniques to damp the free surface waves [174]. Principles of the relaxation zone used in this thesis are described in the next section.

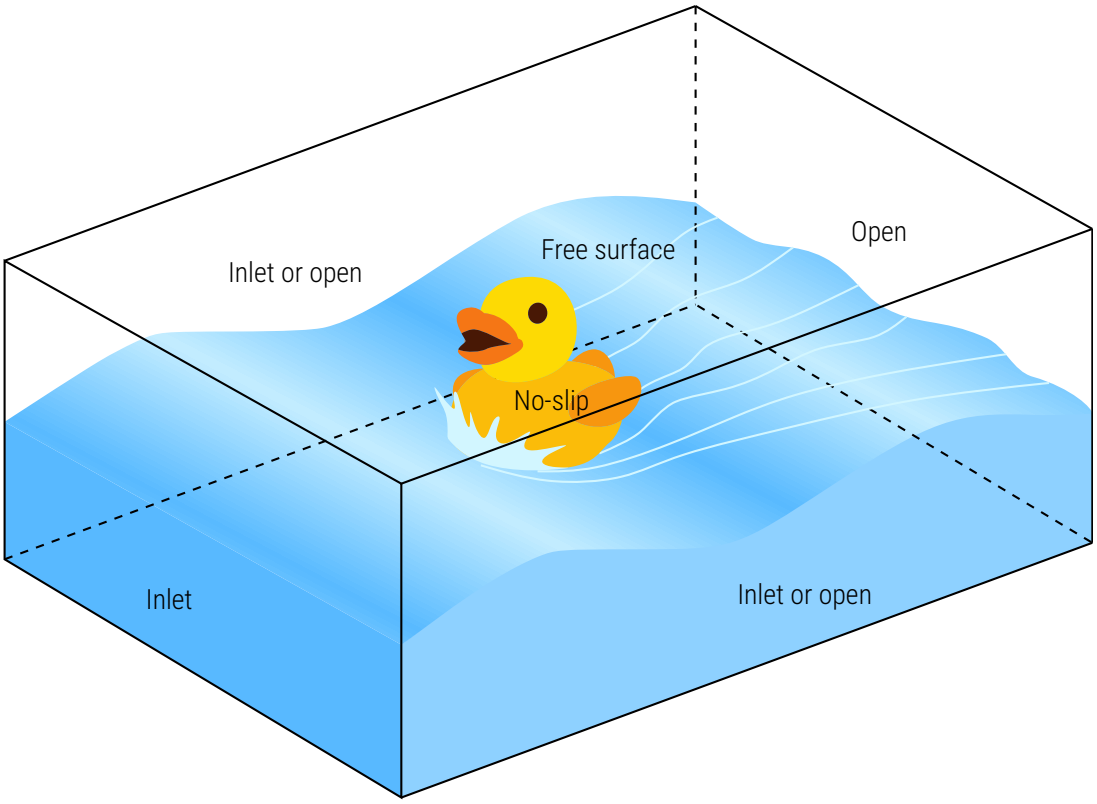

Figure 3.14.: Schematic of a common domain setup.

3.9.4. Relaxation Zones

Spurious reflections from boundaries, waves reflected internally or incomplete convergence in numerical simulations contaminate the results. Reflections between the wave generator and an object add extra energy to the simulation. Relaxation zones are often employed to solve the problems from the boundaries by weighting between the computed solution and some target solution. Generally, the relaxation can be done explicitly or implicitly, depending on the time integration scheme. The momentum equation is treated explicitly in time due to the characteristics of the NSE solving scheme and the Lagrangian nature of the method, as described in section §3.7. Therefore, the relaxation technique within a defined zone is also done explicitly, after the momentum equation had been solved for the velocity at the next time step. The explicit approach of the relaxation of some scalar or vector field inside the relaxation zone is defined as the linear combination of the computed and target solutions:

$$f(\boldsymbol{x}) = \alpha_R \, f_{\text{computed}} + (1 - \alpha_R) \; f_{\text{target}}, \tag{3.101}$$

where α_R is the relaxation weighting function, $0 \leq \alpha_R \leq 1$, and f is some scalar or vector field under consideration. The concept of relaxation weighting is drawn in figure 3.15. Relaxation zones are located within the computational domain. In those zones the flow variables are regularly computed, but afterwards they are relaxed by a fraction of the

imposed solution. The presence of relaxation zones in the domain does not conflict with the concept that is responsible for managing the spawning of fluid points entering the domain and removing of fluid points leaving the domain.

Relaxing the flow at the inlet boundary is straightforward, i.e. equation (3.101) is evaluated to correct the velocity field $\boldsymbol{u}_{\text{computed}}(\boldsymbol{x})$ after it has been solved for the next time step. On the other hand, open boundaries require not to impose some solution, but to smooth out the flow as depicted in figure 3.15. Sometimes a relaxation zone in front of the open boundary smooths out the flow in order to model the outlet boundary. In this thesis, such heavy modifications to the flow are not needed.

The zones in front of open boundaries relax the flow firstly by interpolating the source term in equation (3.50) by equation (3.101), where $\mathrm{b}_{\text{target}} = 0$. This results in minimal production of pressure gradients near open boundaries. Secondly, the flow is regularised in a way that it flows normal to the boundary. This is achieved by interpolating the velocity field from its computed solution to the normal projection of the velocity field, i.e. $\boldsymbol{u}_{\text{target}} = \boldsymbol{n}(\boldsymbol{n} \cdot \boldsymbol{u})$. Assuming that the open boundary is an outlet boundary, the relaxation imposes the velocity vector to be directed outside of the domain, $\boldsymbol{u}_{\text{target}} = \boldsymbol{n} \min\{0, (\boldsymbol{n} \cdot \boldsymbol{u})\}$, where the boundary surface normal $\boldsymbol{n}$ is pointing from the boundary towards the fluid.

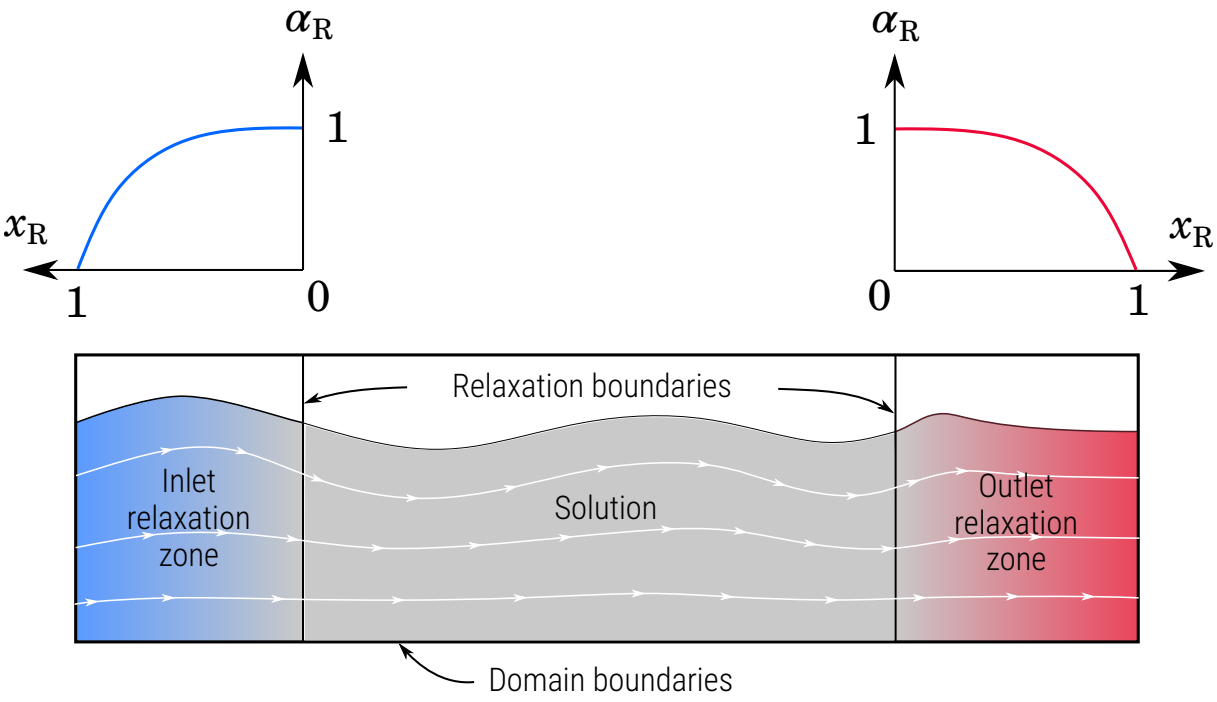

Figure 3.15.: The relaxation zone technique.

Similar to the choice of weighting function for meshless operations, there are numerous options for relaxation weighting introduced in the literature. The exponential weighting function is a sigmoid–like function, which is often used for the relaxation in incompressible

flows [175]. It has the following form:

$$\alpha_R\left(x_R\right) = 1 - \frac{\exp\left(x_R^{\beta}\right) - 1}{\exp\left(1\right) - 1}, \tag{3.102}$$

where x_R is the distance from the relaxation boundary normalised by the length of the relaxation zone ($x_R = 1$ at the physical boundary), and β is the exponent that dictates the steepness of the relaxation, often taken as $\beta = 3.5$ [176]. The function very gently interpolates from the computed solution, and later abruptly into the target solution. The exponential weighting function would belong to a class of spiked functions used in meshless methods. The Spiky kernel [177] is an often used weighting function in the SPH method, which transformed to coordinates of the relaxation zone writes:

$$\alpha_R\left(x_R\right) = 1 - x_R^{\beta}, \tag{3.103}$$

where the exponent β is commonly taken as 3.0.

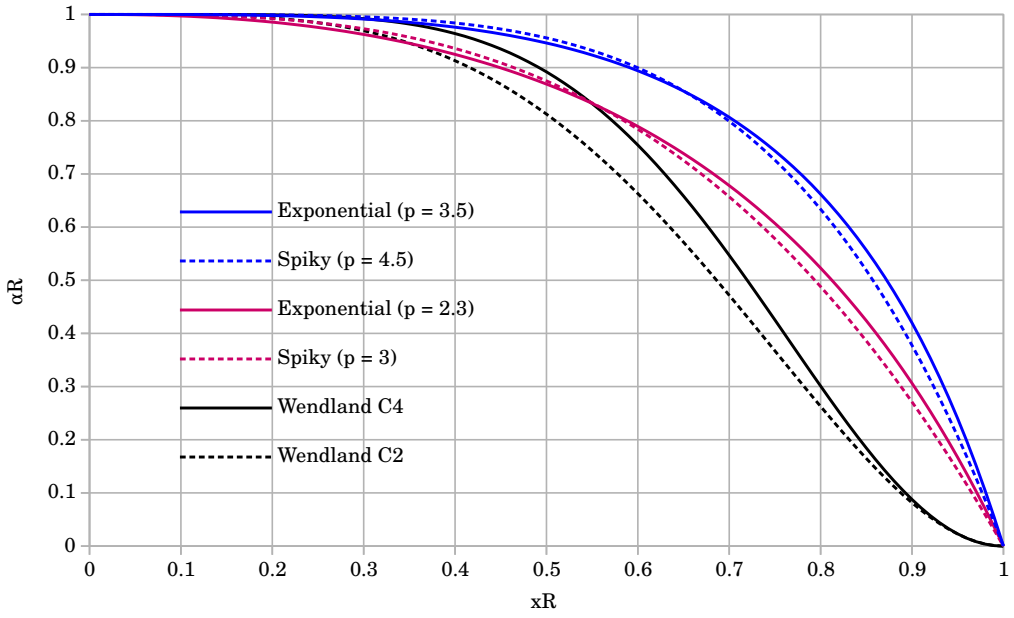

Figure 3.16.: A comparison of the relaxation weighting functions: the exponential weighting functions, Spiky kernels, and Wendland's kernels with various steepness.

The comparison of equations (3.102) and (3.103) is shown in figure 3.16. For the usual exponential weighting function, the Spiky kernel yields a similar curve if $\beta = 4.5$. Generally, relaxation zones require the domain to be extended. Similarly when choosing a meshless weighting function, for a relaxation zone it is questionable how the slight the transition $\alpha_r \to 1$ exhibited in the exponential weighting actually benefits the relaxation for the price of performance. By lowering the exponent β in the exponential and Spiky functions, the relaxation is hastened as shown in figure 3.16. Therefore, the relaxation

zone may be reduced by ca. 15%. The Spiky kernel Equation (3.103) with an integer exponent, $\beta = 3$ or $\beta = 4$, has a simpler form than equation (3.102) and much better computational performance. Figure 3.16 also includes two curves that represent Wendland's weighting functions, which belong to a class of bell–shaped functions. Bell–shaped functions, compared to the spiked functions, have zero–derivative when relaxing from the target to the computed solution. This implies that Wendland's and other bell–shaped weighting functions should be used in relaxation zones where the computed results are not ideally converging, e.g. at the start of simulation where waves are generated into the tank filled with still water.

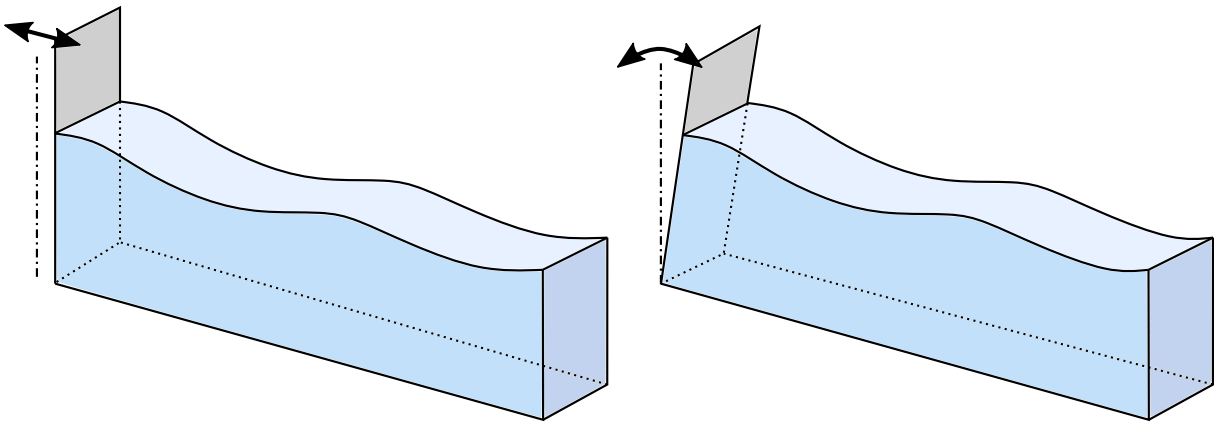

Figure 3.17.: Schematic of common physical wave maker types: piston–type (left) and flap–type (right).

3.9.5. Waves Generation

Calculating green water loads can be performed without the numerical generation of waves, by modelling the deck and by imposing the entry of water on the deck. Reliable wave generation procedure allows for the calculation of green water loads in more detail, i.e. the calculation contains the interaction between the hull form of a ship form and waves. This was described in section §2.7. There are multiple possibilities how to numerically generate waves to spread through the fluid domain.

- The waves can be generated by physically moving a (semi)immersed body. In experiments, there are two common types of the physical wave maker: the piston–type and flap–type, which are depicted in figure 3.17. Appendix H describes the flap–wavemaking theory that is used to impose the movement of a flap to generate the specific wave. This technique is suitable when trying to reproduce the wave generation similar to the experimental wave tank. The disadvantage is that the domain has to be extended in order to allow the wave generated from the flap movement to fully develop before reaching the structure under consideration. In ad-

dition, physical wave makers induce complex flow structures in their vicinity which introduce additional difficulties concerning time stepping.

- Another option is to impose the inflow boundary conditions from section 2.4.3 in combination with a relaxation zone. This takes care of prescribing velocities, pressures and waves amplitude using description methods of waves. At each time step, the description of waves is computed. Then the obtained solution is imposed at the inlet boundary as described in section 3.9.2. Before the Lagrangian advection, the computed flow is relaxed as described in section 3.9.4. The combination of the Dirichlet boundary condition and relaxation technique has proven to be provide adequate computational efficiency and low reflections of both short and long waves [178].
- Choi and Yoon [179] introduced an internal wave–making scheme based on the momentum conservation. The momentum source term, or an internal force, derived from the regular wave theories is added into the NSE. Therefore, waves are generated through the periodic application of internal forces within a specified virtual box. Like other schemes, this one also requires enough space for wave making as it generates an evanescent wave mode near the source region, similar to the piston–type wave maker. Details are given in Appendix I.

4. Implementation

In the following sections, it will be explained how the numerical methodology presented in chapter 3 is implemented to solve the mathematical problems presented in chapter 2. The implementation algorithm is given in section §4.1. Furthermore, section §4.2 and section §4.3 explain how the software implementation executes the solution algorithm efficiently on current hardware.

4.1. Solution Procedure

Although the solution procedure consists of steps which are implemented to execute in parallel as it will be shown later, the solution procedure itself is a sequential algorithm. The sub-steps of a time step are listed as follows:

1. Discard all generated boundary points for the previous time step.
2. Generate boundary points at inlet boundary surfaces using the technique described in section 3.9.2.
3. Generate boundary points at open boundary surfaces using the technique described in section 3.9.3.
4. Generate boundary points at solid surfaces, as illustrated in section §3.4.
5. Find neighbours between points, as described in section 4.2.2.
6. Calculate the renormalisation tensor of each point by equation (3.9) and the offset vector by equation (3.12).
7. Detect points on the free surface using the technique described in section §3.5.
8. Compute the diffusion term using the Laplacian approximation equation (3.36).
9. Set up the right-hand-side vector of the PPE (3.54) by evaluating equations (3.56) for fluid points, equation (3.59) for Neumann boundary points, and by setting values for Dirichlet boundary points.
10. Prepare the preconditioner and solve the PPE (3.54) by an iterative solver without constructing the system matrix as described in section 4.2.4.

11. Calculate the pressure gradient by equation (3.10).
12. Solve the momentum equation for the velocity of fluid points as described in section §3.7.
13. Advance points to their new locations as described in section 3.8.1.
14. Conserve the volume of fluid points by the iterative scheme described in section 3.8.3.
15. Remove points outside the domain.
16. Write results to a file and/or render them on the display as described in section 4.2.5.

4.2. Implementation Specifics

The implementation of the solver is made in a cross–platform manner and therefore can be executed on *Linux*, *Windows*, and *Mac OS X* operational systems. The solver library is made as a stand-alone library that can be used by some other application, and a Graphical User Interface (GUI) is made to visualise and interact with the simulation in real-time. The next few sections describe some of the important implementation details of the solver library and the GUI.

4.2.1. Simulation Set-up

The simulation input file does not incorporate any volumetric mesh, but only bounding surfaces of the domain and objects in the scene. The surfaces can be defined using primitives: list of connected line–segments, list of triangles, box, plane, etc. Alternatively, a surface can be imported into the scene as a list of triangles from the commonly used *Sterelithography* (STL) file format. *JavaScript Object Notation* (JSON) open–standard file format, that uses human readable text in form of attribute–value pairs and array data types, is used as an input for the solver. Detailed description of the file format and input examples are given in Appendix J.

The meshing process is present in the form of an automatic generation of the fluid point cloud. Generally, a body of liquid is surrounded by various boundaries: solid walls, virtual boundaries (inflow, outflow, symmetry), and the fluid free surface. The simulation input defines the geometry of those boundaries. With all relevant boundaries defined, source locations for fluid flooding must also be defined. The space is automatically filled starting from the defined flooding locations by generating points with defined initial spacing Δ, until nearest boundaries are reached. The concept of the flood filling is schematically rendered in figure 4.1. The free surface shape is used as a boundary surface, which is removed after the fluid is filled. Multiple source points can be defined, which then fill

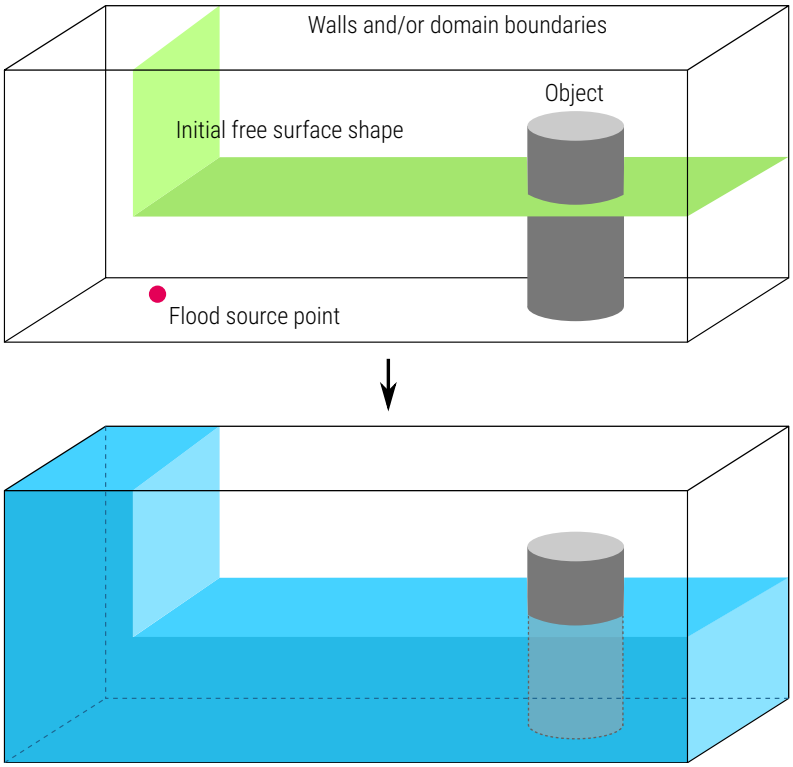

Figure 4.1.: The concept of flood filling algorithm, which fills the fluid domain bounded by boundary surfaces from a source point that is defined in the fluid domain.

some separate spaces with fluid as rendered in figure 4.2. For instance, this feature can be used to fill separate spaces of fluid for the dam–break experiment, where a moving wall releases high column of water. In future, the feature can be used to fill different fluids (e.g. water and air) into the specified domain.

4.2.2. Neighbour Search

Due to lack of the topology information in the meshless point cloud, each point must find its neighbours inside the compact domain at each time instant to perform the interaction. Background grids are usually employed to avoid testing of all points with each other to see if they are close enough to interact. Background grids, also known as acceleration grids, are used to tackle the computational efficiency issue of nearest neighbour search, by dividing the domain space and inserting the meshless points inside the cells. In such a way, any point has potential neighbours in its cell and the surrounding cells, i.e. all background cells that overlap with the compact domain of the point under consideration. The concept is schematically depicted in figure 4.3.

The Cartesian background grid was implemented within the solver by employing atomically built linked lists [180], which allow dynamic insertion of temporary boundary points into cells, needed by the idea presented in section §3.4. More information on the imple-

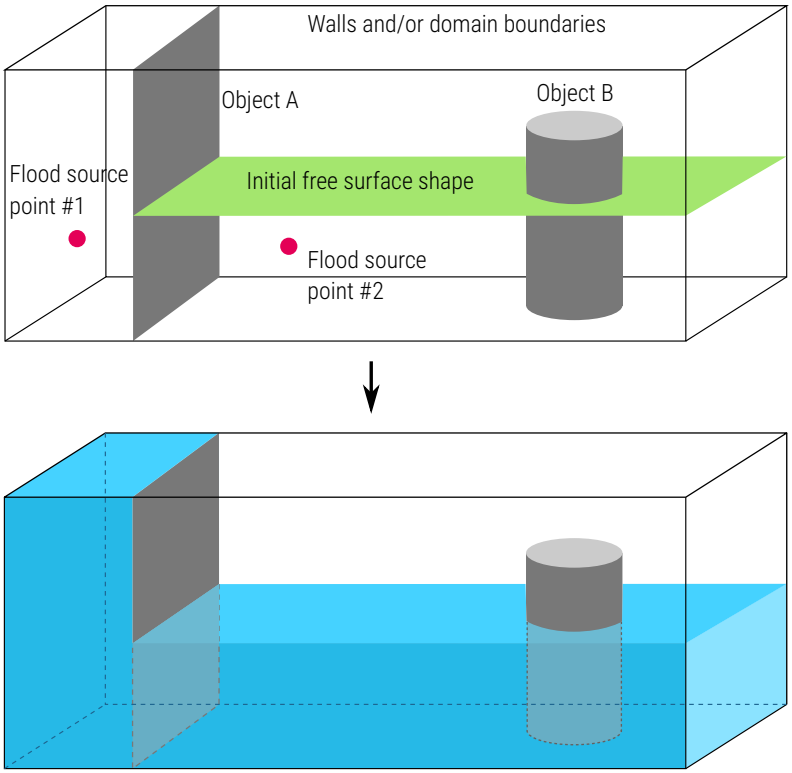

Figure 4.2.: The concept of flood filling algorithm, which fills the fluid domain(s) bounded by boundary surfaces from multiple source points that are defined in physically separated spaces.

mentation and background grids is given in Appendix E.

One should weigh the ratio of the grid cell size and compact sphere size, since it affects the performance of neighbour search. In order to avoid unnecessary accesses through the computer memory, the number of points in cells should not be too high, nor the number of overlapping cells with the compact domains should be to high. Empirically, it was found that the cell size of $2.1\,h$ yields adequate neighbour search performance.

4.2.3. Operations with Boundaries

This section explains how the interaction of fluid points with boundaries is implemented, since it is of crucial importance for the efficiency and robustness of the introduced methodology. Methods that work with many discrete elements that represent surfaces and volumes must employ data structures that enable optimised spatial operations related to the problems of ray–geometry intersecting, geometry–geometry overlap detection, etc. Filling the space with fluid points as described in section 4.2.1 concludes when points reach a boundary. At some location, it should be quickly evaluated whether the fluid can spread or it will hit the boundary that is represented by e.g. triangles in three dimensions. Therefore, a fast ray–triangle intersection algorithm defined by the direction of spreading is needed. Moreover, when the boundary points are needed to be generated as described

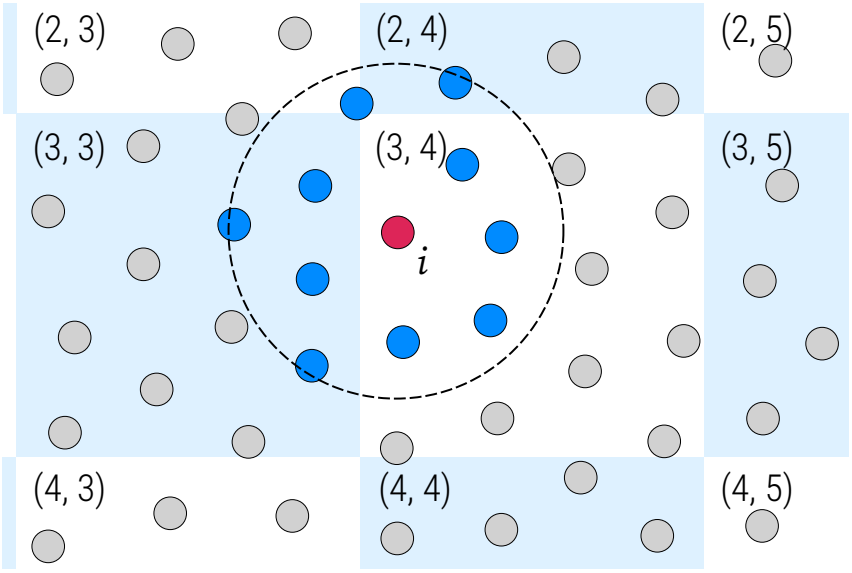

Figure 4.3.: A schematic concept of uniform background grid, where each point is put inside of a square cell indexed in a matrix fashion.

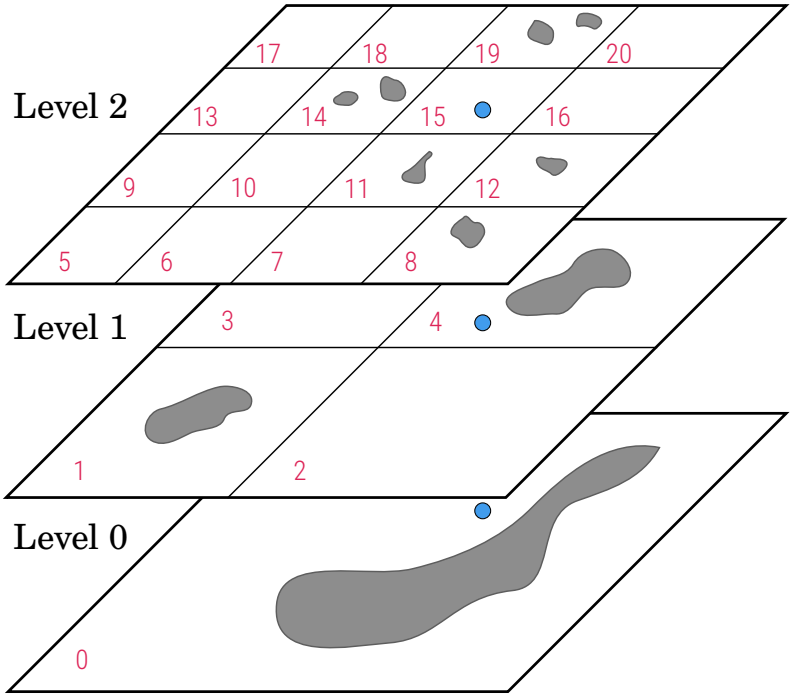

Figure 4.4.: A schematic concept of hierarchical background grid for boundary meshes, where triangles (gray) are put inside of a cell indexed in a tree fashion, and tested location (blue point) propagates to all levels.

in section 3.4.2, points on the edge of the point cloud should be quickly evaluate to check if they are positioned against a wall or another virtual boundary. This is achieved by projecting the points onto the nearby geometry.

If the problem under consideration is described by small amount of triangles, then the necessary test can be performed simply by iterating through the triangles forming the

boundary and testing each triangle. The information that described the geometry takes up a small region in the Random Access Memory (RAM) and may be cached at all time. The pseudo–code is given in the following text.

```
for_each (point in relevant_points):
  for_each (triangle in triangles_list):
    projection = project(point.location, triangle)
    if (distance(point.location, projection.location) < min_distance):
      generate_boundary_point(projection, point) # new and original point
connectivity
```

When the problem contains complex geometry described by large amount of triangles, hierarchical data structures known as "spatial data partitioning trees" are often employed to speed up access to only relevant triangles in the specific volume. Each object may have its own spatial partitioning structure, which remains unchanged while geometry of the object is unchanged, irrelevant of the object translation and rotation. The test being performed for such structure firstly transforms global coordinates of a location vector into local coordinates of the partitioning structure. GPU implementations of sparse data structures are problematic, because updating sparse structures usually involves scattered writing to memory addresses, and traversing involves scattered pointer de-references to access the data. In this thesis, a static tree-like structure is employed where each cell points to the first element inside it, similar to the implementation of nearest–neighbour searching. The idea is depicted in figure 4.4, in which the blue point describes a location being tested for elements by searching cells on multiple hierarchical levels. The pseudo–code considering this space partitioning approach is given in the following text.

```
for_each (point in relevant_points):
  for_each (object in objects):
    local_location = object.transform(point.location)
    triangles_list = object.get_triangles_in_cells(local_location)
    for_each (triangle in triangles_list):
      projection = project(local_location, triangle)
      if (distance(local_location, projection.location) < min_distance):
        generate_boundary_point(projection, point) # new and original point
connectivity
```

4.2.4. Linear–System Solver

The pressure equation is solved iteratively by a Krylov subspace (KSP) method, which is implemented using the matrix–free multiplication kernel without keeping the sparse matrix in memory. Instead of preparing the system matrix before solving, the matrix-row–vector multiplication is repeatedly done when needed. By an empirical evaluation, it was found that performing these mathematical operations in the kernel is faster than

preparing the system matrix before solving. The following iterative methods were implemented and are shown to converge when solving the PPE:

- Conjugate Gradient (CG) method,
- Bi–Conjugate Gradient Stabilized (BiCGStab) method [181], which is a stabilised version of Bi–Conjugate CG, naturalised generalization of the classical conjugate gradient algorithm for Hermitian positive definite matrices to general non-Hermitian linear systems,
- Generalized Minimal Residual (GMRES) method [182], which uses Arnoldi iteration is used to find the vector in a Krylov subspace with minimal residual,
- Transpose–Free Quasi–Minimal Residual (TFQMR) method [183], which overcomes the problems of Bi–Conjugate CG by a look-ahead version of the non-symmetric Lanczos algorithm.
- Quasi–Minimal Residual Conjugate Gradient Stabilized (QMRCGStab) method [184], which is a QMR variant of the BiCGStab method that yields smoother convergence behaviour.

The solvers converge if and only if the boundary conditions are properly posed, which is thoroughly explained in section 3.6.2. It may seem peculiar that the CG method, which is applicable for symmetric system matrices, also works for these cases that are characterised by non-symmetric matrices. This confirms the robustness and stability of the introduced method, although the simulation quickly explodes when the CG solver does not properly converge. The diagonal preconditioner was used to improve the performance and convergence of KSP methods [185]. The QMRCGStab method was found to be the best compromise between efficiency, smoothness of the convergence, and number of iterations needed. Therefore, if not otherwise specified, the solver based on the QMRCGStab method is used in the simulations recorded in this thesis. More information on KSP methods and their optimised implementation is given in Appendix D.

4.2.5. Postprocessing

There are two options on how to examine results of a simulation results: by post-processing exported data or by visualising the results in real-time. The solver can be used as:

- a dynamically linked library that is called from another software,
- a console application with defined input and output parameters,
- a desktop application with a real-time GUI.

The need for creating a GUI arose from the user experience feedback to help users simulate their cases with ease. The application with the GUI was made with the help of *Qt* cross-platform application framework (`www.qt.io`), and offers:

- an interface for setting up a simulation,
- controls for the simulation running process,
- real-time visualisation of the results while the simulation is running.

These features of the application are shown in figure 4.5. *OpenGL* rendering pipeline (`www.opengl.org`) was used for drawing the simulated scene while the solver is running. Meshless point cloud is drawn as a set of spheres by using the pipeline "geometry shaders" for quickly instancing millions of spheres on the screen. The implementation method is given in Appendix F. A simulation scene rendered inside the GUI application is shown in figure 4.5. The bottleneck of the rendering is the transfer of the solver data from the computing device to the GPU. If the solver is being run on the GPU, then the data does not need to be transferred, but is directly used with the help of *CUDA*/*OpenGL* interoperability framework.

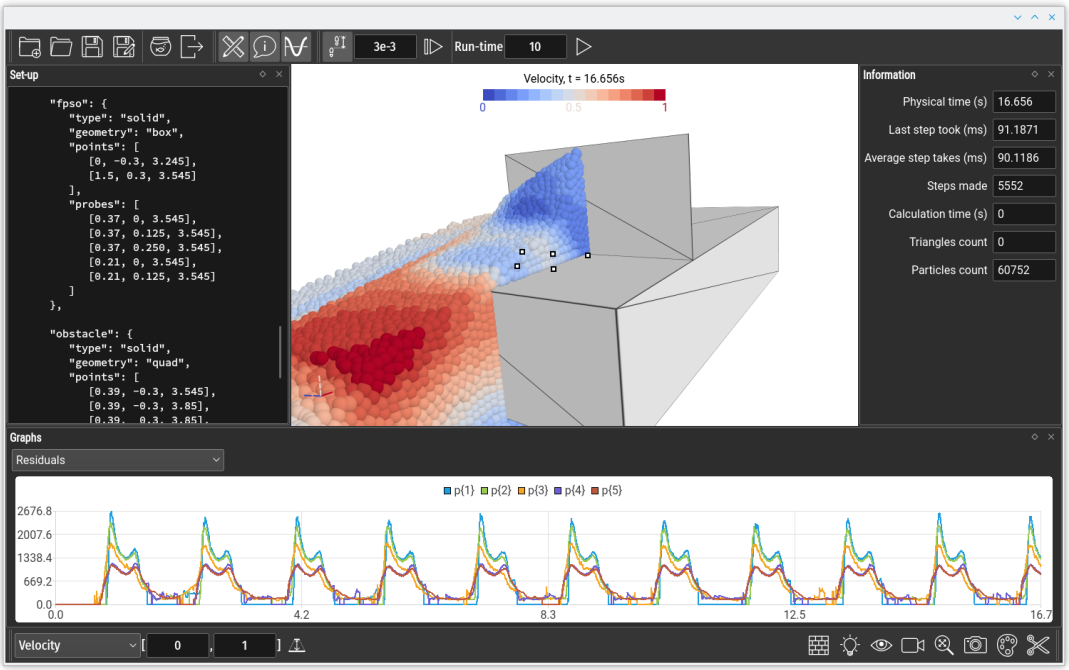

Figure 4.5.: The graphical user interface of the implemented solver.

Even though the GUI application can immediately render the simulation as it progresses, usually users require more detailed inspection of the results in some software specialised for CFD post-processing. Therefore, the simulation input specifies the frequency and other options for exporting the results. The results are written in *Visualization Toolkit* (VTK) file format [186]. *ParaView*, which is a post-processing, data analysis and visualisation application (`www.paraview.org`), was used to inspect the exported data-set of the results in an non-real-time manner.

4.3. High Performance Computing

This section presents the reasoning for the current implementation. Generally, High Performance Computing (HPC) is the aggregation of computing power that delivers higher performance than it is possible to get out of a typical desktop computer or workstation, and is employed to solve large problems in science, engineering and business. HPC is the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term HPC is occasionally used as a synonym for supercomputing, although technically a supercomputer is a system that performs at or near the currently highest operational rate for computers. A cheaper alternative to a expensive supercomputer is a cluster, which is a group of computers considered as a single machine, connected through a high speed network. Clusters can be extended by adding more computers into the group. A short analysis of the current status of hardware is presented in Appendix G.

4.3.1. Parallelism in Computing

HPC includes various techniques of parallel and distributed computing. Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. The forms of parallel computing are: bit–level, instruction–level, data, and task parallelism. Distributed computing is a type of computation in which separate computers, called nodes, coordinate their actions by passing messages, i.e. they interact with each other in order to achieve a common goal, thus achieving computing concurrency. A node in a distributed system can also use parallel computing techniques for its part of the work.

This work does not deal with the optimised execution of the solver on a distributed system or a supercomputer. In order to show the feasibility of the method for everyday engineering use, this thesis rather deals with the problem of fast execution on a single personal computer or a workstation. When the implementation works efficiently on a single computer, it is straightforward to extend its execution on a distributed system via Message Passing Interface (MPI), a library of functions that enables communication between the nodes in a cluster. The problem described in section §4.1 needs to be numerically solved as fast as possible. Contrary to task parallelism problems, which divide the work into different jobs that execute at once, the problem presented here needs to be solved by steps in sequential order, since one step depends on the solution of the former step. The proper technique to assess the problem is to deal with each step by utilising data parallelism, i.e. each processor performs the same task on different pieces of distributed data. In some situations, a single execution thread controls operations on all pieces of data, e.g.

in Graphical Processing Units (GPUs). In other situations, different threads control the operation, but they execute the same code, e.g. in Central Processing Units (CPUs).

4.3.2. Parallel Implementation

By analysing the status of the current hardware, presented in Appendix G, it is obvious that scientific software should perform efficiently on massively parallel architectures. Given the pending convergence of massively parallel architectures, there is little reason to worry about a particular architecture, i.e. if the right algorithms are used, one will be well prepared for any upcoming architecture. For this reasons, the implementation of the methodology, i.e. the pre-processor, solver and post-processor, was programmed in C++ programming language using data–parallel algorithms. After trying and testing various programming libraries that help with data–parallel programming, a HPC library named *Kokkos* [187] was chosen. The library implements a programming model in C++ for writing performance portable applications targeting all major HPC platforms. For that purpose, it provides abstractions for both parallel execution of code and data management. It currently can use system threading, *OpenMP*, *CUDA*, and *ROCm* as backend programming models, and employ MPI for distributed computing. In conclusion, the employed programming models enable the solver to be executed efficiently on multi-core devices (CPUs, GPUs, and other similar acceleration devices) in an optimised manner, adjusted appropriately for each architecture that the solver is running on.

5. Verification and Validation

Verification and validation (V&V) are the primary means to assess accuracy and reliability in computational simulations, i.e. their goal is to ensure that a methodology implementation produces reasonable results for a certain range of problems. Verification is the process of testing the correctness of a mathematical model that describes a specific physical phenomena. It answers to a question whether the equations are solved in the right way. Verification is usually done by comparing the computational results with analytical and numerical solutions for standard benchmark configurations, i.e. representative test cases. Validation is the process of testing the correctness of the description of the actual problem with the proposed model and methodology. It answers to a question whether the right equations are being solved for the specific problem by means of comparing the results with available numerical and experimental data. In the next sections, the numerical methodology introduced in chapter 3 and its implementation described in chapter 4 are verified and validated.

5.1. Novel Laplacians

5.1.1. Approximation

The rate of convergence, the error, and the computational efficiency of the introduced operators will be demonstrated in this section. The features of the operators are compared to the corresponding regular FD five–point stencil formulations and to the typically used SPH scheme. In a unit square, Franke's bivariate function has two Gaussian peaks of different heights and a sharper depression, which is superimposed on a surface sloping towards the first quadrant [188]. It is used as a standard test function for two-dimensional scattered data fitting. The function is defined as:

$$\begin{aligned} f(\boldsymbol{x}) = {} & 0.75 \exp\left[-(9x-2)^2/4 - (9y-2)^2/4\right] \\ & + 0.75 \exp\left[-(9x+1)^2/49 - (9y+1)/10\right] \\ & + 0.5 \exp\left[-(9x-7)^2/4 - (9y-3)^2/4\right] \\ & - 0.2 \exp\left[-(9x-4)^2 - (9y-7)^2\right]. \end{aligned} \tag{5.1}$$

Total root-mean-square error (RMSE) normalised with the ℓ^2 norm of the vector that contains exact values is used as a measure of the accuracy of the obtained numerical results, which is defined as:

$$\text{Error} = \frac{\sqrt{\sum_i \left[\langle \nabla^2 f \rangle_i - \nabla^2 f\left(\boldsymbol{x}_i\right)\right]^2}}{\sqrt{\sum_i \left[\nabla^2 f\left(\boldsymbol{x}_i\right)\right]^2}}, \tag{5.2}$$

where $\nabla^2 f\left(\boldsymbol{x}_i\right)$ is obtained analytically from equation (5.1). Equation (5.2) is used as a relevant measure due to its sensitivity to the discrepancy, while its value is presented relatively to the global situation.

5.1.1.1. Regular Point Arrangement

Firstly, tests were conducted for regular point arrangements, where the points were placed on Cartesian grid nodes with uniform spacing Δ. Six various resolutions were used to study the convergence, $\Delta = \{0.1,\ 0.05,\ 0.025,\ 0.0125,\ 0.00625,\ 0.003125\}$. Wendland's C^2 quintic kernel [189] is used as a smoothing function, which is defined as:

$$W\left(r, h\right) = W_0 \cdot \begin{cases} \left(1 - r/h\right)^4 \left(4\, r/h + 1\right) & 0 \leq r < h \\ 0 & h \leq r \end{cases}, \tag{5.3}$$

where the absolute normalisation coefficient W_0 is irrelevant due to the normalisation. The results of the tests conducted with two smoothing radii, $h = 1.5\Delta$ and $h = 2.5\Delta$, are shown in figure 5.1 where the values of errors are shown on a log-log scale as functions of the particle spacing. All meshless Laplacians perform the same as the second–order accurate FD method when the neighbour points around the i point form the same arrangement as the FD stencil does. When the smoothing radius expands to include multiple rings of neighbour points, then the discrepancy between the meshless and FD method increases, but the second–order slope is preserved and discrete results converge to the analytical solution as $h \to \Delta \to 0$. Numerical errors for $h = 2.5\Delta$, which are plotted in figure 5.1, show that the discrepancy depends on the level of neighbour averaging. Similar to the interpolation and gradient calculation, this implies that the form of the utilised smoothing function has an impact on the accuracy, and that it can be designed to sufficiently lower the error. The introduced Laplacians and the gradient operator are given in FD form. Therefore, when a non-linear function is considered, similar accuracy sensitivity connected with the choice of smoothing function is expected. The superiority of a meshless method is its natural ability to form arbitrary point cloud arrangements. The importance of the smoothing function will therefore be shown in the next section that considers the scattered arrangement of points.

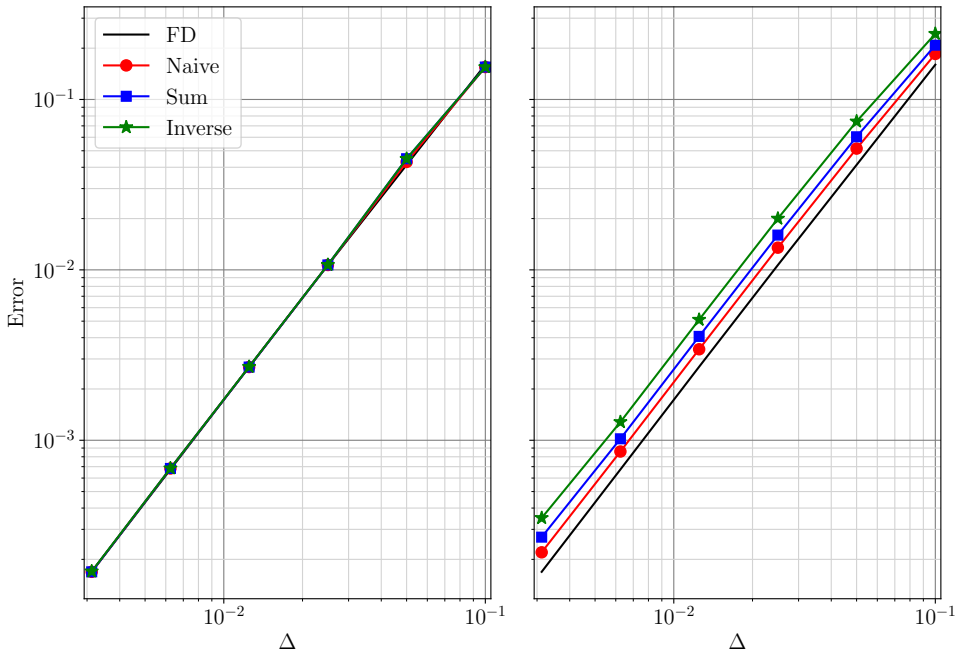

Figure 5.1.: Relative errors of approximate discrete Laplace operators applied to Franke's function and a regular point arrangement, for $h = 1.5\Delta$ (left graph) and $h = 2.5\Delta$ (right graph).

5.1.1.2. Scattered Point Arrangement

In this section, the rate of convergence and computational efficiency of approximate Laplacian operators are examined for scattered or irregular arrangements of points. The scattering of the locations of points is done after the points are placed on a regular grid with uniform spacing Δ, as described in the previous section. Each point is randomly shifted by a vector $\boldsymbol{e}_i c\Delta/2$, where c is the measure of chaos or randomness, and $\boldsymbol{e}_i$ is the spatial vector whose components are random values between –1.0 and 1.0. Evidently, for $c = 100\%$ two neighbouring points spawned on a regular grid can be shifted in such a way that their new locations coincide. Conversely, shifting points can move them farther away, so the minimum smoothing radius should be controlled to keep the nearest neighbours inside the compact radius after the shift. In order to keep diagonal neighbours after the shift, to provide each point with a sufficient number of encircling points, the compact radius value should exceed:–

$$r_{min} = \sqrt{d}\Delta\,(1 + c)\,. \tag{5.4}$$

Before going into detail about each of the introduced Laplacian formulations, a proper smoothing function should be chosen. It follows from equation (3.31) that the naive version of an approximate Laplacian is sensitive to the error of close neighbours, scaled by their weight and inverse squared distance. As an example, the image on the left in figure 5.2 shows a coarse Voronoi representation of the Laplacian values of the test function obtained with the naive version using the smoothing function expressed by equation (5.3),

for $\Delta = 0.05$, $h = 2.7\Delta$ and $c = 90\%$. The error defined by equation (3.31) can be leveraged by modifying ψ_{ij} so that closer points have less influence. Therefore, a compromise smoothing function can be introduced, where the near points contribute maximally, and the farthest neighbour points do not contribute excessively, since it would produce a surpassingly averaged result for functions that non-linearly change inside the smoothing area. In conclusion, a new function of a blunt shape is given as:

$$W(r,h) = \begin{cases} 1-(r/h)^8 & 0 \leq r < h \\ 0 & h \leq r \end{cases}. \tag{5.5}$$

It should be noted that equation (5.3) is still used for the evaluation of the gradient, while equation (5.5) is used for equation (3.30). Functions given with equations (5.3) and (5.5) are plotted in figure 5.3, and an improved result that employs both equations is shown in the image on the right in figure 5.2. The sum version does not use a precalculated gradient, but a renormalisation tensor and offset vector are used in equation (3.36). Therefore, the sum version uses a single smoothing function, which should be designed so that the operator performs adequately. It was empirically found that an example of such a function is the spiked symmetrical function defined by the following equation:

$$W(r,h) = \begin{cases} 1-(r/h)^{0.4} & 0 \leq r < h \\ 0 & h \leq r \end{cases}. \tag{5.6}$$

It was found that both naive and inverse versions are somewhat sensitive to the shape of the smoothing function for the gradient. The inverse version uses a combination of equations (5.3) and (5.6) for the gradient and Laplacian expressions, respectively. It should be noted that the introduced smoothing functions are designed empirically to prove the adequate performance of the derived discrete operators, and that their optimal variants are topics for future work.

Numerical tests were performed for the three levels of randomised scattering, $c = \{30\%, 60\%, 90\%\}$. Each test was repeatedly executed minimally twenty times, and the average value of the errors calculated with equation (5.2) for each execution is used as the measure of accuracy. Figures 5.4, 5.5 and 5.6 show the obtained relative errors of the approximate discrete Laplace operators applied to equation (5.1) for levels of scattering $c = 30\%$, $c = 60\%$, and $c = 90\%$, respectively. It can be noticed that the numerical error on the coarse distributions is generally lower for a smaller smoothing radius, and that the error increases with an increase in the smoothing radius. The same is noticed for regular point arrangements, which implies that far neighbour points on a highly non-linear changing area are not desirable candidates for the approximation. Nevertheless, equation (5.4) should be satisfied, which is challenging for dynamical problems in practice where the

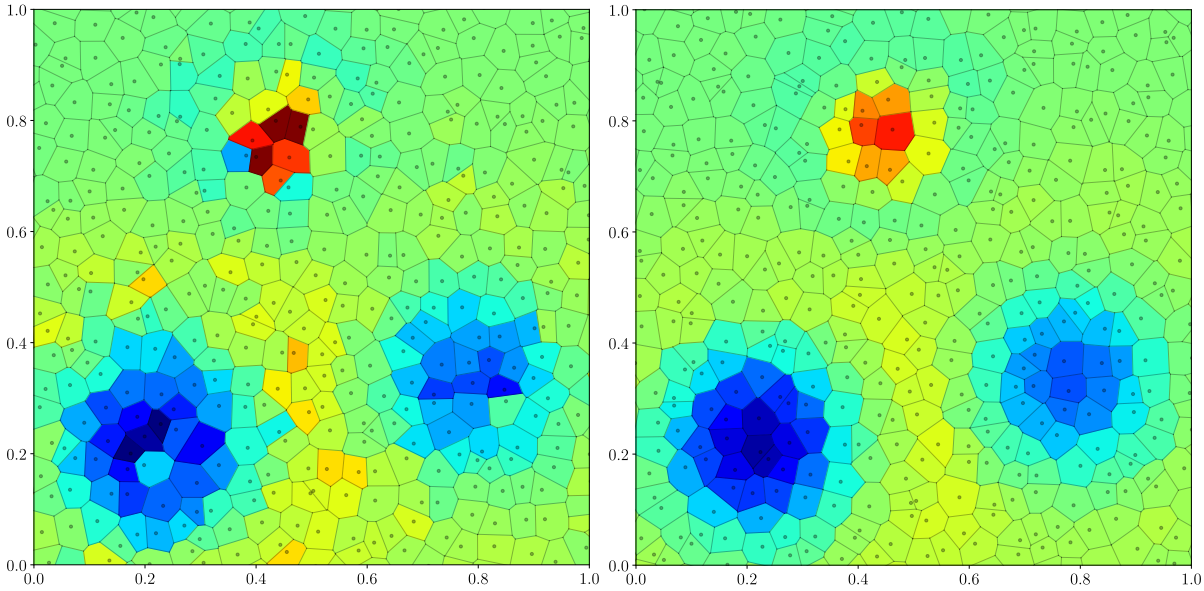

Figure 5.2.: Qualitative representation of the Laplacian naive version values for $\Delta = 0.05$, $h = 2.7\Delta$ and $c = 90\%$, obtained with the default smoothing function (left image), and modified smoothing function (right image).

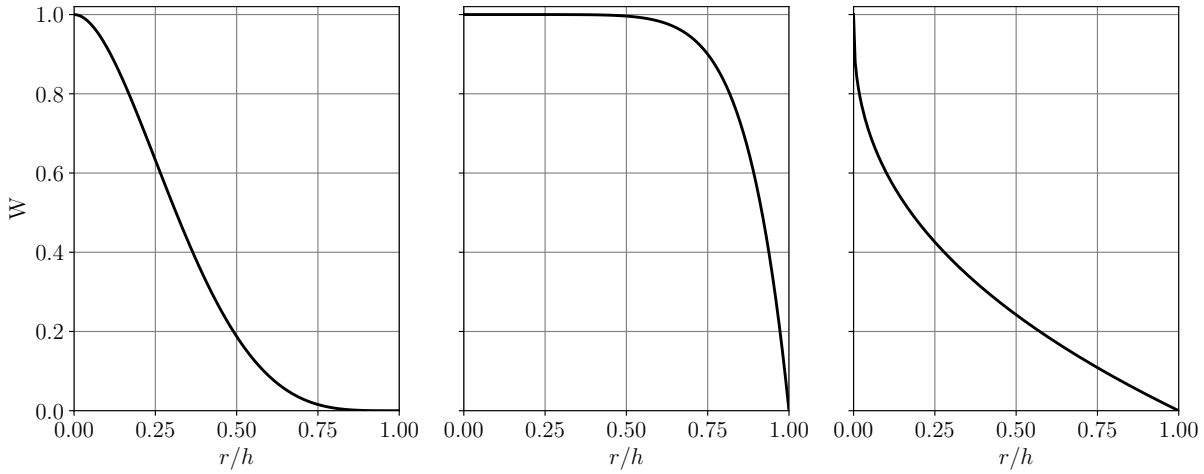

Figure 5.3.: Smoothing functions: Wendland's kernel (left graph), blunt function (centre graph), and spiked function (right graph)

scattering level of points is high, often $c > 30\%$. Figure 5.4 shows that the error slope for relatively low chaotic scattering is almost second–order as observed for the regular arrangement, and evidently smaller errors are obtained with larger smoothing radii as $\Delta \to 0$. For the sake of comparison, SPH Laplacian results are also graphed to present its divergence property (for a more detailed review, interested readers are referred to [134]). Figures 5.5 and 5.6 show results for high levels of irregular scattering and present a similar trend to figure 5.4, i.e. results converge when $\Delta \to 0$ and $h/\Delta \to \infty$. For the most chaotic case shown in figure 5.6, the error slopes of the tested Laplacians still obey the first–order convergence rate, until they reach a minimum error calculated with

equation (5.2). The difference between the error expressions of the naive and the sum version were analysed in section §3.3, but they still seem to offer a similar performance for any combination of a randomness and smoothing radius. This can be attributed to the fact that equation (3.31) can diminish with the proper choice of two smoothing functions, and that equation (3.37) cannot diminish as desired since a single trade-off smoothing function is used. The basic inversion version of the Laplacian achieves better results, and has a longer convergence slope in any case for the price of an additional tensor inversion computation. Readers interested in an analysis of the full inversion version of the discrete Laplacian are referred to [126], where the authors present results for the scattered arrangement of points perturbed with a Gaussian distribution with $\sigma = 0.1$. For small values of Δ, a maximum relative error of the full inversion is moderately lower than that of the basic inversion version. In this thesis, the levels of chaotic irregularity $c < 30\%$ are not considered in detail, because dynamic point clouds in practice are rarely regular in such an amount.

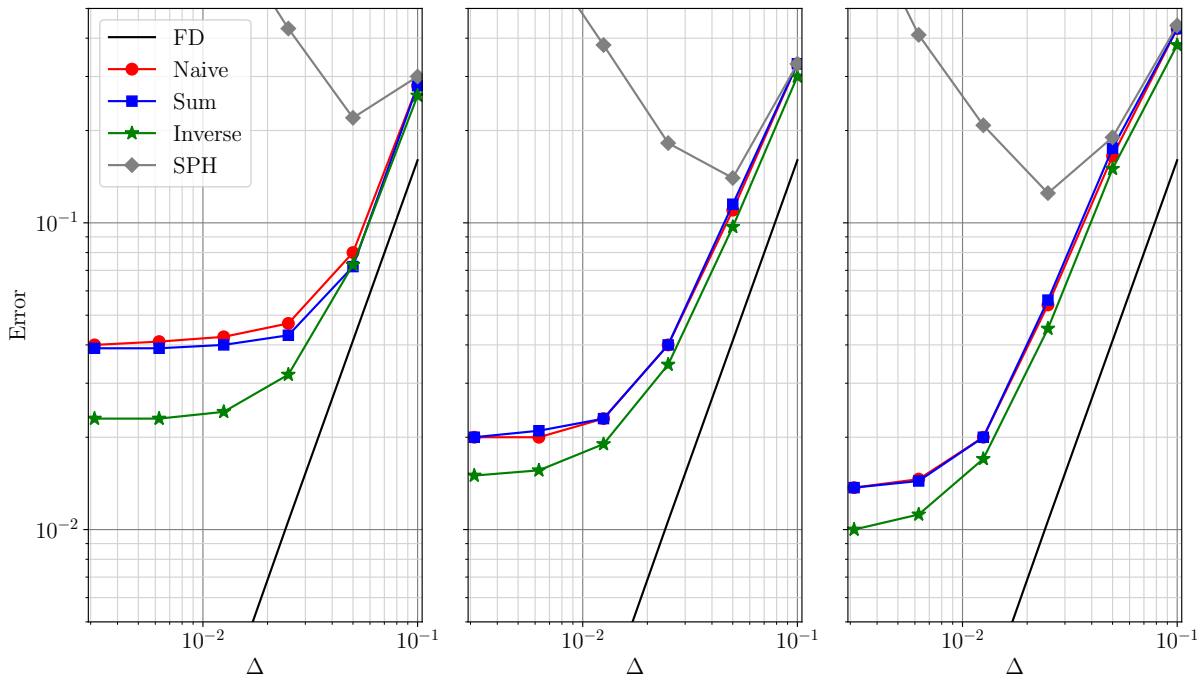

Figure 5.4.: Relative errors of approximate discrete Laplace operators applied to Franke's function and a scattered point arrangement $c = 30\%$, for $h = 2.0\Delta$ (left graph), $h = 2.7\Delta$ (centre graph) and $h = 3.5\Delta$ (right graph).

5.1.1.3. Computational Efficiency

Numerical tests were implemented in *Python 3* scripting language. Table 5.1 presents the relative time needed to evaluate the Laplacian value for a single interpolation point. The results are normalised with respect to the measured time of the execution of the

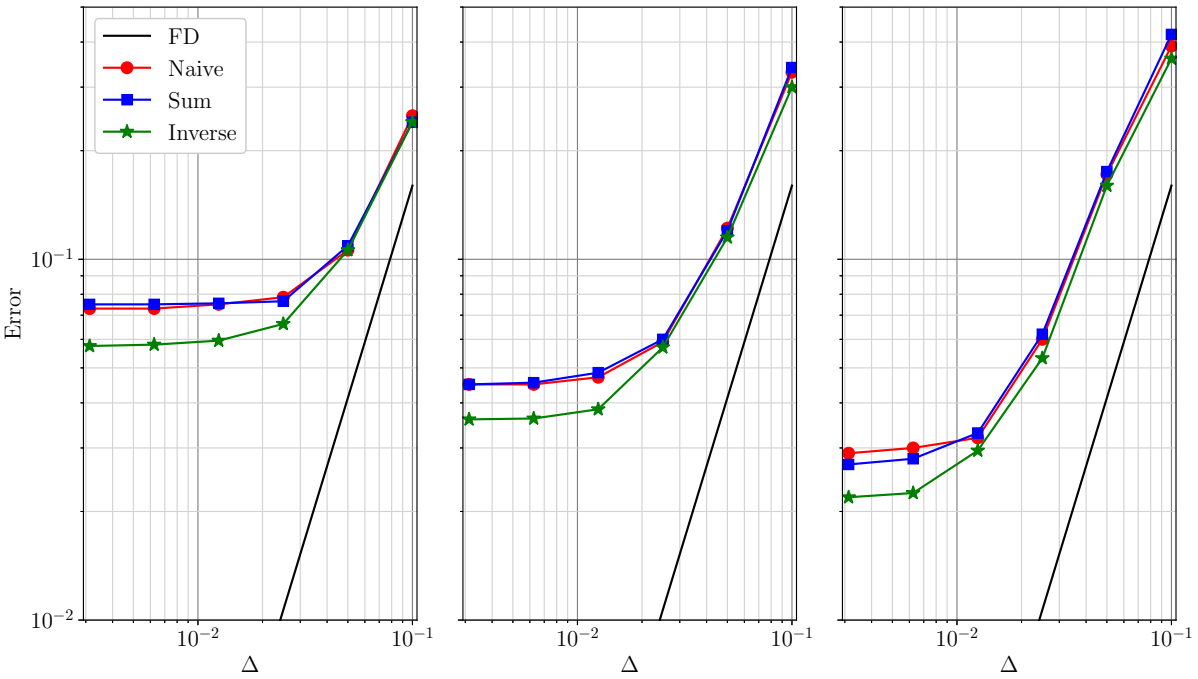

Figure 5.5.: Relative errors of approximate discrete Laplace operators applied to Franke's function and a scattered point arrangement $c = 60\%$, for $h = 2.2\Delta$ (left graph), $h = 2.7\Delta$ (centre graph) and $h = 3.5\Delta$ (right graph).

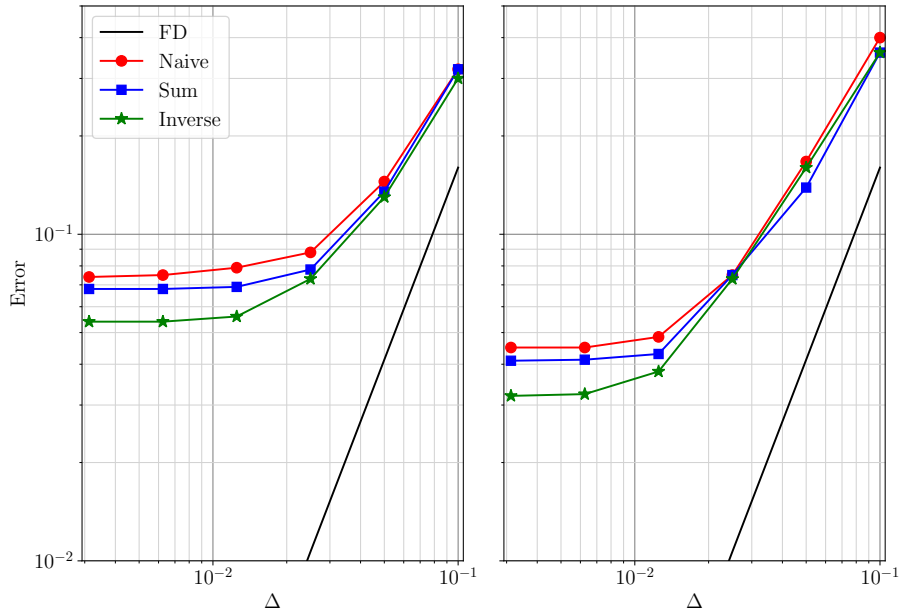

Figure 5.6.: Relative errors of approximate discrete Laplace operators applied to Franke's function and a scattered point arrangement $c = 90\%$, for $h = 2.7\Delta$ (left graph), and $h = 3.5\Delta$ (right graph).

naive version of Laplacian approximation, done with $h = 2.0\Delta$. It can be seen that the sum version is the fastest among formulations derived in this thesis, which is due to the fact that the renormalised gradient is not required within the calculation in contrast to

the naive version and versions that include additional tensor inversion. As expected, the classical SPH method is even faster since it does not use renormalisation techniques, but direct summations. If the renormalised offset vector $\mathbf{B}_i \boldsymbol{o}_i$ is known, the sum version can be evaluated about as fast as the SPH Laplacian.

h	Naive	Sum	Basic inverse	Full inverse	SPH
2.0Δ	1.00	0.82	1.17	1.72	0.23
2.7Δ	1.68	1.30	2.09	2.87	0.38
3.5Δ	3.01	2.30	3.62	4.50	0.69

Table 5.1.: Relative CPU time needed for an evaluation of the approximate Laplacians.

5.1.1.4. A Remark on the Effect of Boundaries

James Clerk Maxwell proposed to call $\nabla^2 f(\boldsymbol{x})$ the concentration of f at the point $\boldsymbol{x}$, because it indicates the excess of the mean value in the neighbourhood of $\boldsymbol{x}$, over the value at that point. The Laplacian can be interpreted up to the scaling factor, as an operator that computes the rate at which the average value of f on a sphere's surface centred at $\boldsymbol{x}$ deviates, as the sphere grows. If the $f(\boldsymbol{x})$ is C^2 in the neighbourhood of $\boldsymbol{x}$, from the Taylor series expansion (3.5), the following can be proven:

$$\begin{aligned} \nabla^2 f(\boldsymbol{x}) &= \lim_{r \to 0+} \frac{2d}{r^2} \frac{1}{A(\mathcal{S}_r)} \int_{\mathcal{S}_r} \left(f(\tilde{\boldsymbol{x}}) - f(\boldsymbol{x}) \right) \mathrm{d}A(\tilde{\boldsymbol{x}}) \\ &= \lim_{r \to 0+} \frac{2d}{r^2} \left(f_A - f(\boldsymbol{x}) \right) \end{aligned} \tag{5.7}$$

where $\mathcal{S}_r$ is the geometry of the $(d-1)$-dimensional sphere of radius r located in $\boldsymbol{x}$, and the term A is the surface area of the sphere. This interpretation of the operator states that the correct evaluation of equation (5.7) requires the existence of field values around $\boldsymbol{x}$, i.e. evaluation of the discrete Laplacian requires surrounding neighbour points, which is not fulfilled near the boundaries. A common treatment for the problem is to introduce ghost points that carry extrapolated field values, or to modify discrete operator expressions that implicitly take extrapolated values into account. The extrapolation procedure in any case cannot yield fully accurate values since it depends on the derivatives calculated without the missing neighbour points.

5.1.2. Boundary Value Problems

In this section, Poisson and diffusion equations are solved in a strong formulation and the results are analysed. The equations are solved in two and three dimensions, on regular and irregular domains. Discrete Laplacians that are covered in this thesis are generally

described with a summation of finite differences, and therefore the problem is described and solved straightforwardly in a strong formulation. A linear system is built by transforming a summation expression for each point i, e.g. $\langle \nabla^2 f \rangle_i = \nabla^2 f(\boldsymbol{x}_i)$, into a matrix row and by setting known values for boundary points to impose the Dirichlet boundary condition. Analogous to the approximation analysis in section §5.1.1, the relative error of a solution is given by the following expression:

$$Error = \frac{\sqrt{\sum_i \left[\langle f \rangle_i - f(\boldsymbol{x}_i)\right]^2}}{\sqrt{\sum_i \left[f(\boldsymbol{x}_i)\right]^2}}, \tag{5.8}$$

where $\langle f \rangle_i$ is the numerically computed solution at the location of i-th point, and $f(\boldsymbol{x}_i)$ is the exact solution obtained analytically. The tests described in section §5.1.1 have verified that the introduced Laplacians yield the same results as the FD method for a sufficiently small smoothing radius. Indeed, the solutions of the Poisson equation on a regular grid will follow the same trend. Thus, the tests on scattered point arrangements will be analysed in the following text.

5.1.2.1. Square Domain with Dirichlet BC

The forcing term of the Poisson equation is a scalar field described by the analytical Laplacian of equation (5.1), and the solution is found for a unit square domain with Dirichlet's boundary condition. Figure 5.7 shows solutions on a Voronoi tessellation for the purpose of a qualitative review of the sum version of the Laplacian, where the tests were done for points with an irregularity level of $c = 60\%$, and a smoothing radius $h = 2.7\Delta$, for two different resolutions $\Delta = \{0.05, 0.025\}$. The trend of the exact solution is preserved in both cases. Moreover, the results show that even divergent approximate Laplacian formulations used within the SPH method weakly converge to the exact solution. Figure 5.8 shows relative errors of the Poisson equation solution obtained with the direct solver, for three levels of scattering and smoothing radii, $c = 30\%$ with $h = 2.0\Delta$, $c = 60\%$ with $h = 2.7\Delta$, and $c = 90\%$ with $h = 3.5\Delta$. The errors of the solutions when employing the introduced discrete operators are compared to the FD method on the regular grid, and the KGF–SPH method, since its Laplacian shows some similarity to the sum version [142]. Firstly, it should be noticed that the KGF–SPH method quickly reaches the constant convergence slope, meaning that the diverging curve shown in figure 5.4 blocks further convergence on finer resolutions, i.e. even linearisation by decreasing the particle spacing does not improve the results. Although the inversion version perform better than the sum version for the performed approximation tests, larger discrepancies are yielded for the Poisson–equation solutions when compared to the sum version results on highly irregular arrangements. In contrast, it is still slightly more accurate for the $c = 30\%$ level

of irregularity. This non-synchronised behaviour of the inversion version of the Laplacian may be attributed to the non-optimal choice of the smoothing function. The sum version of the discrete Laplacian shows improved performance in any case, and its computational efficiency is covered in section 5.1.2.5.

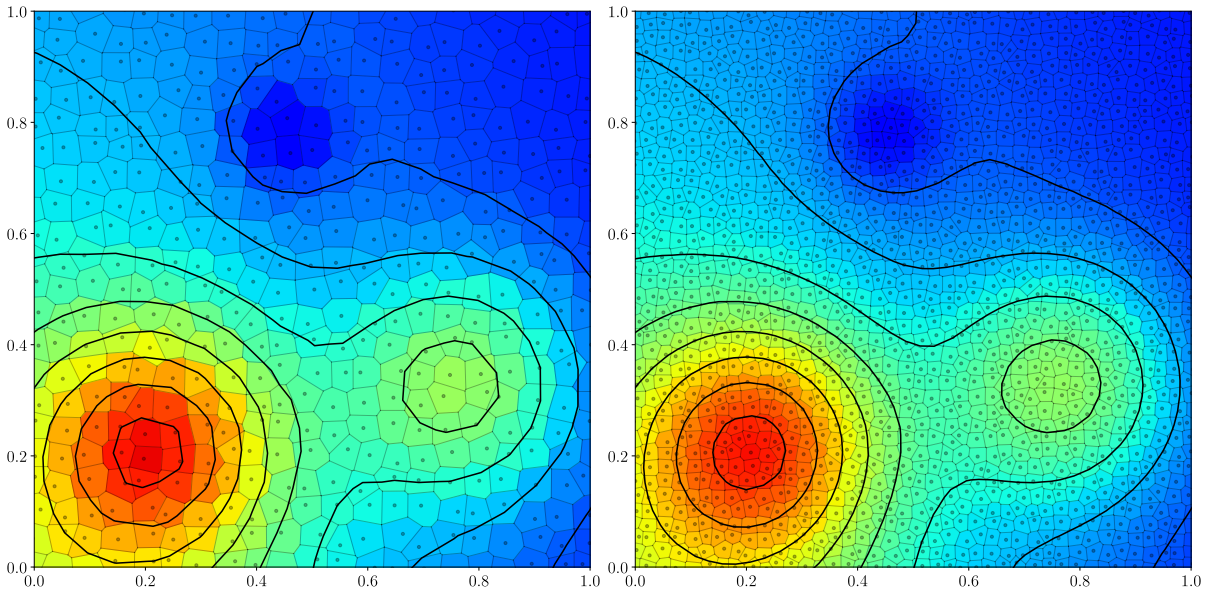

Figure 5.7.: The Poisson equation solution for the Laplacian sum version, $c = 60\%$, and $h = 2.7\Delta$. The solution for the coarser resolution $\Delta = 0.05$ is shown on the left, and the finer resolution $\Delta = 0.025$ is shown on the right. Thick curves represent qualitative contour levels of the exact solution.

5.1.2.2. Irregular Domain with Dirichlet BC

A diffusion problem $\nabla \cdot (\beta \nabla u) = f$ taken from [190] is considered, which is defined with a variable diffusion coefficient given as $\beta(x, y) = 2 + \sin(xy)$ and the exact solution of the problem given as $u(x, y) = e^x (x^2 \sin y + y^2)$. The Dirichlet BC is applied to the irregular boundary with the geometry of a star–shaped function parameterised by $(x(\theta), y(\theta))$, where $x(\theta) = \cos(\theta)[0.5 + 0.2\sin(5\theta)]$ and $y(\theta) = \sin(\theta)[0.5 + 0.2\sin(5\theta)]$ with $\theta \in [0, 2\pi]$. The scattering of the points is determined after the points are placed on a regular grid inside the irregular domain. The points closest to the boundary are projected onto the boundary and tagged to impose the Dirichlet boundary condition. This results in irregular point distribution along the boundary, as well as inside the domain. Figure 5.9 shows a solution example using the sum version of the Laplacian. The relative errors of the solution for the described diffusion problem is shown in figure 5.10 for three levels of scattering and smoothing radii, $c = 30\%$ with $h = 2.0\Delta$, $c = 60\%$ with $h = 2.7\Delta$, and $c = 90\%$ with $h = 3.5\Delta$. The SPH method diverges in this case, and therefore, the results are omitted from the graphs. The FD solution with the irregular boundary was obtained by setting $c = 0\%$ and by using the smallest possible compact radius. While the

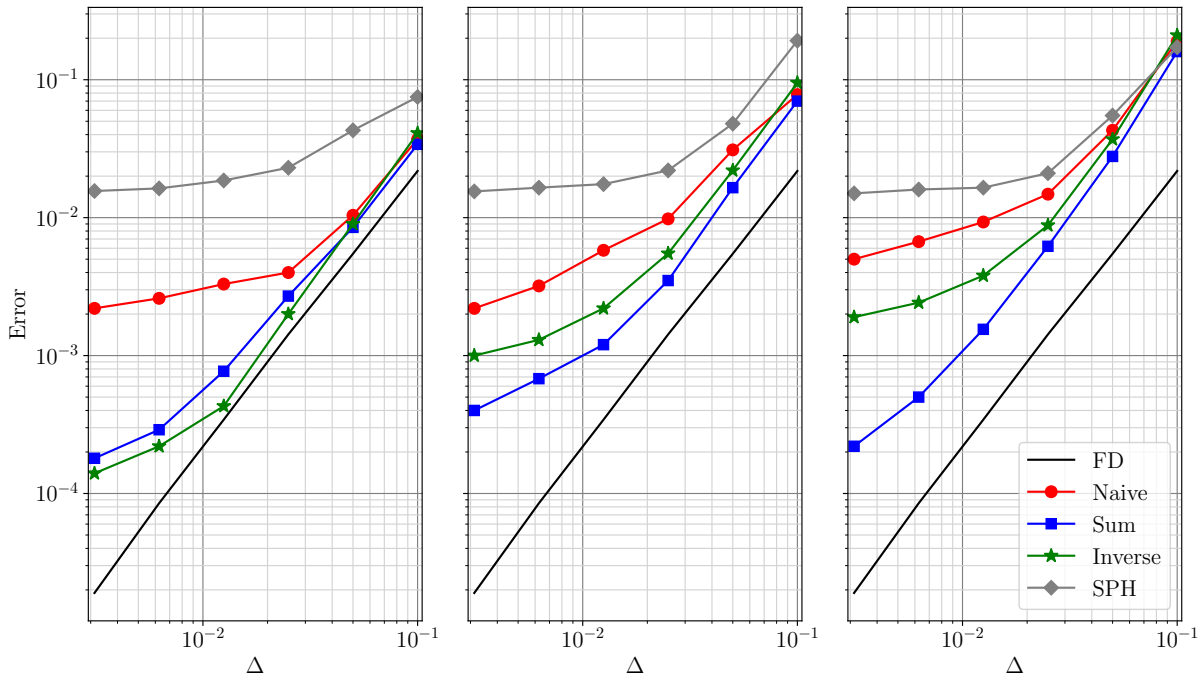

Figure 5.8.: Relative errors of the Poisson equation solutions for three scattered point arrangements, $c = 30\%$, and $h = 2.0\Delta$ (left graph), $c = 60\%$, and $h = 2.7\Delta$ (centre graph), and $c = 90\%$, and $h = 3.5\Delta$ (right graph).

sum version of the Laplacian performs adequately, the inversion version of the Laplacian performs worse than expected.

5.1.2.3. Irregular Domain with Robin BC

In the following test, the diffusion equation defined as $\frac{\mathrm{d}u}{\mathrm{d}t} = \nabla^2 u + g$ is solved with the Robin BC $\nabla u \cdot \boldsymbol{n} + \alpha u = f$ imposed on the irregular interface, where g is the source term, and $\boldsymbol{n}$ is the outward normal pointing to the outer region of the domain. An example taken from [191] is considered, with the irregular domain in two spatial dimensions given by the zero isocontour $\phi = 0.4\cos(8\theta) + \sqrt{x^2 + y^2} - \pi$ with $\theta \in [0,\, 2\pi]$. Firstly, the points are placed on a regular grid inside the square domain $[-1.5\pi,\, 1.5\pi]^2$. After the scattering of the points is determined, the points closest to the zero isocontour are projected onto it and tagged to impose the Robin BC. The BC is imposed with an α coefficient set to 1.0, and the diffusion equation is solved from time $t = 0$ to $t = 1$ with the implicit–Euler time discretisation. The discrete gradient operator for the Robin BC is described by equation (3.10). The analytical solution is given as $u(x,\, y) = -e^{-2t}\cos x \cos y$. Figure 5.11 shows a solution example using the sum version of the Laplacian, and the relative errors of the solution for the described diffusion problem are shown in figure 5.12 for three levels of scattering, $c = 30\%$, 60% and 90%. In contrast to the former test cases, where the compact radius varied according to the level of the scattering, here the compact radius

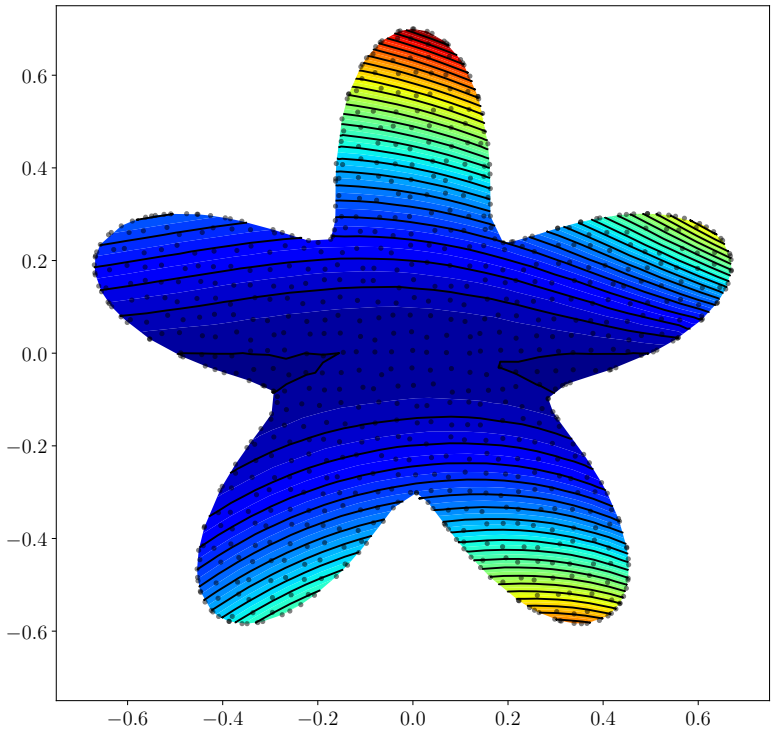

Figure 5.9.: Solution to the Poisson equation on an irregular domain for the Laplacian sum version, $\Delta = 0.0375$, $c = 60\%$, and $h = 2.7\Delta$. Thick curves represent qualitative contour levels of the exact solution.

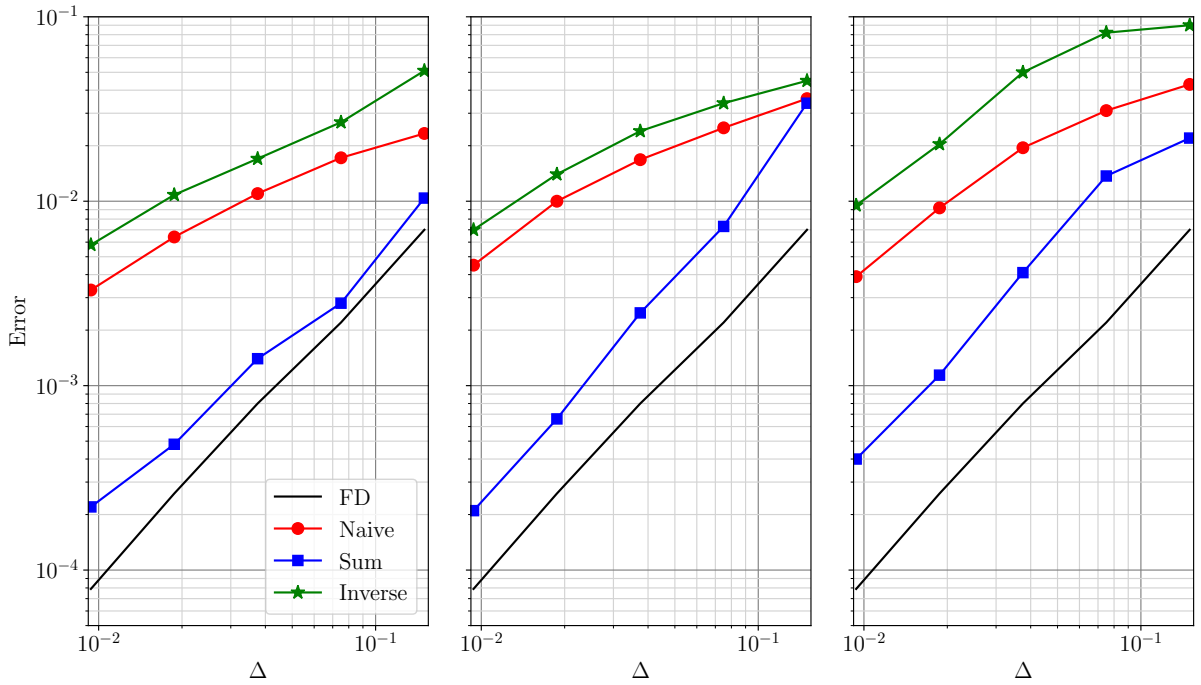

Figure 5.10.: Relative errors of the Poisson equation solutions on an irregular domain for three scattered point arrangements, $c = 30\%$, and $h = 2.0\Delta$ (left graph), $c = 60\%$, and $h = 2.7\Delta$ (centre graph), and $c = 90\%$, and $h = 3.5\Delta$ (right graph).

was kept constant, $h = 2.7\Delta$. For the described problem, the solutions made with the SPH and naive version of the Laplacian show a linear rate of convergence. However, the solutions made with the sum and inversion versions of the operator in any case follow the FD convergence trend, although with somewhat shifted values. The three graphs in figure 5.10 show how the level of scattering affects the convergence slopes.

Even though the Neumann and Robin BCs can be properly imposed with the introduced spatial operators, various difficulties when solving problems with shocks and jump conditions are encountered. Apart from the mixing of the domains inside a compact sphere near an interface, the other difficulties are similar to those of classical central–difference methods. FD methods simply discretise the relevant equations on interpolation points, often without conserving quantities like mass, momentum, and energy. They are proven useful mostly for phenomena where there are no strong shocks. In contrast, an elliptic equation with discontinuities in the solution, its gradient, the diffusion coefficient and the flux across an irregular interface, is efficiently solved with a FV method [192]. The contributions from the discontinuities can be included on the right-hand-side of the linear system, preserving its positive symmetric definiteness. In a meshless context, Hopkins [119] solves the sharp shock–capturing problem by employing spatial operators introduced in section §3.2.1 and a Riemann solver in a moving frame to describe the physics of the interaction between the particles, which results in a FV meshless method.

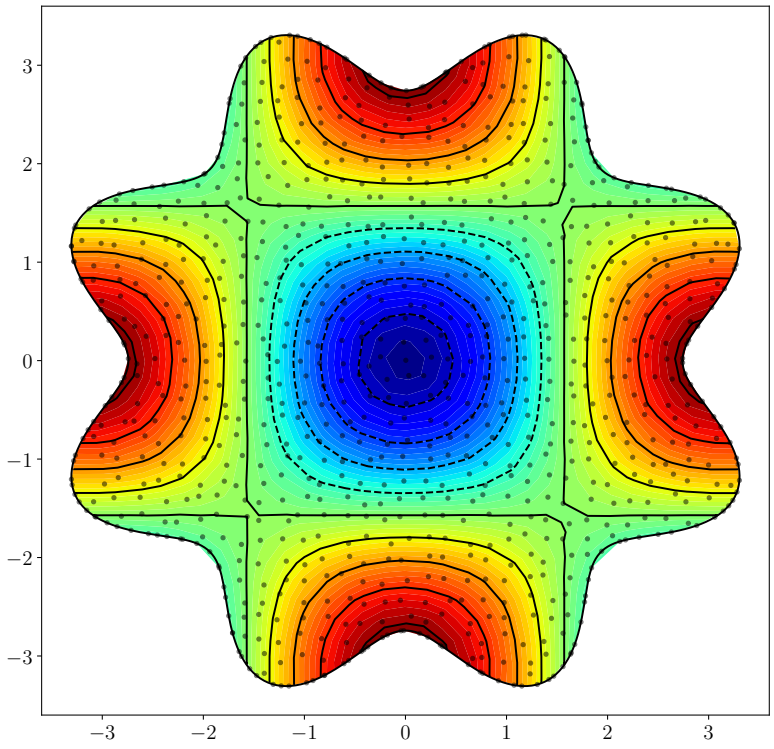

Figure 5.11.: The contour plot of a diffusion equation solution for the Laplacian sum version, $\Delta = 0.2$, $c = 60\%$, and $h = 2.7\Delta$. The thick curves represent qualitative contour levels of the exact solution.

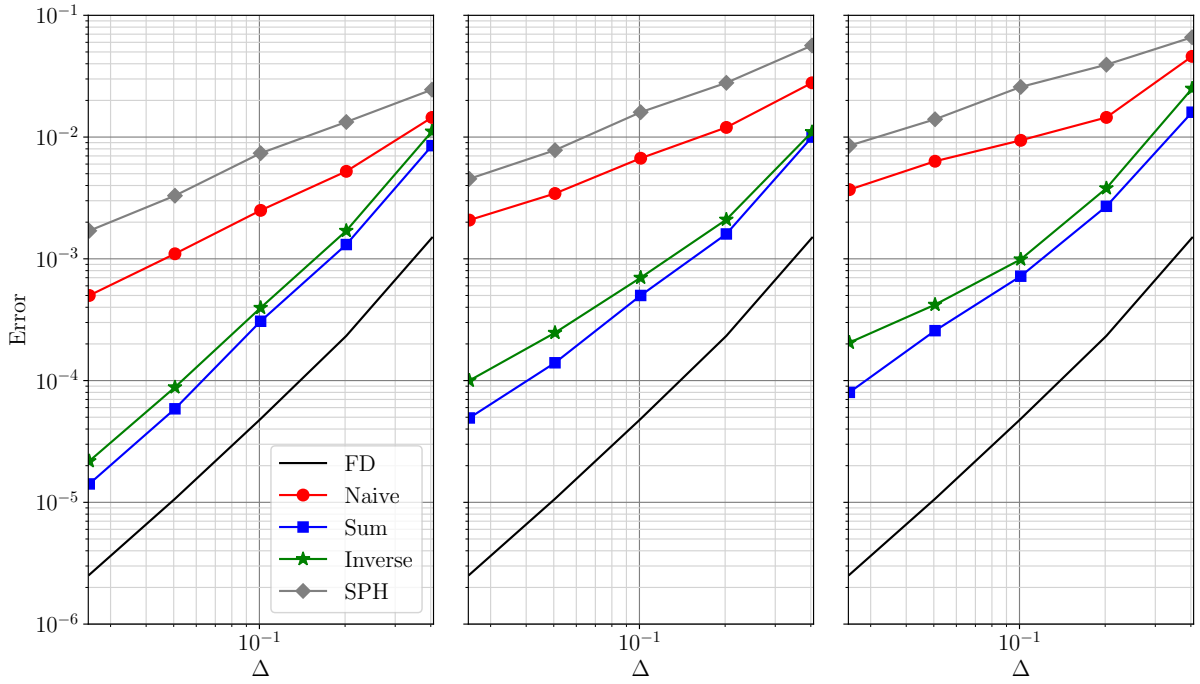

Figure 5.12.: Relative errors of the diffusion equation solutions for three scattered point arrangements, $c = 30\%$ (left graph), $c = 60\%$ (centre graph), and $c = 90\%$ (right graph), for the constant compact radius $h = 2.7\Delta$.

5.1.2.4. Spherical Domain with Dirichlet BC

A three-dimensional diffusion problem $\nabla \cdot (\beta \nabla u) = f$ taken from [190] is considered and is defined by a variable diffusion coefficient given as $\beta(x, y, z) = xyz$, and the exact solution of the problem is given as $u(x, y, z) = \sin(4\pi x)\sin(4\pi y)\sin(4\pi z)$. The Dirichlet BC is imposed on the spherical boundary defined by the zero isocontour $\phi = \sqrt{\left(x - \frac{1}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2} - 0.3$. Firstly, the points are placed on a regular grid inside the cubical domain $[0, 1]^3$. After the scattering of the points is determined, the points closest to the zero isocontour are projected onto it and tagged to impose the Dirichlet BC. The compact radius was kept constant, $h = 2.7\Delta$, to test the robustness of the introduced operators for high levels of scattering in three-dimensional space. The relative errors of the solution for the described problem are shown in figure 5.13 for three levels of scattering, $c = 30\%$, 60% and 90%.

5.1.2.5. Computational Efficiency

Compactly supported meshless methods form large sparse linear systems to mathematically model various phenomena, and ordinarily their implementations rely on efficient iterative solvers such as the bi–conjugate gradient (BiCG) method, the generalised minimal residual (GMRES) method, etc. In this thesis, the quasi–minimal residual (QMR) method was employed for the analysis. Unlike the GMRES method, the QMR method

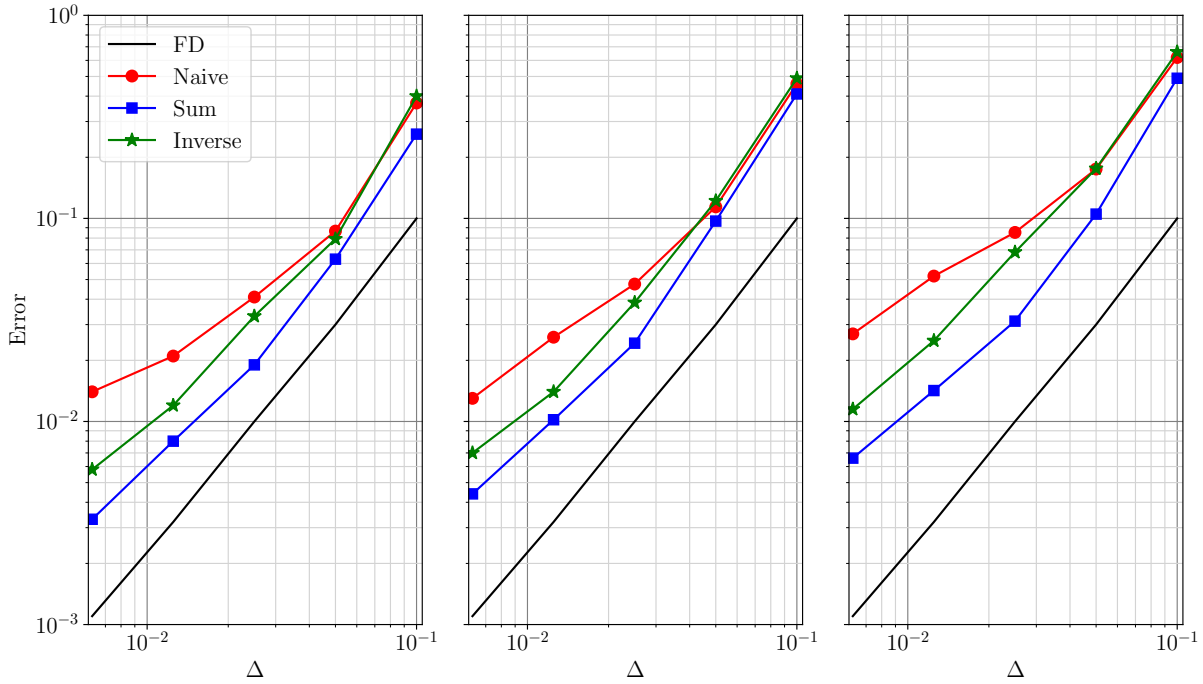

Figure 5.13.: Relative errors of the 3D diffusion equation solutions for three scattered point arrangements, $c = 30\%$ (left graph), $c = 60\%$ (centre graph), and $c = 90\%$ (right graph), for the constant compact radius $h = 2.7\Delta$.

contains one convergence loop that makes computational efficiency evaluation easy to interpret. In addition, the convergence behaviour of the QMR method is typically much smoother than that of the BiCG method.

The evaluation of the invested solver's effort (i.e. its number of iterations) needed to reach a certain relative error of the solution is shown for one representative problem in figure 5.14, i.e. for the scattered point arrangement defined with the following parameters: $\Delta = 0.025$, $c = 60\%$ and $h = 2.7\Delta$. The FD method noticeably reaches the lowest relative error value, but slowly converges due to the fact that the QMR solver is unsuitable for the FD matrix. Figure 5.8 confirms that the sum version of the Laplacian offers the best performance among the tested meshless operators and converges very fast with the employed solver. The SPH Laplacian shows very weak convergence progress, and the naive version follows a similar trend, but with a noticeably stronger convergence slope. Interestingly, the inversion version of the Laplacian starts with the strongest convergence slope, which weakens and thus never reaches a lower error value than the sum version error, even though its approximation errors are lower, as presented in figure 5.5. More insight on this discrepancy is needed, which may be partially attributed to the non-optimal choice of the smoothing function. To conclude, fast convergence rates of the QMR solver with the sum version of the Laplacian and its approximation efficiency presented in table 5.1 prove that the sum version offers a good compromise between computational efficiency and accuracy.

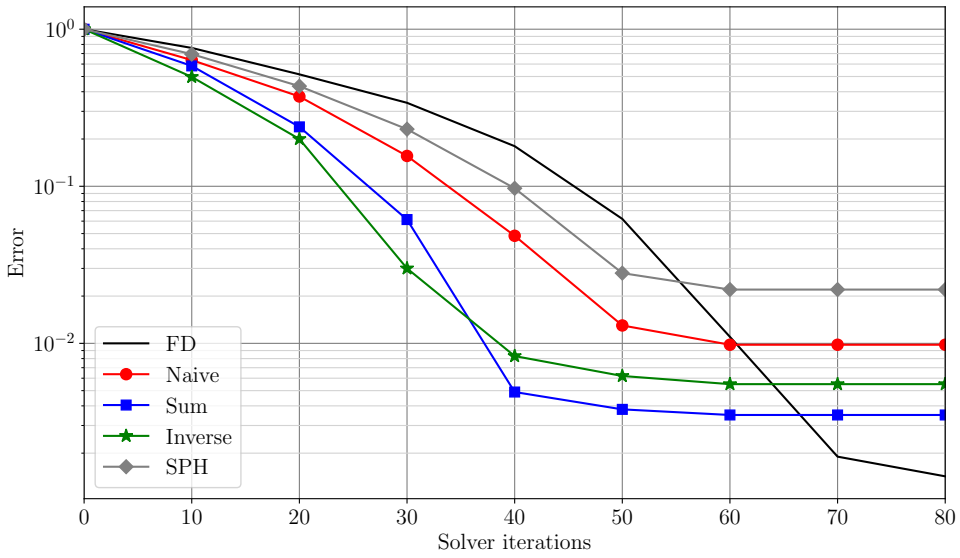

Figure 5.14.: Linear–system solver convergence applied to the Poisson equation in strong form, for the scattered point arrangement: $\Delta = 0.025$, $c = 60\%$ and $h = 2.7\Delta$.

5.2. Cavity Flow

The lid–driven cavity problem involves viscous flow inside a cavity in which the top wall moves horizontally, while other walls remain stationary. The classical cavity problem features a square cavity and a constant velocity of the lid. Because of the simple two-dimensional geometry and boundary conditions, it is widely considered as a benchmark case for new CFD solvers. The problem does not have an analytical solution, unless it is modified to include a specific source term in the momentum equation [193]. The schematic of the flow problem is shown in figure 5.15.

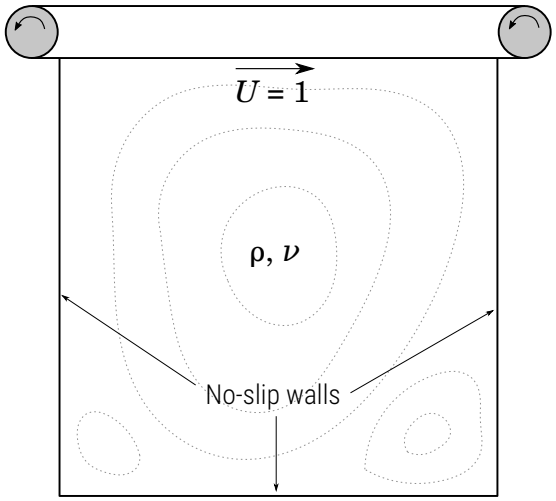

Figure 5.15.: Schematic of the lid–driven cavity flow set-up.

There is no free surface, so the PPE is subjected only to Neumann boundary conditions.

The Neumann problem does not have a unique solution, i.e. it has a solution that is unique up to a constant. There are various approaches how to deal with the problem, but in this thesis, the matrix diagonal dominance is enhanced as explained in section 3.6.2 and QMRCGStab iterative solver is used. BiCGStab and GMRES solvers can also handle the system of equations, but too many solver iterations results in the divergence.

Internal flow problems, such as the flow inside a cavity, are challenging for meshless Lagrangian methods. Slight numerical compression and expansion of the fluid point with free surface is hard to detect, while the solid boundaries force a constant volume throughout the simulation. Therefore, a Lagrangian method must oblige to the continuity constraint through the velocity field and physical volume conservation at all time in order to keep the simulation stable.

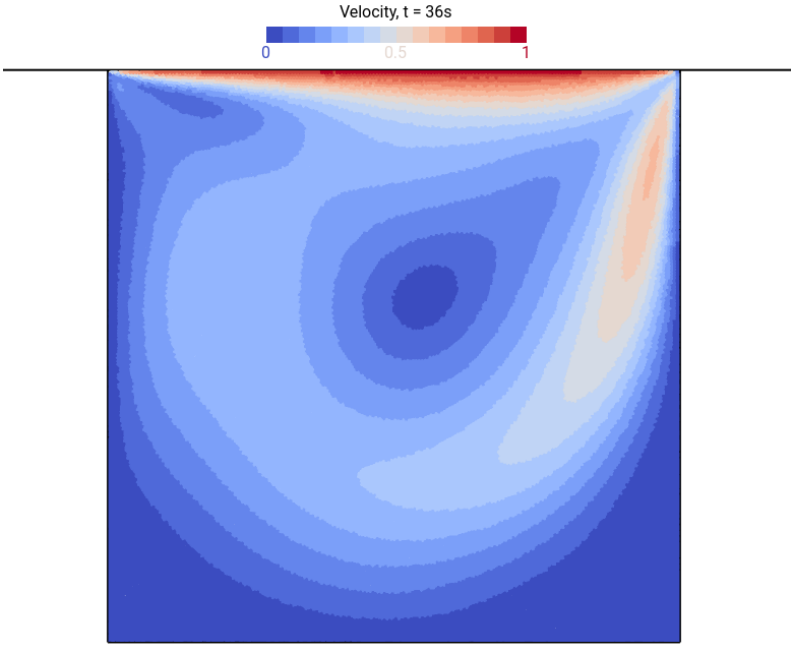

Figure 5.16.: Contour plot of the velocity magnitude inside the cavity, for the 200×200 grid and $Re = 400$.

The imposed lid velocity is constant, $U = 1\,\mathrm{s}^{-1}$, and the cavity is a unit square. The Reynolds number considers the lid velocity, cavity size, and fluid viscosity. Therefore, the Reynolds number is defined by $Re = \nu^{-1}$, and the kinematic viscosity of $\nu = 0.0025$ corresponds to the considered test case $Re = 400$. The flow was thoroughly studied by Ghia *et al.* [194] on two different spatial resolutions that yielded negligible difference between the corresponding results. The results obtained by the meshless solvers are compared to those obtained by Marchi *et al.* [193]. The authors revisited the square cavity problem using the FVM on a 1024×1024 grid and verified their results by comparing them to 18 other studies, in order to provide a reliable set of data for comparisons. The problem is simulated by the implemented solver on three discretisation refinement levels. The

initial discretisations, i.e. point clouds, that filled the cavity were initialised as uniform grids made of: 50×50, 100×100, and 200×200 points. Figure 5.16 shows the magnitude of the velocity field after the simulation initialised by 200×200 points has converged to a steady state.

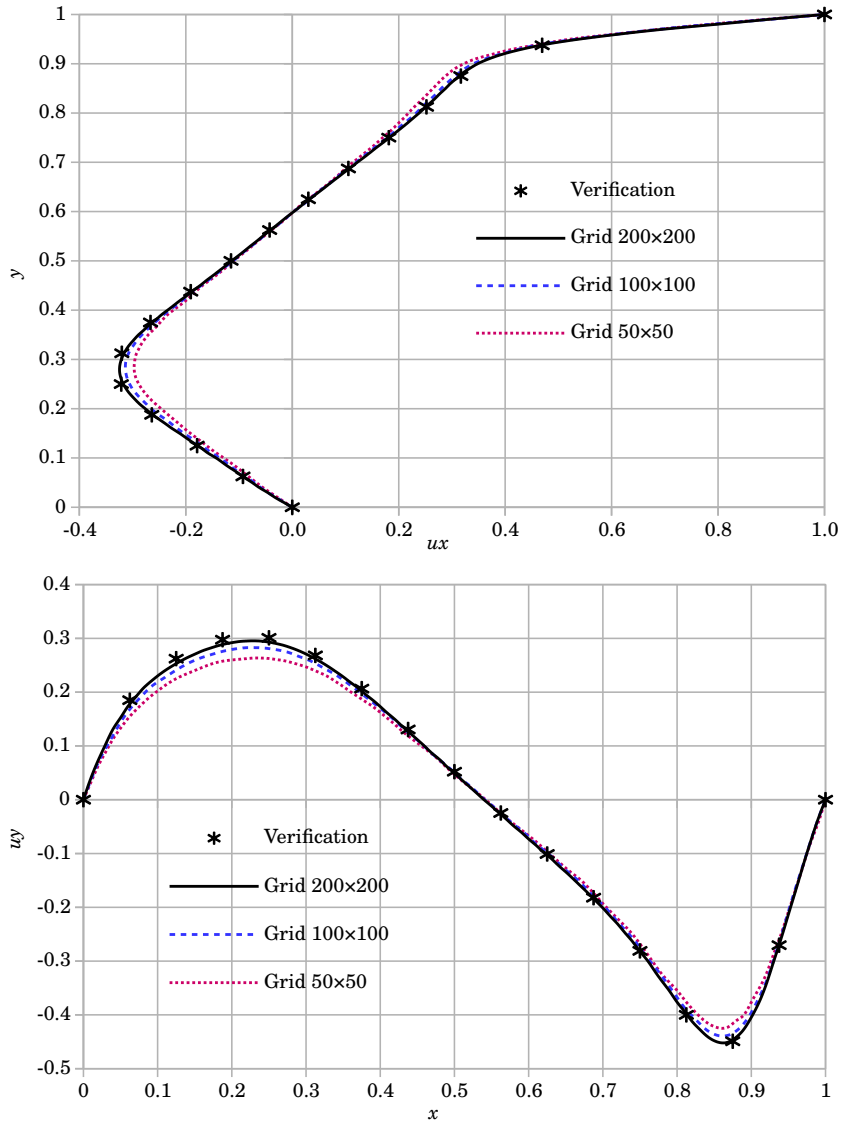

Figure 5.17.: Velocity profiles normal to the vertical (top graph) and horizontal (bottom graph) centrelines, for $Re = 400$ and different discretisation densities, compared with the results of Marchi *et al.* [193].

The main features of the flow for benchmarking are the horizontal velocity profile along the vertical centreline, and the vertical velocity profile along the horizontal centreline of the cavity. The velocity profiles in the two directions at cavity centrelines are shown in figure 5.17. The results obtained on the three discretisations are in good agreement to the referent data. Furthermore, the results converge to the true solution by refining the discretisation, that is the initial point cloud. The velocity–profile curves extracted from the finest discretisation excellently fit to the referent data, which is gratifying since the

referent results were obtained on much finer grid.

Marchi *et al.* [193] note that they do not aim to present an optimized numerical model neither for CPU time nor for computational memory consumption. Their single core implementation converged to the solution in 5 days and 16 hours. The GPU solution for the 200×200 grid was obtained in circa 10 minutes for which the simulation reached 40 seconds of physical time, which is yet to be optimised.

Since the results are in very good agreement, it may be concluded that the introduced Laplacian adequately drives the viscous flow through adequate discretisation of the viscous term and the PPE. Moreover, the Lagrangian advection of the fluid remains stable and conservative at all time.

5.3. Water Entry Experiments

In this section a set of experiments is numerically reproduced to verify that the solver can handle complex FSI and impacts. The problem of water entry was firstly motivated in the design of seaplane structures by von Kármán and Wagner. Water entry or slamming problems usually include a moving body that impacts or slams the undisturbed free surface and enters water. During the water entry, the body may completely submerge, depending on the body momentum, shape, etc. This section verifies the introduced solver for such impacts under various circumstances. The experiments include symmetrical and non-symmetrical water entries of simple and complex shapes in deep and shallow water.

5.3.1. Symmetrical Wedge

Various experiments on an accelerated wedge with a flare angle of 30° were conducted at MARIN, in order to study bow–flare slamming and the wave run-up around the bow of an FPSO. The wedge was 202 mm wide and 175 mm high. During the experiments, a pressure transducer was located 30 mm above the wedge tip. The two selected experiments were numerically reproduced in [195] using a IBM–VOF solver. The authors used time traces of the wedge movement from the experiment as the input for the numerical solver, which are graphed in figure 5.18. In similar fashion, the two experiments are reproduced using the introduced method and the numerical evolution of pressure signal is compared to the experimental data and numerical results from [195]. The authors note that the measured movement contained high-frequency noise, which was filtered out to enable numerical differentiation. The simulations were performed on two discretisations defined by initial point spacing $\Delta = 5$ mm and $\Delta = 10$ mm.

In the first experiment, the wedge started to fall from the height of 200 mm above the initial waterline, which resulted in relatively high entrance velocity of 1.43 m/s. The

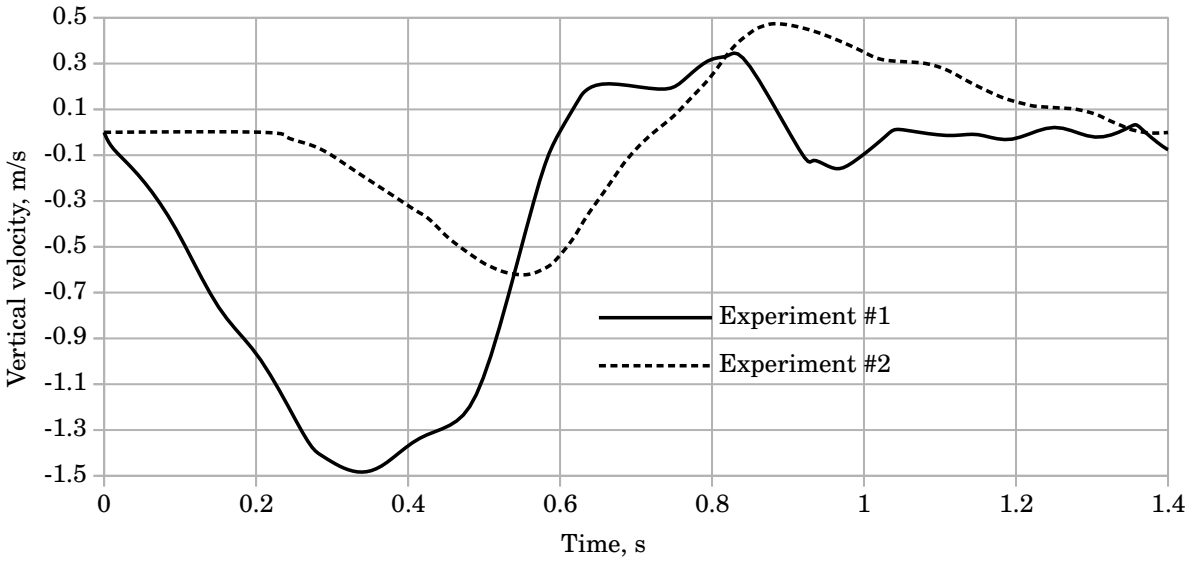

Figure 5.18.: The vertical component of the wedge velocity for the two slamming experiments [195].

value of the time step was set to 0.5 ms. The evolution of the vertical component of the wedge velocity is plotted in figure 5.18. The pressure evolution obtained by the simulation is plotted in figure 5.19, and compared to the experimental data and numerical solution obtained by the IBM–VOF solver. During the water entry, the wedge had entirely submerged and not lifted out of the water. The simulated pressure signals are in good agreement with the experimental signal, and overall more accurate than the IBM–VOF results. All numerical simulations yield somewhat higher pressure magnitude at the start of the water entry. The two simulated discretisations give approximately the same results. The discrepancy at the start of the entry obtained by the novel method and the IBM–VOF solver may also be attributed to the imposed movement, since the oscillating experimental signal was filtered.

In the second experiment, the wedge started to fall while its tip was touching the initial waterline. This resulted in a much smaller value of the entrance velocity compared to the first experiment. The value of the time step was set to 1 ms. The pressure evolution obtained by the simulation is plotted in figure 5.20. The reaction of water on the body entrance is less impulsive than the first experiment. Therefore, the total pressure is comparable to the hydrostatic pressure. Dynamic effects were still reproduced, represented by the two peaks and a hollow in the signal. It should also be noted that the simulated pressure signals in figures 5.19 and 5.20 are not filtered, while the IBM–VOF signal is filtered, because the solver yielded pressure spikes that were pronounced directly after the high-velocity water entry [195].

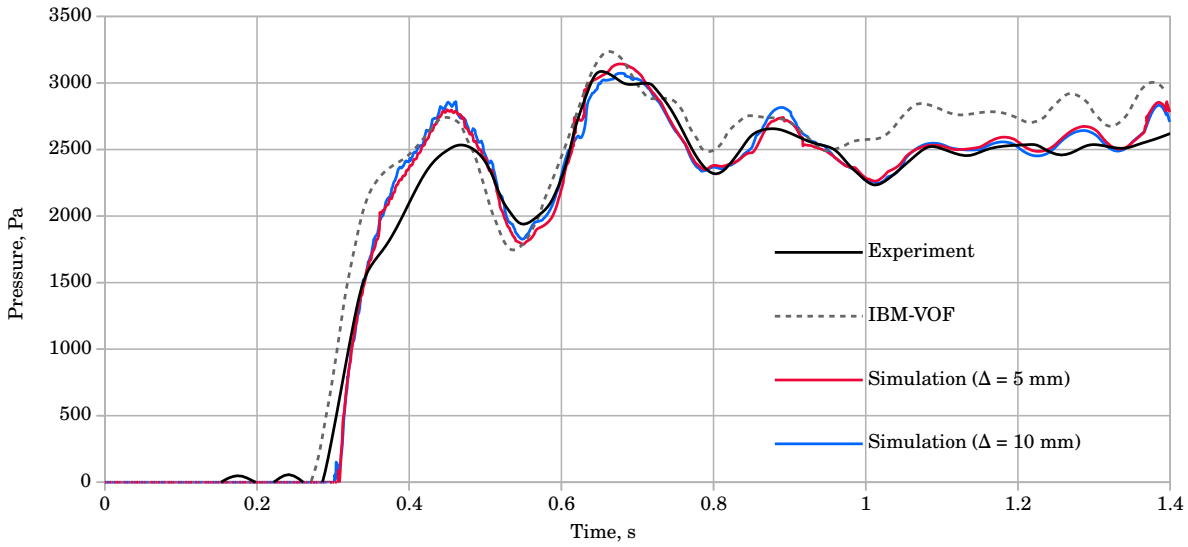

Figure 5.19.: Pressure sensor readings for the first numerical experiment of symmetrical water entry compared to the experimental measurements [195].

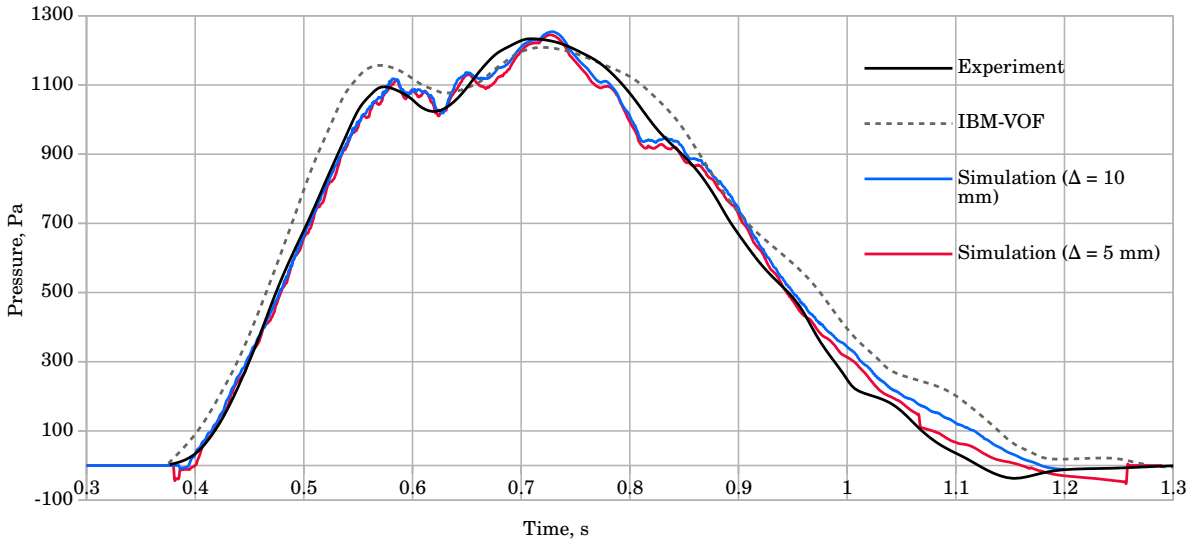

Figure 5.20.: Pressure sensor readings for the second numerical experiment of symmetrical water entry compared to the experimental measurements [195].

5.3.2. Tilted Wedge

Non-symmetric water entries are studied to understand various problems in marine engineering. The hydrodynamic behaviour of the impact is different from those predicted by the symmetric entry. Generally, the non-symmetric problem is less investigated compared to the symmetric entry. During the campaign "Wave–induced loads on ships (WILS III)", drop experiments of a wedge with 30° of deadrise angle and mass of 68.3 kg were conducted. The wedge was titled by 20° about the tip, and then dropped from a height of 500 mm above the initial waterline. A pressure sensor was located on the titled side that formed an angle of 10° with the waterline, distanced 50 mm from the wedge tip along

the flare. The tilting of the wedge produced a pressure signal with a steeper rising and declining slope than non-tilted experiments. In the comparative study [196], 20 solvers were tested by simulating the problem.

Compared to other numerical experiments described in this thesis, this experiment was simulated by using weakly coupled FSI. In other words, the wedge was falling under the influence of gravity, and its movement during the water entry was also dictated by forces in fluid acting on the wedge. The force was obtained by integrating the pressure along the wetted surface in a simplistic manner, i.e. by summing the product of the pressure value and approximate surface area of all boundary points generated on the wedge geometry, $\boldsymbol{F} \approx \sum_i \boldsymbol{n}_i \, p_i \, \Delta^2$. Except for validating the solver for hydrodynamic loads, this numerical experiment proves that the methodology may be extended for coupling with a more complex rigid–body solver or a FEM structural solver.

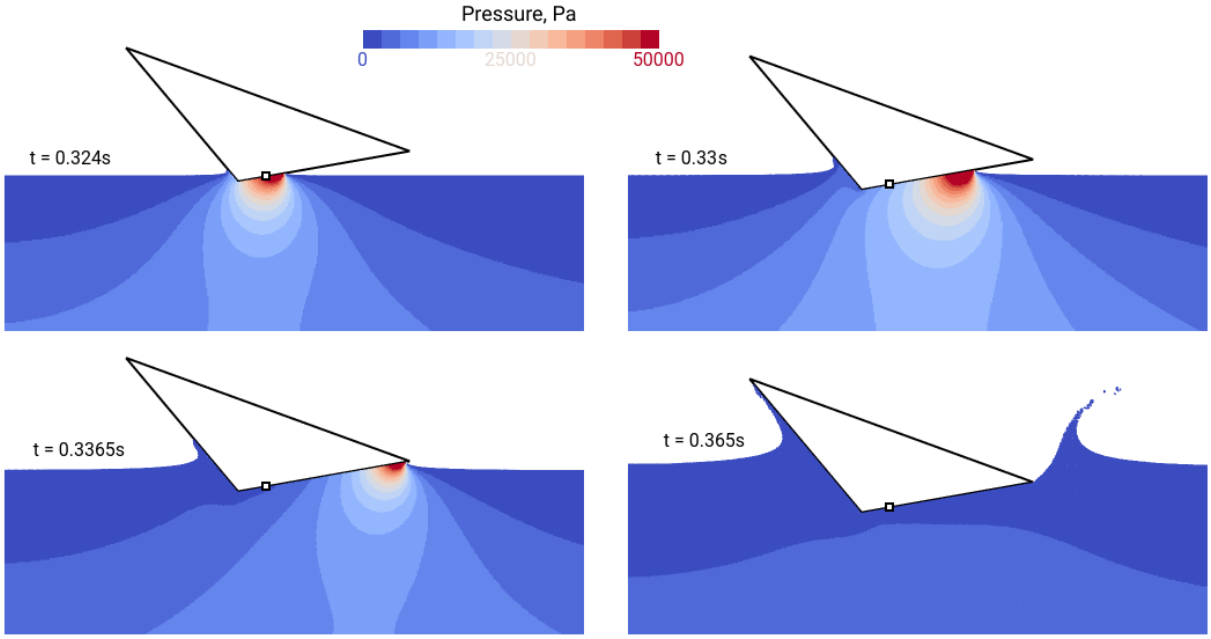

Figure 5.21.: Snapshots of four time instants of the non-symmetrical water entry simulation, rendering contours of the pressure field.

A series of snapshots of the simulation is shown in figure 5.21. The visualised pressure field shows how during the entry a concentration of high-pressure area moves from the wedge tip along the flare. The high gradient of the pressure diminishes when it reaches the wedge corner, which generates a strong jet that separates from the wedge.

The evolution of the pressure signal recorded by the pressure sensor is plotted in figure 5.22. The experimental signal was shifted in time to overlap with the simulation. The left image compares the experimental signal with the results of simulations performed on two spatial discretisations, and the right image compares the experimental signal with the results of simulations performed on the finer discretisation and two time step values.

Due to an extremely short rise and decay time of the pressure at impact, and the simple implementation of force integration for the weak coupling, the value of the time step was chosen to be rather small. The two tested time–step values were 0.1 ms and 0.25 ms, which correspond to classical CFL numbers of 0.3 and 0.7, respectively. First thing is to observe that the simulations yield stable similar non-oscillating pressure signals for two discretisations and time–step values. The larger time step produced a smoothed-out solution in between the steps where the smaller time step detects some changes in the pressure. Nevertheless, the simulated pressure signals correspond well to the experimentally obtained pressure signal and referent numerical results given in [196].

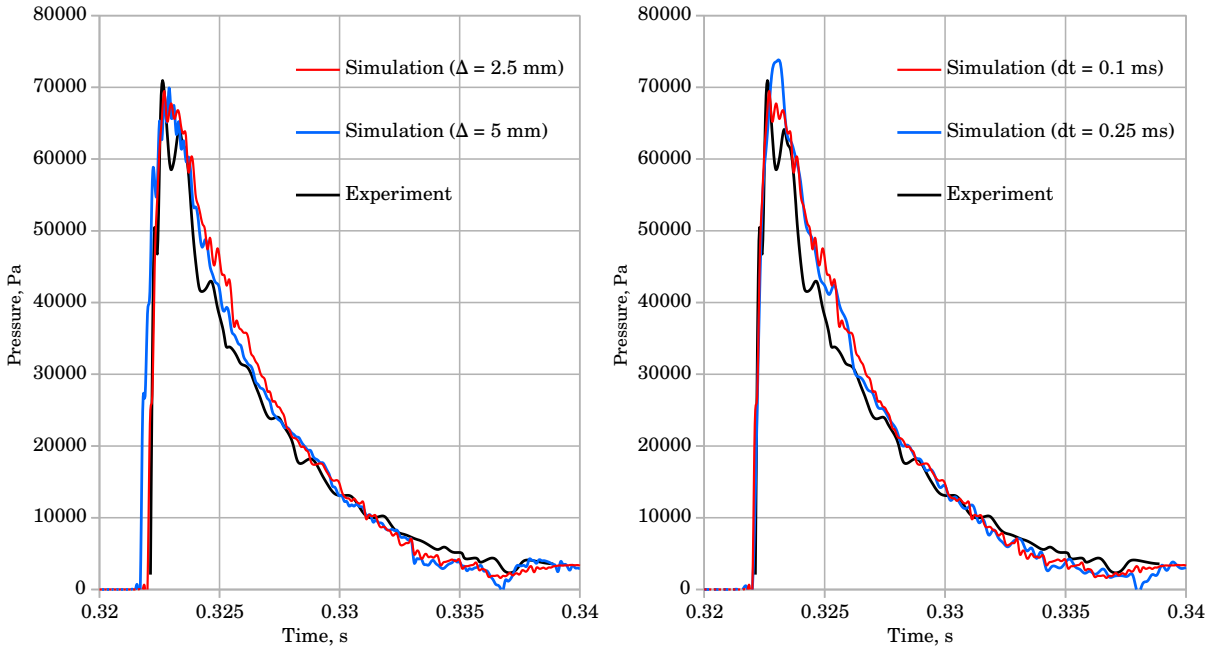

Figure 5.22.: Pressure sensor readings for the non-symmetrical water entry simulations with different time steps and point spacings, compared to the experimental measurements [196].

5.3.3. Wedge and Shallow Water

A high-speed impact on the free surface results in impulsive hydrodynamic loads, which is controlled by the entry speed and the vessel geometry. For shallow water impact, the presence of the bottom may significantly amplify the hydrodynamic loads, by constraining the water motion beneath the object. Jalalisendi *et al.* [197] experimentally investigated the problem of shallow–water entry of a wedge by using PIV, which provides means to determine slamming loads and insight into the flow physics that can be further used to validate numerical models. By tracking particles one may obtain the velocity field, and the pressure field can be reconstructed by solving the PPE. Therefore, an experiment described in [197] is simulated in order to validate the novel methodology.

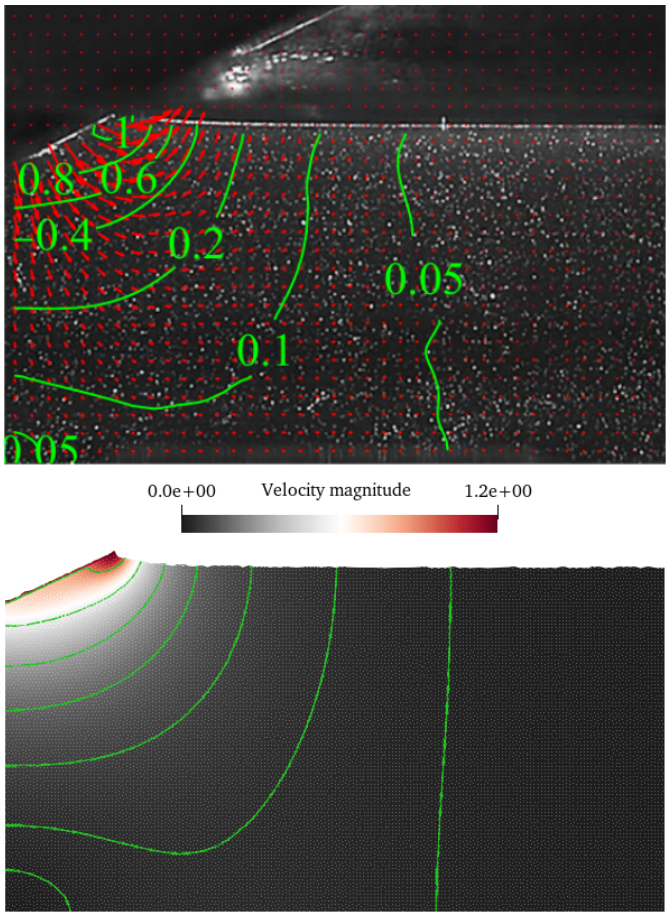

Figure 5.23.: Contour plot of the velocity magnitude in m/s around the wedge for the entry depth of 5 mm. Comparison of the numerical solution (bottom image) and the PIV experiment [197] (top image).

An impact of a two-dimensional wedge against a shallow water column of finite height is considered. The water tank was 800 mm wide and the water was filled up to the height of 50 mm. The wedge had a deadrise of 25° and width of 200 mm. At the time instant when the wedge tip touched the free surface, the wedge was moving with speed of 1.5 m/s, and the following 10 ms the speed of the wedge dropped linearly to 0.98 m/s. This movement is set as input for the numerical solver. The simulation was set-up with $\Delta =$ 0.5 mm, which is also in the order of magnitude of the PIV equipment that tracked the experiment. The time step was set to a constant value of 0.2 ms.

Figure 5.23 compares the simulated velocity field to the PIV results. The velocity magnitude and contours are plotted, while the experimental values are taken from [197], which are normalized with respect to the instantaneous wedge speed. For deep water, the velocity of water near the bottom is zero. For shallow water, the presence of the bottom wall forces the fluid to advect along the transverse direction. This is visible in figure 5.23, where the contours indicate movement near the bottom wall. The contour shapes correspond to experimental ones in both low and high-velocity areas in fluid. The area of fluid where the high-velocity gradient and maximum velocity occurs is located at the intersec-

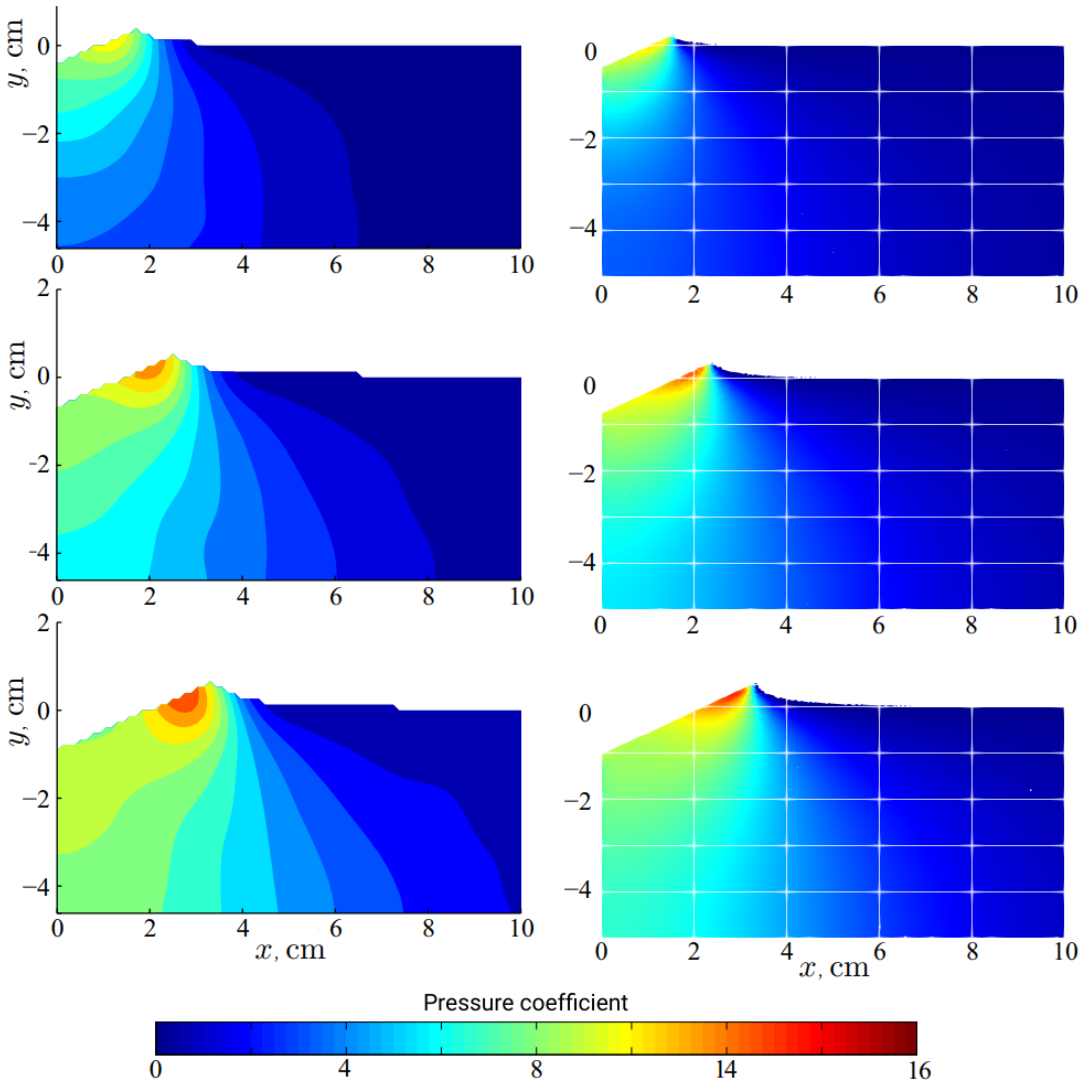

Figure 5.24.: Contour plot of the pressure coefficient around a wedge entering the shallow water for three entry depths (top to bottom): 5 mm, 7.5 mm, and 10 mm. The numerical solution (right column) is compared to the PIV–reconstructed results [197] (left column).

tion of the still free surface and the wedge, which is also well reproduced. In addition, the numerical simulation reproduced a small area at the bottom, below the wedge tip, where the fluid particles do not move. As the wedge enters fluid, maximum fluid speed becomes much larger that the speed of the wedge, e.g. the maximum velocity magnitude in fluid is 2.2 m/s at the entry depth of 10 mm.

The pressure field was reconstructed from the experimental velocity field by solving the PPE. Since the gravity was neglected, the simulations were also performed in zero-gravity conditions, thus comparing the hydrodynamic pressures. The comparison of reconstructed and simulated contours of the pressure coefficient is shown in figure 5.24. The pressure coefficient was obtained by normalising the pressure value by $\rho U\left(t\right)^2/2$, where $U\left(t\right)$ is the instantaneous speed of the wedge. The rows in the figure correspond to three time instants

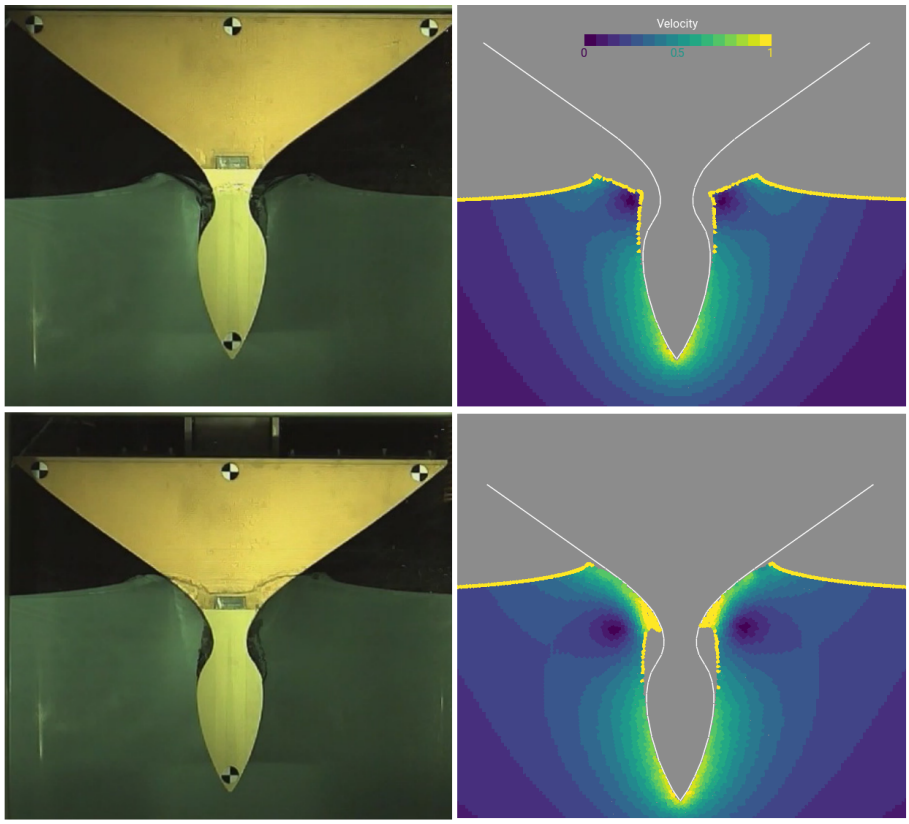

Figure 5.25.: Free fall of the ship–bow section. Comparison of the numerical solution (right column) to the experiment photographs [198] (left column) for two time instants: 240 ms (top row) and 260 ms (bottom row).

captured for entry depths of 5 mm, 7.5 mm, and 10 mm. The numerical solution and the experiment are in close agreement for each entry depth. Numerically obtained shapes of contours and the coefficient magnitude coincide with those obtained by PIV and pressure reconstruction, while there is some discrepancy in the thickness of the high-pressure area.

5.3.4. Ship Bow Section

During the campaign "Wave–induced loads on ships (WILS III)", drop tests of two-dimensional ship sections were conducted. The drop test #11 of a containership bow section (SS11) is chosen for the validation purposes, as it represents the most difficult test case to reproduce numerically due to the concave geometry and air entrapment [198]. The experimental model III is characterized by the bulbous shape, which is dropped from 300 mm, i.e. at the start of the drop, the bottom touches the free surface. The readings from three pressure sensors located along the bow flare are used to validate numerical solutions. Information on the section geometry and sensor locations can be found in Monroy *et al.* [198].

Pressure sensor #1, which is located on the flare just above the bulbous shape, experiences

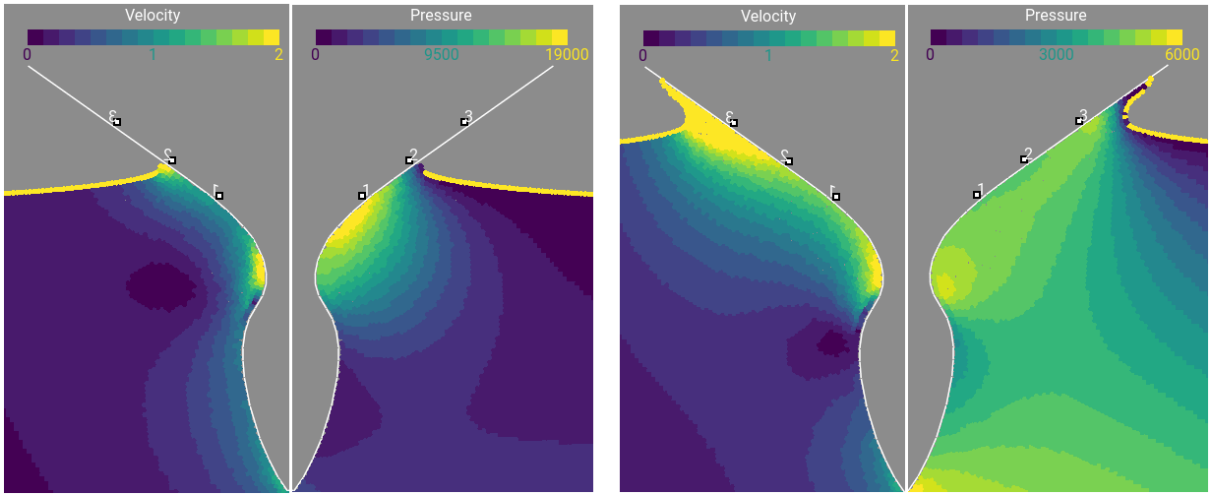

Figure 5.26.: The pressure and magnitude of the velocity field after the flare impacts the free surface, at two time instants: 270 ms (left image) and 300 ms (right image). Locations of the sensors are plotted as squares and designated as 1, 2, and 3.

much larger pressure peaks than the other two sensors. The reason for this is that the formed air cavity, shown in figure 5.25, implodes with high velocity. Figure 5.26 renders snapshots of two instants after the implosion, with the pressure and velocity–magnitude field plotted. The pressure peaks at the low–velocity area from where the implosion begins, which is located just below sensor #1. PBD and Lagrangian advection allow for slight compressibility in cases of immediate implosions or tensile forces, keeping the scheme stable. The evolution of pressure in time is plotted in figures 5.27–5.29 for the three pressure sensors, respectively. For all three sensors, the simulated pressure spike and peak values are in good agreement with the experimental measurements. Both simulated and experimental curves exhibit a rise in pressure due to added mass after 0.4 s.

Figures 5.28 and 5.29 also include results presented by Monroy *et al.* [198]. The authors note that *BV–Slam,* which is an in-house potential flow solver based on a Generalized Wagner Model (GWM), fails to accurately predict the force and pressure applied to the concave part of the section, but overall results are satisfactory. The authors also present a FVM–based solution using *OpenFOAM*, which was shown to be mesh dependent, and the overpredicted pressure peak in figure 5.28 may be attributed to neglecting important compressibility effects in the solver. In terms of pressure peak values and slope, the proposed method shows better agreement with the experiment than the FVM solver and potential–flow solver. However, the impact and the pressure spike is somewhat shifted in time. The shift occurs due to slower closure of the cavity from excessive relaxation of PPE near the boundaries and slightly weaker advection of points near free surface where points have low neighbour count.

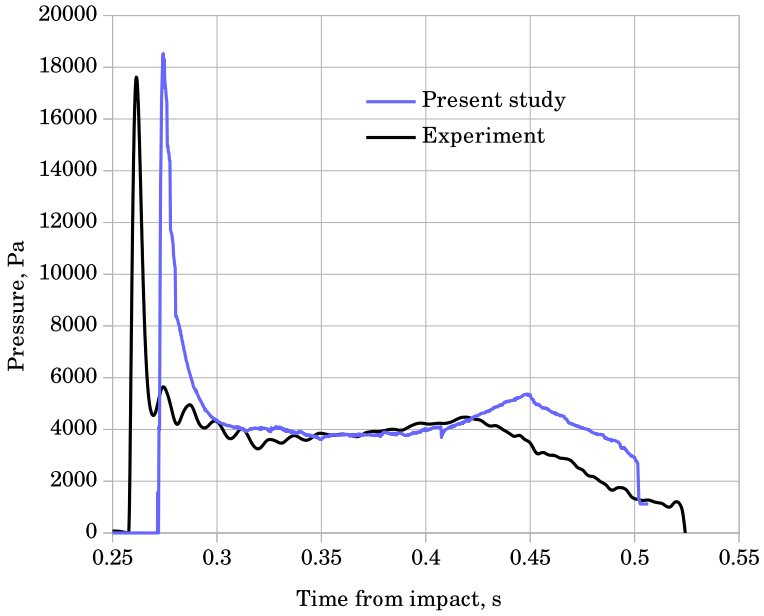

Figure 5.27.: Evolution of the pressure read by sensor #1. Comparison of the proposed method and the experimental results [198].

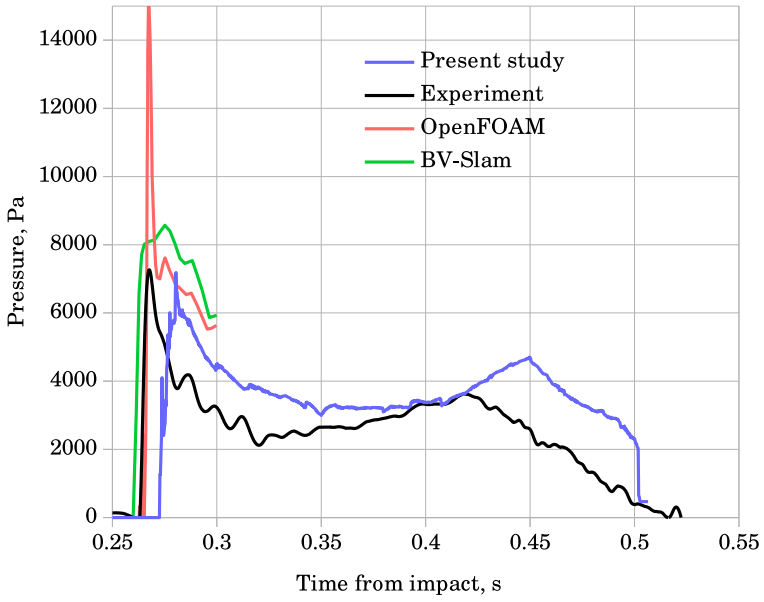

Figure 5.28.: Evolution of the pressure read by sensor #2. Comparison of the proposed method and the experimental results [198].

5.4. Dam Break

Compared to water-entry problems in which water is still while objects move, dam break problems consider fixed geometry while water moves. A dam break experiment is numerically simple to set up, since no inflow or outflow boundary conditions need to be imposed. Moreover, there is a resemblance between a usual case of green water flow on

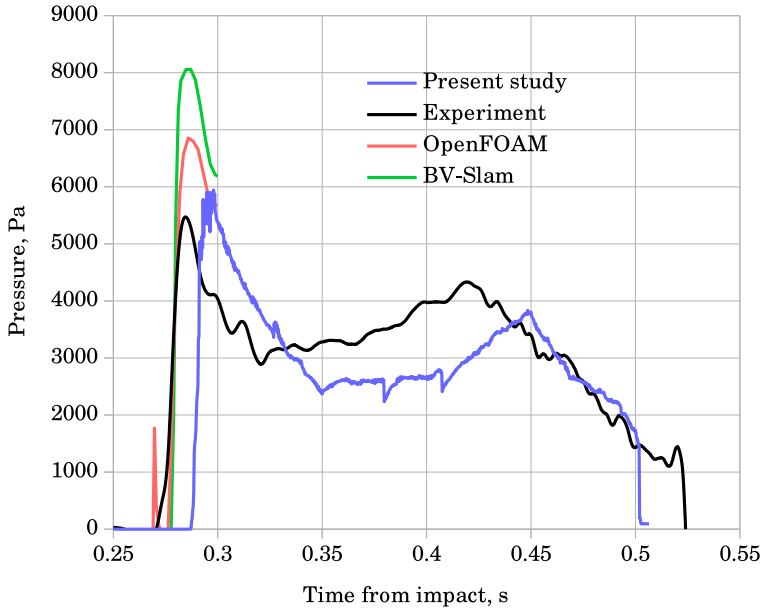

Figure 5.29.: Evolution of the pressure read by sensor #3. Comparison of the proposed method and the experimental results [198].

the deck and the theoretical dam breaking problem [20, 48]. Water on the deck forms a high velocity water jet and violently impacts a structure like an impinging jet.

Some dam break studies that were done numerically have reported various issues. For instance, Kleefman *et al.* [199] and Hu and Kashiwagi [200] reported highly unstable pressure peaks, while Arai *et al.* [201] observed diverging peak pressure when refining the mesh. The issues mostly arise due to sudden changes in the flow direction that the model formulation cannot properly describe. The authors agree that the stabilisation of the pressure evolution in numerical techniques will contribute to further studies. Mokrani and Abadie [202] employed a RANS–VOF method for dam break simulations over the dry bed. Their analysis of the results shows the large spatial and temporal variability of the pressure field after impact, highlighting the need for extreme care when performing measurements. Arai *et al.* [201] note that for finer grids more accurate impact pressures are obtained, while in the case of coarse grids, lower impact pressure are computed due to the local averaging effect.

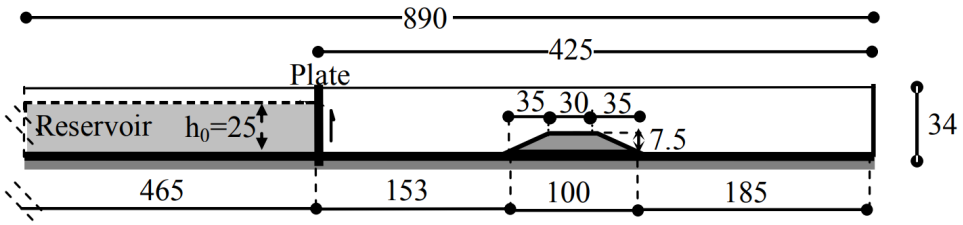

Figure 5.30.: Test arrangement for the dam–break experiment with a trapezoidal obstacle [203]. All dimensions are in cm.

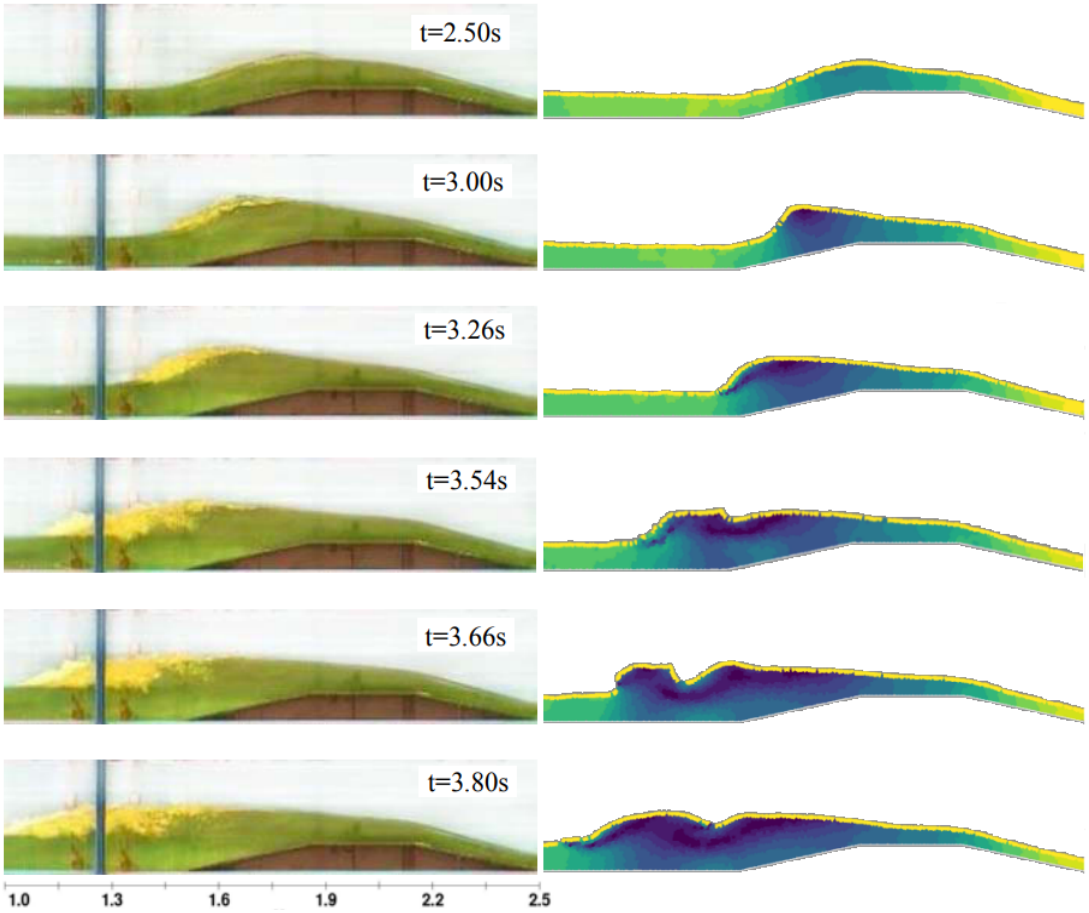

Figure 5.31.: Comparison of free surface profiles for the flow over the trapezoidal step obtained experimentally [203] (left column) and numerically (right column) at various time instants (top to bottom): t = 2.5, 3.0, 3.26, 3.54, 3.66, 3.80 s. The computed velocity magnitude is qualitatively plotted.

Two validation experiments are given in the following text, which are used to investigate the abilities of the introduced solver to reproduce the kinematics and dynamics of the impulsive fluid motion during dam breaking.

5.4.1. Trapezoidal Obstacle

Ozmen-Cagatay and Kocaman [203] presented an experimental investigation of dam–break flow over initially dry bed with a bottom obstacle, for the purpose of validation of RANS and Shallow Water Equations (SWE) codes for problems with bottom slope effects. A water column, which is 0.25 m high and 4.65 m long, is released to flow downstream on a dry bed. As shown in figure 5.30, a trapezoidal obstacle located on the bed is responsible for developing complex flow patterns. This test case highlights not only the bottom slope effects but also those of abrupt change in channel topography, with the formation and propagation of negative bore behind the obstacle. The free surface profiles that were acquired with cameras along the channel are compared to the computed results.

Figure 5.31 renders the evolution of free surface profiles in time and the magnitude of velocity field for a flow over the trapezoidal step. A complex flow pattern develops due to the presence of the obstacle, i.e. the wave is reflected and forms a bore travelling upstream. The negative bore observed on the free surface behind the obstacle is successfully predicted with the numerical solver. As the bore progresses, velocities increase while other parts move up the obstacle. The general characteristics of this mixed flow regime, i.e. the fluid kinematics behind the obstacle, are well captured.

The discrepancy of the free surface contour evolution commences after the reflected wave breaks. The process of wave breaking usually entraps some amount of air, which is revealed as foam in figure 5.31. Furthermore, the high-magnitude vorticity near the free surface entraps more air as the broken wave progresses, which is shown in figure 5.31. Most of numerical methods for incompressible flows suffer from such issues, since the main reason for the discrepancy is vorticity damping near the free surface and neglecting the entrapped air. This can be alleviated by utilising fine discretisation near the free surface, and by allowing slight fluid compressibility with vorticity confinement. These issues are not top priority, since they occur after the impacts and do not significantly affect the general flow characteristics.

5.4.2. Impact Against the Wall

Lobovsky *et al.* [204] conducted classical dam–break experiments in a tank, with the set-up arrangement shown in figure 5.32. The fixed prismatic tank was divided into two separate parts by a removable gate, which was controlled by a release system with a sliding mechanism, a weight inducing the gate motion and a damping system. The tank with inner dimensions of $1610 \times 600 \times 150$ mm was 20 mm thick to avoid hydroelastic effects. The breadth of the tank was chosen so that the resulting dam break flow experiments could be idealized as a two-dimensional phenomenon and wall effects could be considered as not affecting the main flow dynamics [205]. Nevertheless, this experiment is simulated as a three-dimensional problem to test the introduced solver in a complex experiment defined by moving parts and strong dynamics. For the experimental runs when digital images and videos were recorded, the fluid was dyed using a small amount of fluorescein in order to increase fluid's contrast, without exhibiting notable influence on the studied fluid dynamics. The vertical motion of the gate was induced by a falling weight with the median velocity value of 3.46 m/s, which is large enough so that its effect on the liquid column collapse is minor. The fresh water could be considered Newtonian with the density of $997\,\mathrm{kg/m^3}$ and the kinematic viscosity of $8.9 \times 10^{-7}\,\mathrm{m^2/s}$. The authors experimented with two filling heights: 300 mm and 600 mm. The smaller filling height presents more complex experiment to simulate due to thinner wave front and larger average velocity. Moreover, for the 600 mm case the front develops a wave tongue at the initial instances

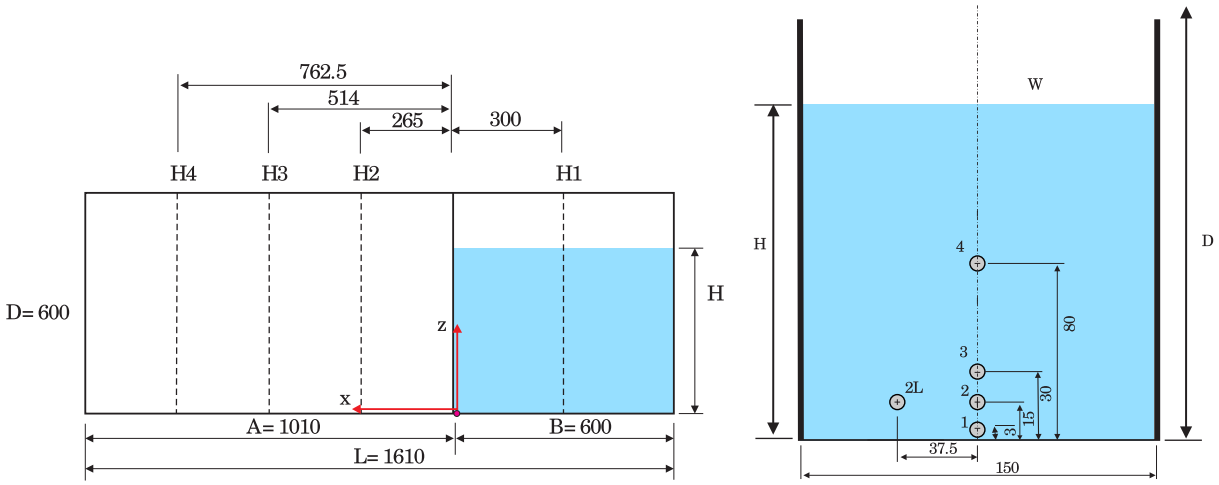

Figure 5.32.: Schematic view of the experimental dam–break setup made by Lobovsky *et al.* [204]. All dimensions are in mm.

of the wave propagation. Therefore, the experiment with the filling height of 300 mm is chosen for the simulation.

Pressure sensors were embedded in the downstream wall on four heights from the bottom. Unlike in the previous dam break studies, the impact pressure is recorded right above the horizontal bed as the pressure probes of a small diameter are used [204]. The pressure signal from the lowest sensor #1, which is located only 3 mm from the bottom, is chosen as the most relevant data for validation. This sensor experiences highest magnitude of the pressure peak and lowest rising time of the pressure during impacts.

The authors presented an analysis of the experimental repeatability of a general dam break problem with emphasis on impact pressures when the wave front hits the downstream wall. They varied different factors, and analysed the changes in the measured peak pressures without managing to attribute them to a particular physical cause. For this reason, median pressure value and the 2.5% and 97.5% levels are plotted along with the numerical solutions, for which the authors note to be relevant for design aspects when considering transient loads in these types of flows. Repeatability of the pressure signal is more certain for the tail of the signal than for the impact event. Slightly varying hydrodynamic characteristics of the wave front before the impact results in different peaks of the pressure. It is expected that the issue of numerical repeatability and uncertainty is comparable to that observed in such experiments.

The experiment is conducted to analyse two-dimensional flow characteristics, so the problem is numerically modelled in three dimensions in order to validate that the solver adequately reproduces two-dimensional flow characteristics of the solution in three dimensions. The initial point spacing was set to $\Delta = 5$ mm, which resulted in a point cloud made of 238000 points that represent the body of water. The value of the time step was constant and set to 1 ms. One time step took 228 ms on average to compute on the GPU.

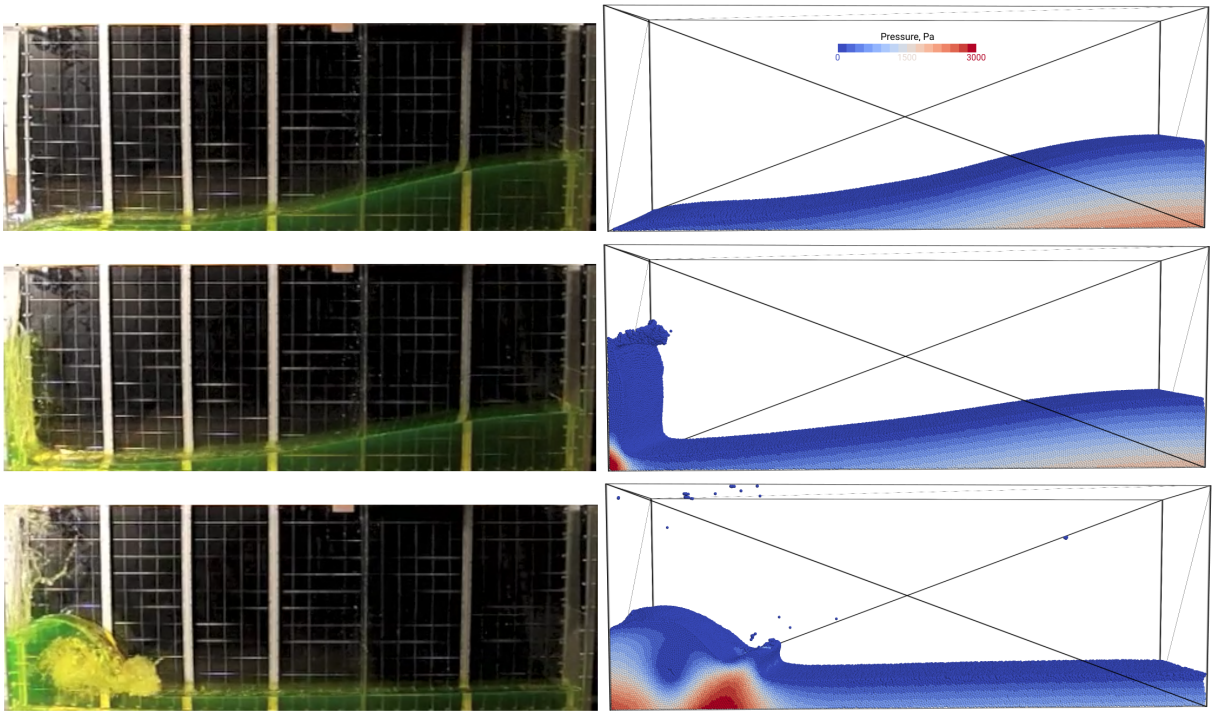

Figure 5.33.: Evolution of the simulated dam break with filling height of 300 mm, and comparison with the photographs of the experiment [204].

Three pressure sensors were placed on the impact wall on the same level, 3 mm above the bottom, which verified two-dimensionality of the solution. Besides the sensor located in the centreplane, two other sensors were shifted 4 mm from the centreplane. Three snapshots captured during the simulation are shown in figure 5.33, which are compared to the images extracted from the video of experiment #91 [204]. The contour plot of the pressure field shown in the figure verifies the smoothness of the solution in the whole fluid domain before, during, and after the impact. Like in the experiment, the front of the downstream wave, travelling along the horizontal bed, does not display significant instabilities prior to the impact on downstream wall.

The numerically obtained pressure signals calculated at the sensor locations are compared to the experimental signal in figure 5.34. The figure shows the results of two different simulations, which differ only in the imposed value for the gate velocity. It is evident that the pressure signal evolves differently for different initial parameters, i.e. slower gate removal induces a bit slower wave front and consequently lower pressure peaks. In either case, the three sensors reproduced almost the same pressure signal, validating that the solver reproduced two-dimensional flow in an three-dimensional environment. It is important to note that the numerical signal remains smooth during the impact of the water front and the wall, which lasts for less than 20 ms. Usually projection–based meshless methods, such as the incompressible SPH and MPS methods, suffer from significant oscillations in pressure field during impacts. The trends of the pressure curves extracted from numerical

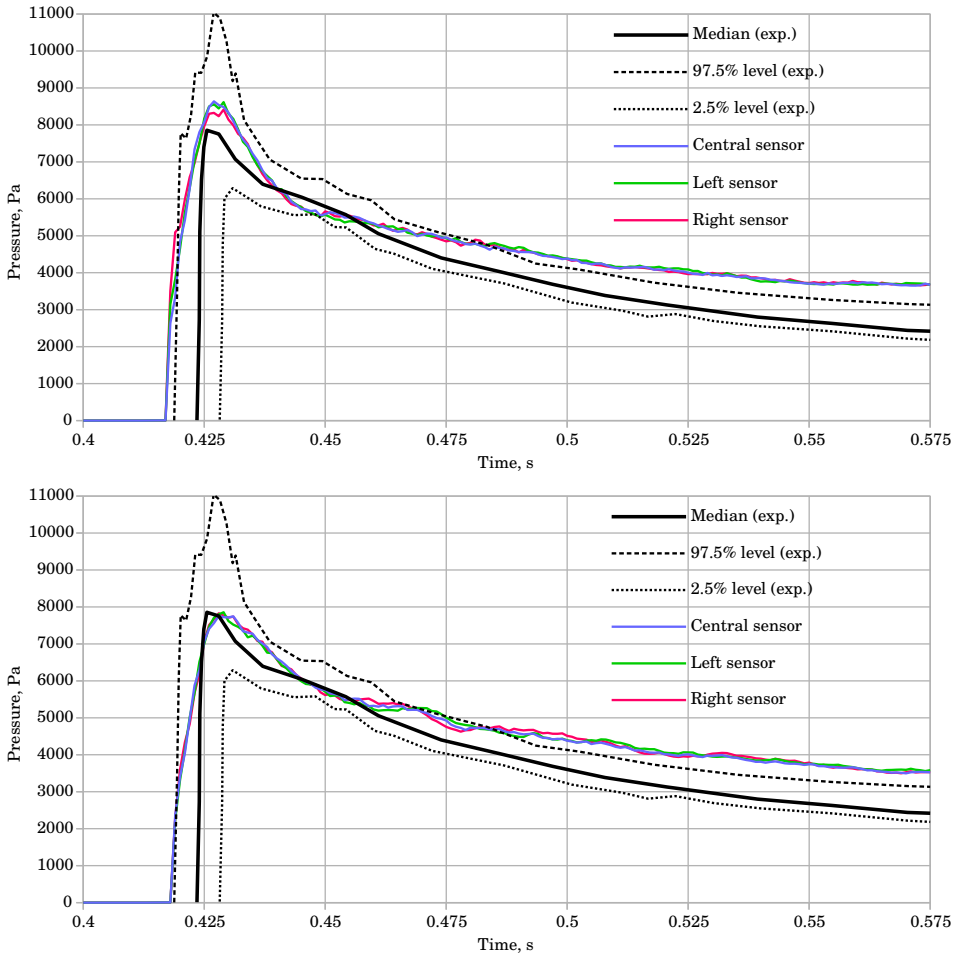

Figure 5.34.: Evolution of the pressure signal during the impact. Comparison of the proposed method and the experimental measurements [204].

simulations follow the experimental ones.

5.5. Sloshing Experiments

The sloshing of liquids is a strongly nonlinear problem, where such pressures arise that may cause structural damage of the tank or may endanger the ship stability. Variously shaped waves that break, jets, spraying and splashing are involved in a moving tank. In the last twenty years, many approximate and complex numerical methods have been introduced in order to predict the loads that arise during sloshing [1]. Peak pressure values during impulsive impacts vary due to the stochastic nature of sloshing, even under a simple harmonic excitations [206]. As such, the phenomenon is very challenging to predict using existing numerical algorithms. Analogous to the water entry experiments, sloshing experiments are simulated in order to investigate abilities of the introduced solver to reproduce kinematics and dynamics of the sloshing in tanks.

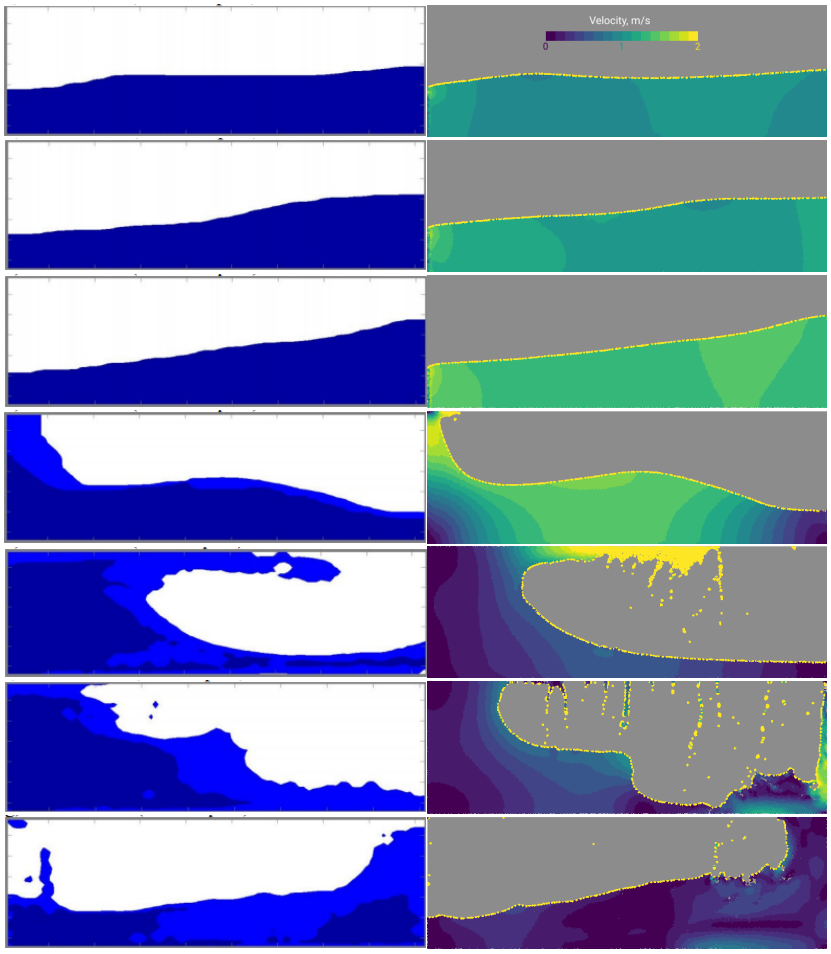

Figure 5.35.: The container filled with water before and after the sudden impact. Comparison of the experimental [207] (left column) and simulated (right column) free–surface profiles at various time instants (top to bottom): $t = 1.34$, 1.68, 1.94, 2.04, 2.17, 2.44, 2.97 s.

5.5.1. Sudden Impact

Khezzar *et al.* [207] conducted experiments using a moving test rig to study the water sloshing phenomena. The container partially filled with water, with dimensions of 550 × 175 × 175 mm, was moved along its longitudinal axis and subjected to a sudden stop. The acceleration defined by $a = 0.68t^{-0.11}$ was imposed on the container until $t = 1.98$ s, when it suddenly stopped. The reached terminal velocity was $U = 1.41$ m/s. The kinematics of water before and after the sloshing in the half-filled tank under sudden impact is investigated numerically and compared to the experimental results.

After 1 s of tank motion, a wave starts travelling in the direction opposite to the movement. The numerical simulation reproduced the travelling wave before the impact with the build-up of the water against the aft wall, which can be seen in the top three images of figure 5.35. Immediately after the impact, the water moves forward and accumulates on the top–front corner of the container, similar to impacts that occur in dam break problems. Then a

high–velocity jet is formed from the corner, which travels to the other end of the container and falls down against the aft wall. Finally, the joined body of water oscillates left–right. Free surface profiles computed numerically and extracted from the experiment show a good agreement, which are rendered in figure 5.35.

5.5.2. Rectangular Tank

Single degree-of-freedom (1–DOF) harmonic motions are simple to model and simulate, but such excitations can lead to unbalanced responses, especially when the frequency is close to the resonance frequency. Kishev *et al.* [208] conducted 1–DOF sloshing experiments in a rectangular tank with dimensions of 600 × 300 × 100 mm. The tank filled with water level, H, was oscillated in sway with the motion described as $\eta\,(t) = A\,\sin\,(2\,\pi\,t/T)$, where T is the oscillation period and A is the oscillation amplitude, which was constrained to $A = 50$ mm.

The two following experiments were simulated to validate the accuracy of the solver with particular regard to the pressure field. Firstly, the test case defined with $H = 250$ mm and $T = 1.0$ s was fitted with a pressure sensor on the left wall of the tank, 235 mm above the bottom. The comparison between the numerically and experimentally obtained pressure evolution at the sensor location is shown in figure 5.36. Peak values, and rise and decline of the simulated pressure for the developed flow in the swaying tank are in good agreement with the experimental measurements. The solver also managed to reproduce negative pressure values where tensile forces occur due to sway oscillation, i.e. changes in the motion direction. Secondly, the test case defined with $H = 120$ mm and $T = 1.5$ s was fitted with a pressure sensor on the left wall of the tank, 100 mm above the bottom. The numerically and experimentally obtained pressure evolution at the sensor location is shown in figure 5.37, which shows a very good agreement between the results.

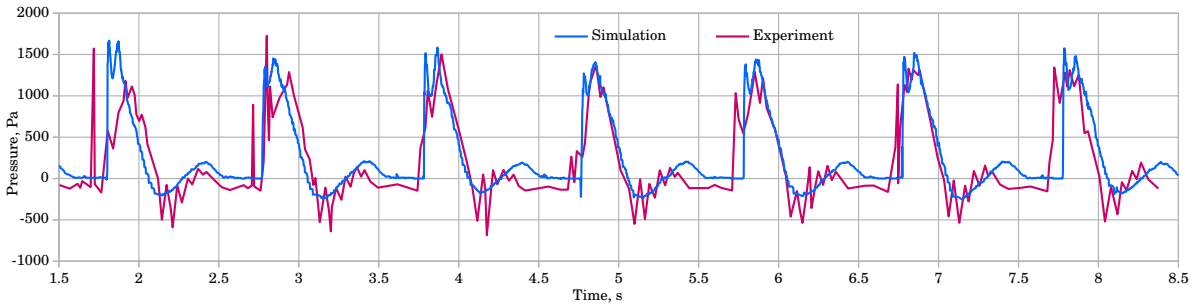

Figure 5.36.: Numerically and experimentally obtained [208] pressure at the sensor location for the swaying rectangular tank defined by $H = 250$ mm and $T = 1.0$ s.

The low–filling case is more challenging to simulate due to occurrence of plunging and collapsing waves forming in the shallow water before the impact. Even if the two-

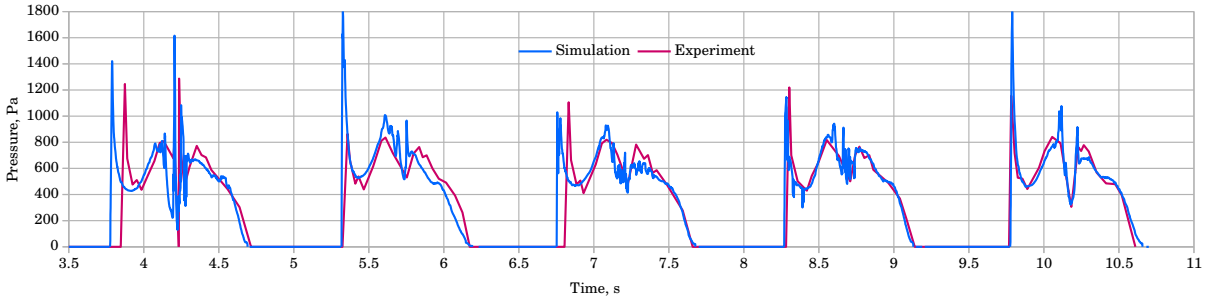

Figure 5.37.: Numerically and experimentally obtained [208] pressure at the sensor location for the swaying rectangular tank defined by $H = 120$ mm and $T = 1.5$ s.

Table 5.2.: Comparison of the average pressure impulse within a rectangular–tank oscillation period, obtained numerically and experimentally [208].

Test case	Experiment, Pa s	Simulation, Pa s	Relative deviation, %
$H = 250$ mm	257.2	267.7	+4.1
$H = 120$ mm	479.0	493.0	+2.9

dimensionality of the flow is ensured, the pressure peak values on impulsive impacts vary due to the stochastic nature of sloshing. Local differences in flow compared to the small sensor area and instrument limitations yield different magnitudes of the peaks at impacts [206]. To overcome the problem of comparing scattered experimental and numerical values, the pressure is integrated in time and the average pressure impulse of a period is used as a comparable measure. The pressure impulse is also a relevant parameter for the structural response, and their comparisons for the two test cases are listed in table 5.2. Very good agreement is shown between the numerically and experimentally values, obtained by integration in time.

To validate the accuracy of the solver with regard to flow kinematics, an experiment defined with $H = 120$ mm and $T = 1.3$ s was simulated, and the resulting free surface profiles are compared to those from the photographs of the experiment. The comparison is shown in figure 5.38, where the pressure field is coloured in order to show the smoothness of the solution. The simulation adequately reproduces the development of the free surface, even the stages of its fragmentation and splashing.

5.5.3. LNG Carrier Tank

Bunnik and Huijsmans [209] conducted sloshing experiments on a 1:10 scale section of an LNG carrier. Compared to previous sloshing model experiments, the experiments are conducted on a significantly larger model scale, which should reduce the effect of stochastic measurements [206]. The case with a rather low tank filling ratio of 10% was simulated,

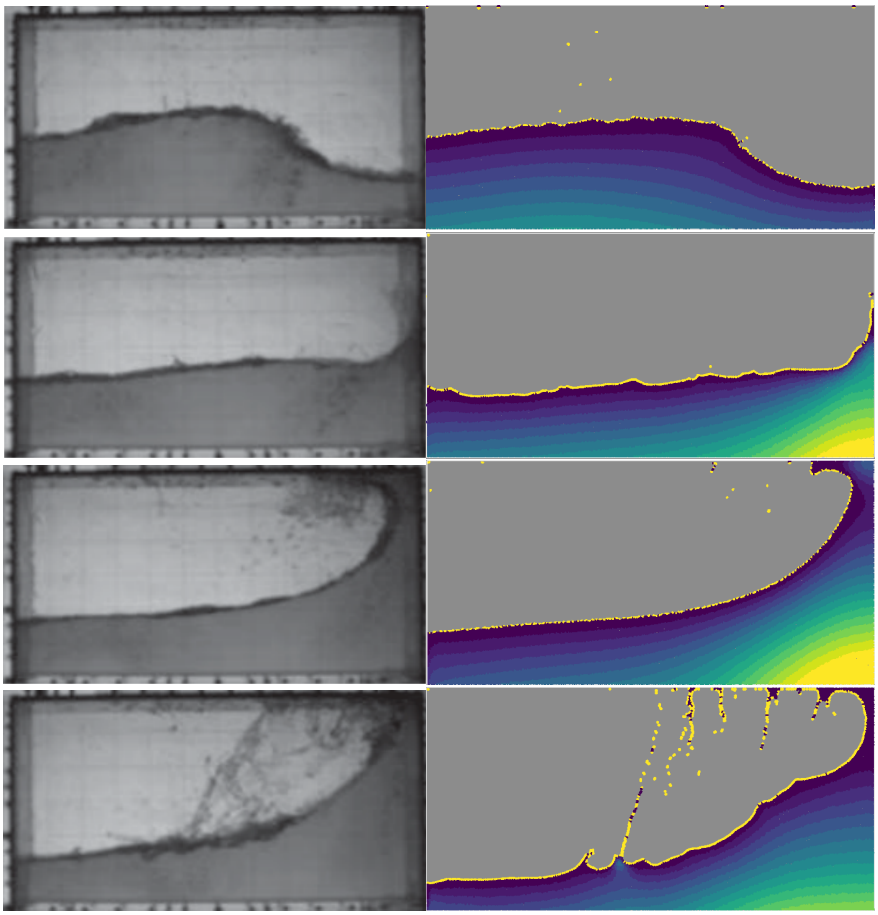

Figure 5.38.: Comparison of the sloshing experiment photographs [208] (left column) and simulated free surface profiles (right column) for $H = 120$ mm and $T = 1.3$ s, at time instants (top to bottom): $t = 1.1T$, $1.2T$, $1.3T$, $1.4T$.

which corresponds to the initial water depth of 438 mm. The 1–DOF sway motion was imposed with a period of 3.2 seconds and the motion amplitude of 100 mm.

Table 5.3.: Comparison of the average pressure impulse within a oscillation period of the LNG tank, obtained numerically and experimentally [209].

Experiment, Pa s	Simulation, Pa s	Relative deviation, %
9668.0	9148.9	–5.4

Although the motion amplitude is small compared to the tank width of 5.4 m, significant sloshing motion is developed as shown in figure 5.39, and hence significant pressure peaks. Figure 5.40 shows the numerically and experimentally obtained pressure evolution for the sensor fitted at the location $\boldsymbol{x} = \{1656, 152\}$ mm from the centre of the tank bottom. The numerically obtained pressure curve is consistent with the experimental measurements after the sloshing flow has fully developed, i.e. after simulating 15 seconds of the swaying motion. The average pressure impulse of a oscillation period is given in table 5.3. The numerically obtained value is close to the experimentally obtained one, although slightly underestimated. After the impact, the jet rises along the side–wall and impulsively falls

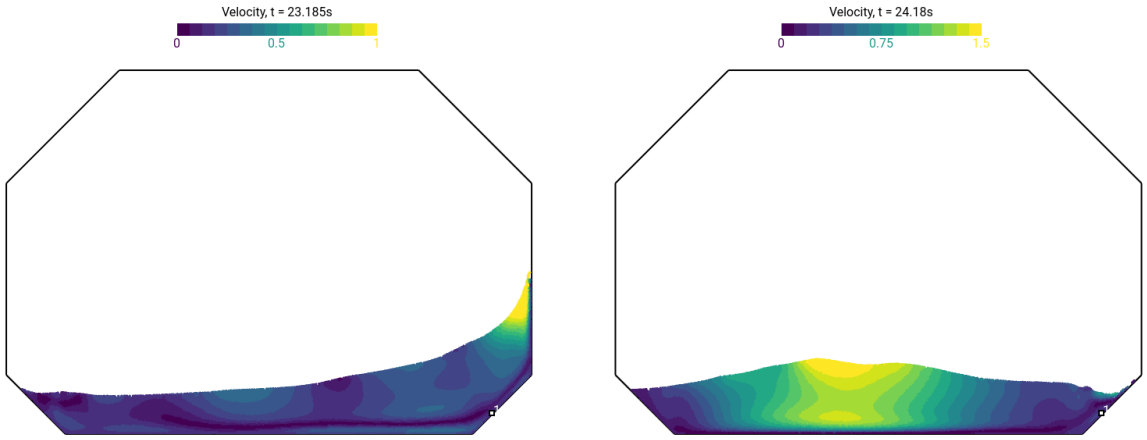

Figure 5.39.: The free surface profiles and velocity magnitudes during the impact (left image) and after the formed jet falls back down (right image).

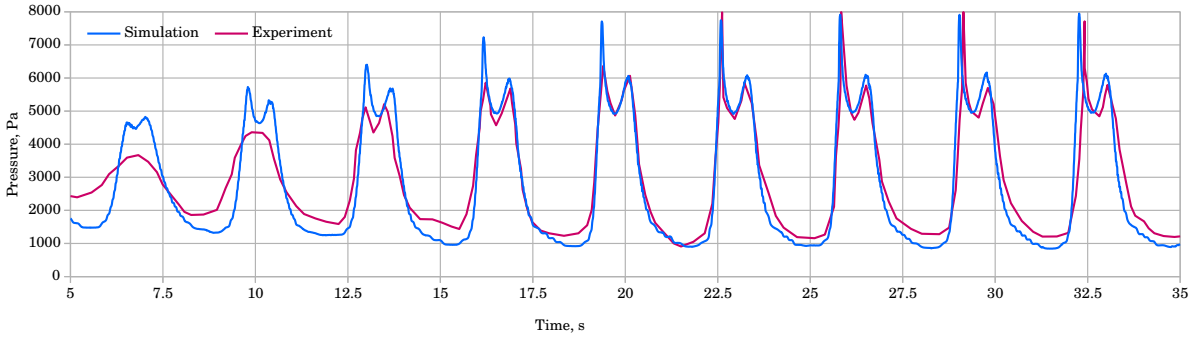

Figure 5.40.: Numerically obtained and experimentally measured [209] pressure at the sensor location for the swaying tank section of an LNG carrier.

down disturbing the flow near the sensor as shown in figure 5.39. The momentum of the falling jet is the culprit of the underestimation, which will be assessed in future work. The magnitudes of peak values are almost constant after the oscillatory flow has fully developed, quite as in the experiment. In other words, the flow over the period remained consistent and two-dimensional both in the experiment and simulation.

5.6. Green Water by Dam Breaking

Green–water flow on the deck has commonly been related to the dam–break flow, as explained in section §1.2, so the standard design analysis approach to estimate the velocity in a green water incident is to use dam–break solutions. Various solutions were proposed for various scenarios of the dam–break flow [48]. When simulating wave trains to study local details, reflected waves must not contaminate results. Therefore, the first numerical experiments conducted to validate the introduced numerical methodology for green water events are simulating dam breaking that results in a wave overtopping a structure. After

the dam breaks, through the action of gravity the column of water interacts with the dry or wet bed and generates a single incoming wave. Different parameters of the experiment generate different waves, i.e. types of green water. Features of the resulting wave depend mainly on the ratio of the initial water depths upstream, h_1, and downstream, h_0, of the gate.

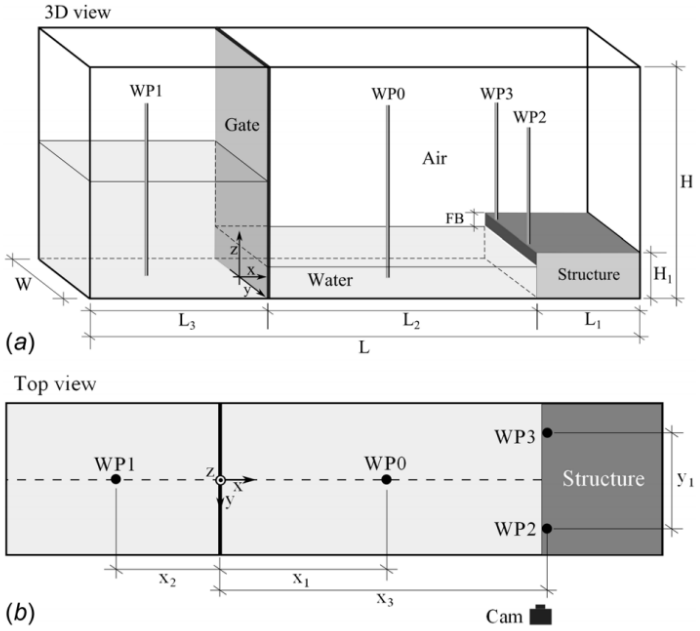

Figure 5.41.: The dam–break experimental set-up: main dimensions dimensions of the tank and the rectangular structure [210].

5.6.1. Wave Patterns

Hernández–Fontes *et al.* [210] experimentally investigated the generation of isolated events of green water on a fixed structure using the wet dam–break approach. They verified that it is possible to reproduce different types of green water resembling those obtained with unbroken regular waves reported in literature. The experimental setup is shown in figure 5.41, where $H = 450$ mm, $W = 335$ mm, $H_1 = 150$ mm, $L_1 = 195$ mm, $L_2 = 505$ mm, and $L_3 = 300$ mm. The testing matrix is presented in table 5.4. The ratios were chosen to generate incoming waves with undular bores [210]. The tests were repeated five times to calculate mean values, which are actually chosen as simulation parameters.

Figure 5.42 shows the free surface patterns captured by the video camera during the experiments, which are compared to those obtained by numerical simulations. The images captured the situation 120 ms after the water level reached the deck level. The simulations have successfully reproduced three different wave patterns from the experiments described in [210]. First four cases form a plunging wave on the deck and an elliptical cavity at

Table 5.4.: Target and measured initial water levels and freeboard heights for the experiment [210].

Case #	Target				Measured			
	h_0, mm	h_1, mm	h_0/h_1	FB, mm	h_0, mm	h_1, mm	h_0/h_1	FB, mm
1	108	180	0.6	42	109.8	179.7	0.611	40.2
2	120	200	0.6	30	122.2	201.0	0.608	27.8
3	126	210	0.6	24	128.6	212.7	0.605	21.4
4	132	220	0.6	18	134.4	221.6	0.606	15.6
5	144	240	0.6	6	145.1	238.2	0.609	4.9
6	108	270	0.4	42	110.4	272.4	0.405	39.6
7	120	300	0.4	30	124.8	305.6	0.408	25.2

the deck leading edge. Cases with lower depth, i.e. higher freeboards, form steeper waves whose peaks are close to the deck leading edge. The case #4 is transitional, forming a mild wave wetting the deck. The case #5 forms the second pattern where the water simply rises above the deck level while the wave peak is still far from reaching the structure. While the image from the three-dimensional experiment shows no cavity, the ideally two-dimensional simulation forms a very small cavity. The third pattern generated by the last two cases, #6 and #7, forms a large cavity and a fluid arm that falls under gravity effects, hits the deck and traps the cavity air. This type of pattern is sometimes referred to as a hammer–fist. The peak of the travelling wave is near the deck edge or above the deck. The authors in [210] note that the cases #1 and #6 have the largest cavities for their water height ratios presented in table 5.4. The magnitude of the velocity shown in figure 5.42 validate the formation of water patterns on the deck. Generally, the velocity magnitude is high near the deck corner, where the high pressure gradient develops. The cavities are formed depending on the magnitude of the pressure gradient near the deck corner. Areas of high velocity magnitude are also generated around the wave peaks that travel towards the structure. The water in case #5 smoothly progresses onto the deck due to the low pressure gradient near the corner and the wave peak, which remained far from the deck. Therefore, the resulting complex motion of the shape of free surface stems from the pressure gradient evolution near the interface and along the structure wall. In conclusion, the resulting free surface of each test case is properly reproduced by the numerical method. Hernández–Fontes *et al.* [210] demonstrated that it is possible to generate isolated green water events of different types on a fixed structure using the wet dam–break approach, and here it is demonstrated that these green water events of different types may also be numerically reproduced. Therefore, future work will deal with simulating yet untested scenarios with various parameters to find new forms of isolated green water events.

The dynamics of dam breaking process is sometimes not modelled in numerical experiments, i.e. the dam or gate simply disappears instead of moving upwards. In cases of

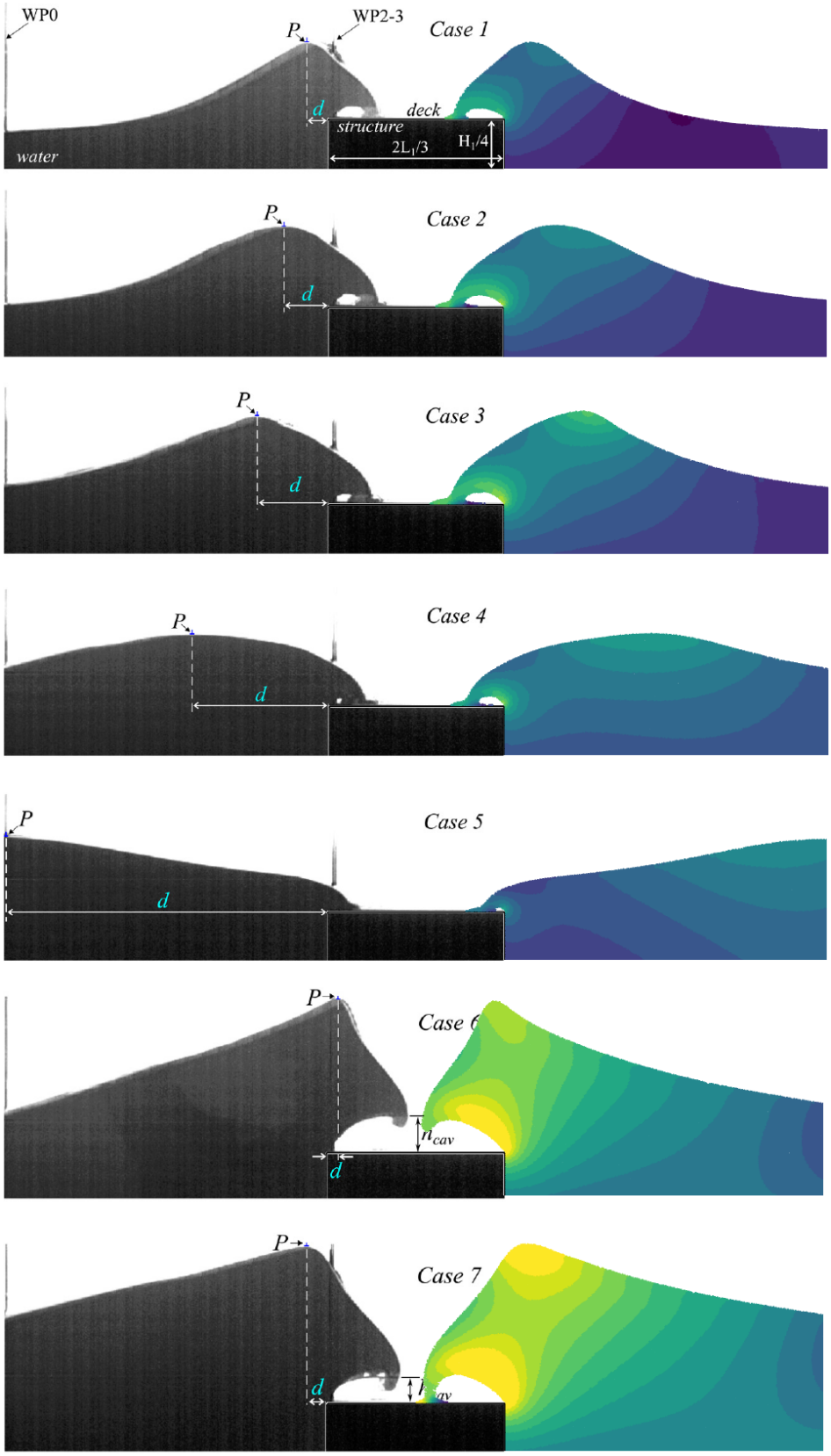

Figure 5.42.: The patterns found for the different cases. The experiment photographs [210] (left column) and simulation snapshots (right column) were taken 120 ms after the water level reached the deck level for each case.

dam breaks over a wet bed, the movement of the gate plays an important role for forming the incoming wave. The described numerical experiments needed to incorporate precise gate movement, as modifying the gate velocity for 10% leads to completely different wave pattern than the expected. Two probes, WP0 and WP1, which are depicted in figure 5.41 measured the water depth at locations 150 mm left and 250 mm right from the gate, respectively. Figure 5.43 compares the measurements made in two simulations to those obtained experimentally. Since the authors of the experiment note that the gate release was tuned to ensure opening times $t < \sqrt{(2h_1/g)}$ by using a free fall weight, it is challenging to reproduce the exact movements from the experiment. The two simulations differ only in the imposed movement for the gate. It can be seen that slightly varying the gate movement yield different steepness of the incoming wave. Nevertheless, the numerically obtained values at wave–probes are in very good agreement to the experimental data. It should be noted that the bump detected by WP1 is a small wave that travels backwards after opening the gate, due to a vortex forming by the accelerating flow around the moving gate bottom. Finally, the oscillations in numerical readings of the small backflow wave occur due to a simple, non-smooth implementation of wave probing in numerical simulations.

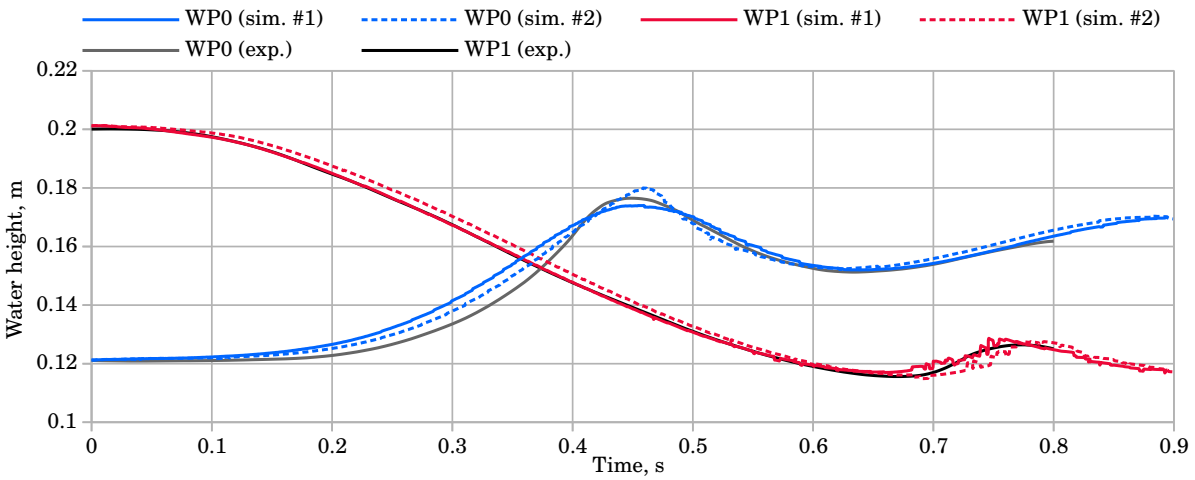

Figure 5.43.: Readings of the wave probes for two different simulation cases that differ only in imposed velocity of the gate, which are compared to the experimental measurements [210].

Without the gate modelled, the high water column falling under the action of gravity forms a mushroom-like jet structure [211]. The mushroom grows while travelling downstream and eventually the front breaks. The broken wave front continues travelling downstream, while inducing subsequent small-scale wave breakings. This flow development significantly affects the wave pattern on the deck, which is characterised mostly by separation of fluid clumps from the incoming wave, due to the built-up momentum at the interface. The development of these events and how they contaminate the resulting wave pattern is

rendered in figure 5.44. Validation of numerical methods for coastal engineering usually relies on experimenting with dam breaks and their complex patterns, such as mushroom-like jets. Therefore, the proposed method may be considered to be used for problems such as the tsunami wave run-up.

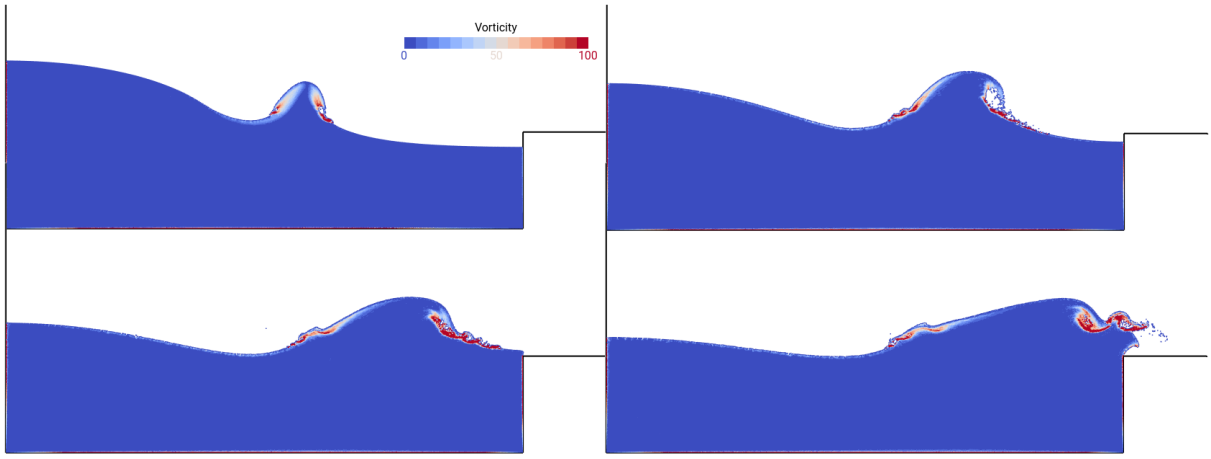

Figure 5.44.: The development of the wet dam–break flow simulated without the gate with shown areas of high vorticity and mushroom-like structures.

5.6.2. Force on the Deck

After validating the generation of isolated shipping water events, the following numerical experiments deal with vertical loads due to the shipping of water onto the deck. The numerical experiment is set-up according to the experiments conducted by Hernández-Fontes *et al.* [212]. The configuration of the experimental set-up is similar to that shown in figure 5.41, where $H = 475$ mm, $W = 500$ mm, $H_1 = 150$ mm, $L_1 = 392$ mm, $L_2 = 1258$ mm, and $L_3 = 300$ mm. Additionally, a force balance was embedded into the deck that measured the slowly varying vertical force on the deck. The heights of the water columns were chosen as $h_0 = 144$ mm and $h_1 = 240$ mm. The authors of the experiment note that the chosen ratio of water columns, $h_0/h_1 = 0.6$, yields a stable undular bore without three-dimensional effects due to wave breaking. A wave probe was located 5 mm in front of the deck edge, which measured the freeboard exceedance, i.e. the difference of the incoming wave and the deck heights. Time series of vertical loads were obtained experimentally from the force balance measurements. The experiments were repeated five times, and the averaged time series is used here to validate the simulation results. At the peak of the force curve, relative minimum and maximum deviations from the average curve are less than 2%. Due to the good repeatability, this experiment is taken as the first validation experiment for green water loads. The authors of the experiment also tried to validate the new convolution model to predict the loads. Results demonstrated that using the convolution model improved the representation of the time series of loads

compared with the traditional dam–break approach. It managed to capture the peaks and the decay tendencies observed in the experimental data in an approximated way, although significantly over-predicting the loads.

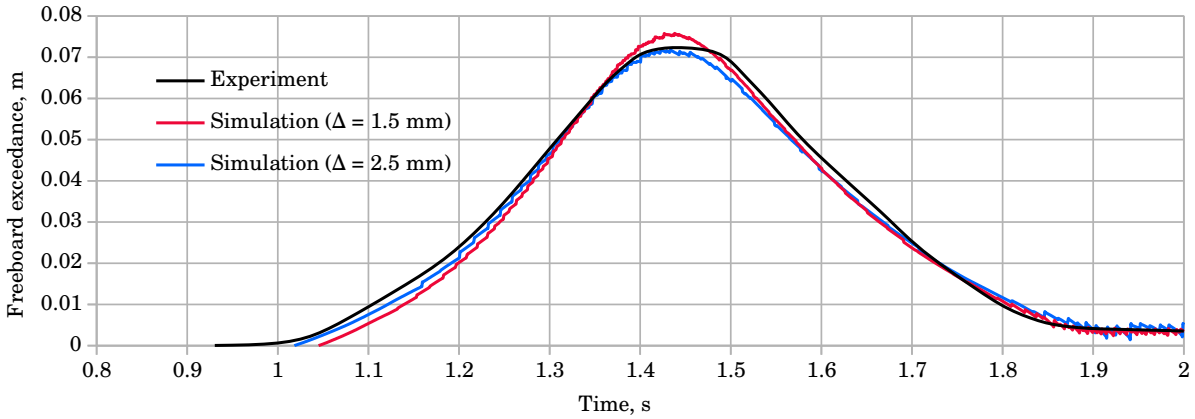

Figure 5.45.: Numerical and experimental readings [212] of the wave probe for the wave generated by the dam break. The simulations differ in imposed velocity of the gate.

Two simulations with initial point spacing of 2.5 mm and 1.5 mm were performed, which consisted of 44000 and 112000 number of points, respectively. The readings of the wave probe located in front of the deck edge are shown in figure 5.45 through values of freeboard exceedance. The finer discretisation yielded 3% higher freeboard exceedance compared to the coarse discretisation. On the other hand, both simulations had similar evolution of the force excreted on the deck, which are graphed in figure 5.46. The simulations have adequately predicted the rise of the force as the water enters the deck. The experiment has detected a sudden drop in pressure as water slides along the deck, which was not detected by the simulations and may be due to aeration effects, similar to those captured by the experiment shown in figure 1.7. Some amount of oscillations are present in the numerical curves before the peak, which are due to the high pressure–gradient area at the edge of the deck that is responsible for developing and imploding air pockets. The convolution and dam break models from [212] have excessively over-predicted the experiment measurements, while the results of the numerical simulations are in very good agreement with the experiment measurements.

5.7. Green Water on a FPSO Model

In this section, a validation of green water loads on a static structure is conducted. Several experimental results on green water have been published, but there is small amount of data available to be useful for CFD validation. Here the experiments conducted by Lee *et al.* [67] are numerically reproduced to understand the physics of green water and to

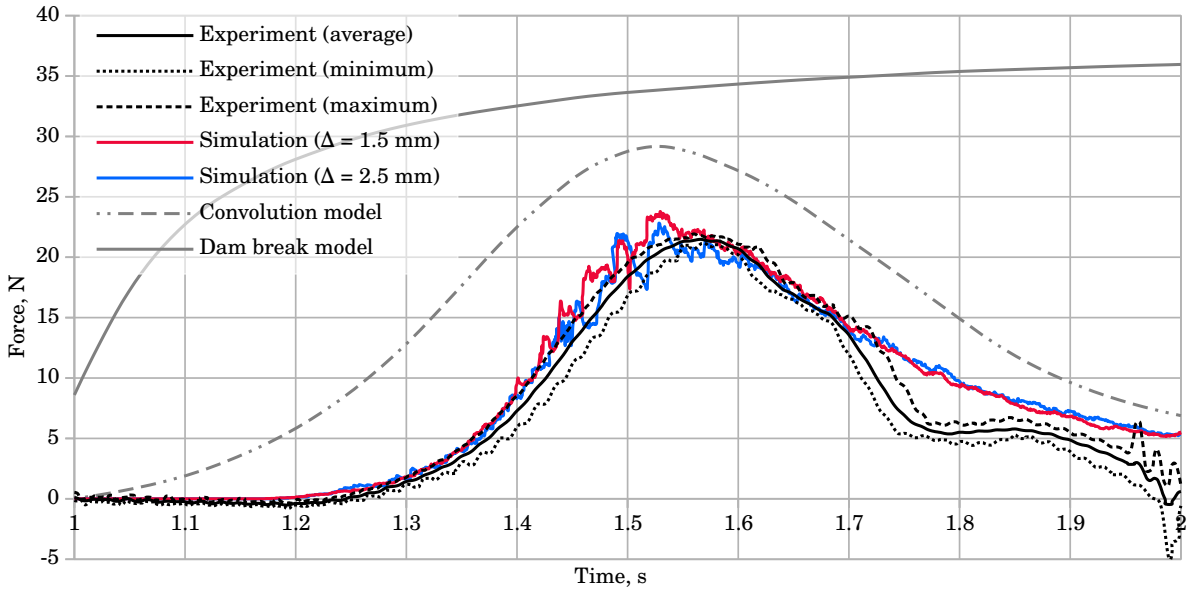

Figure 5.46.: Simulated and experimentally measured [212] force on the deck imposed by the wave that is generated by the dam break.

quantify the pressure distributions due to green water on deck. The model geometry and the locations of the pressure sensors are shown in figure 5.47.

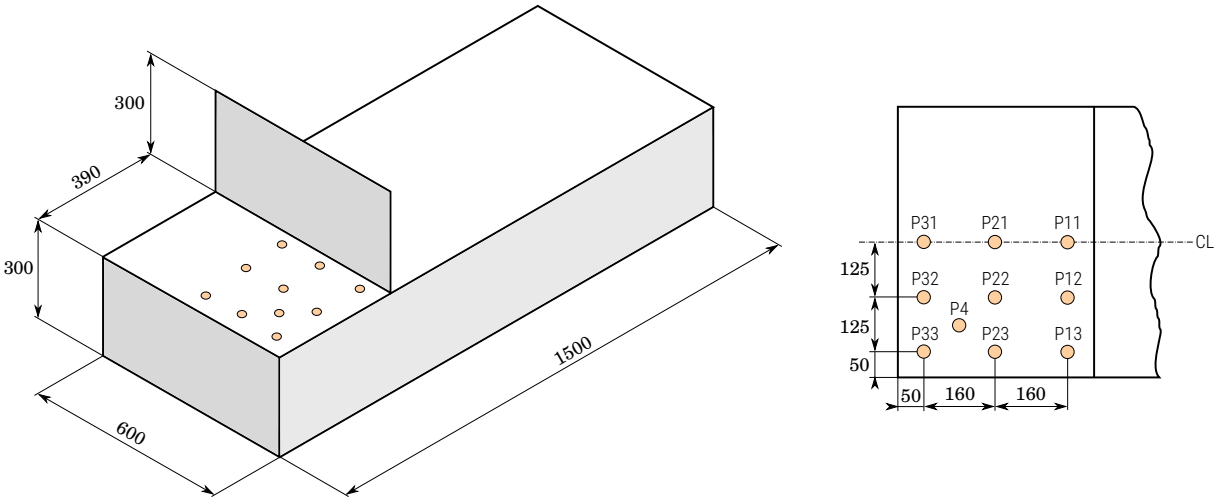

Figure 5.47.: Fixed FPSO model main dimensions and positions of the pressure sensors on the deck.

The shown geometry, named 'Rect0' by the authors of the experiment, is a simplified model of an FPSO vessel. Other tested geometries have a stem angle of 5 degrees (named 'Rect5') or a rounded edge deck with a vertically straight stem (named 'Round'). The model was fixed since the focus of the thesis was on the flow behaviour and pressure on deck due to green water. Ten pressure sensors were installed at the deck of the model, and the camera was recording the flow development on the deck. The validation study focuses on comparing pressure signals from five innermost sensors that record complex

pressure evolutions during green water events: P11, P12, P13, P21, P22. The results from all three different models showed very similar shapes with the largest pressure measured at P11. Nine regular wave cases were investigated by the authors of the experiment. The cases were chosen as a combination of three wavelengths (λ_W = 2.25, 3.0 and 3.75 m) and three wave steepness values ($2A_W/\lambda_W$ = 0.04, 0.05, and 0.06), which are listed in table 5.5. The experiment with Rect0 model and wave #9 is chosen to be reproduced for the validation study, which yields largest impact pressures.

Table 5.5.: Wave calibration results for the experiments conducted by Lee *et al.* [67].

Wave #	Target		Measured	
	A_W	λ_W	A_W	λ_W
1	45	2.25	45.07	2.25
2	56.25	2.25	56.74	2.25
3	67.5	2.25	67.13	2.25
4	60	3.0	60.38	3.001
5	75	3.0	74.11	3.003
6	90	3.0	90.56	3.002
7	75	3.75	75.37	3.746
8	93.75	3.75	90.84	3.749
9	112.5	3.75	109.8	3.75

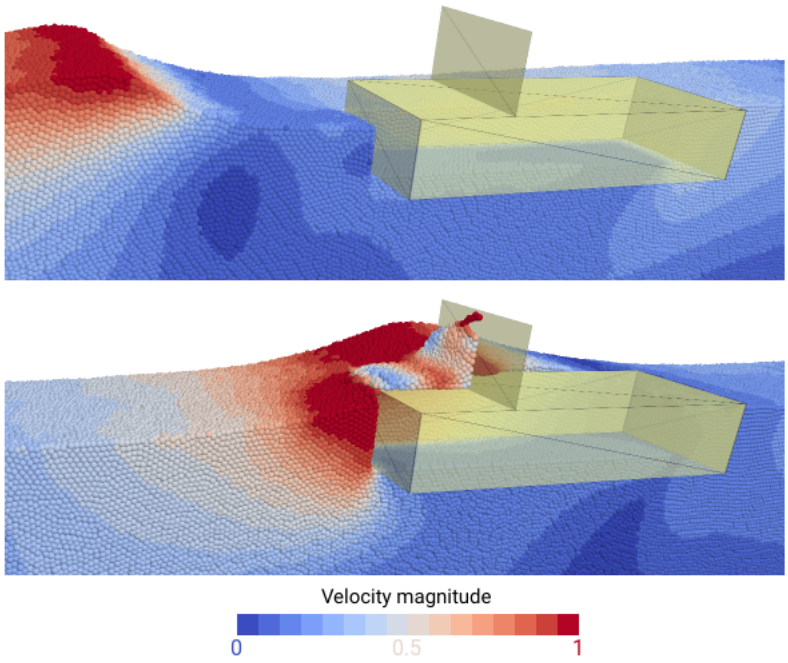

Figure 5.48.: Couple of snapshots captured during the green–water simulation defined with wave #9 and $\Delta = 20$ mm.

One way coupling of the potential–wave flow and meshless solver is done using the relaxation zones, i.e. by imposing known solution at the boundaries of the meshless domain and gradually transitioning to the nonlinear solution towards the middle of the domain.

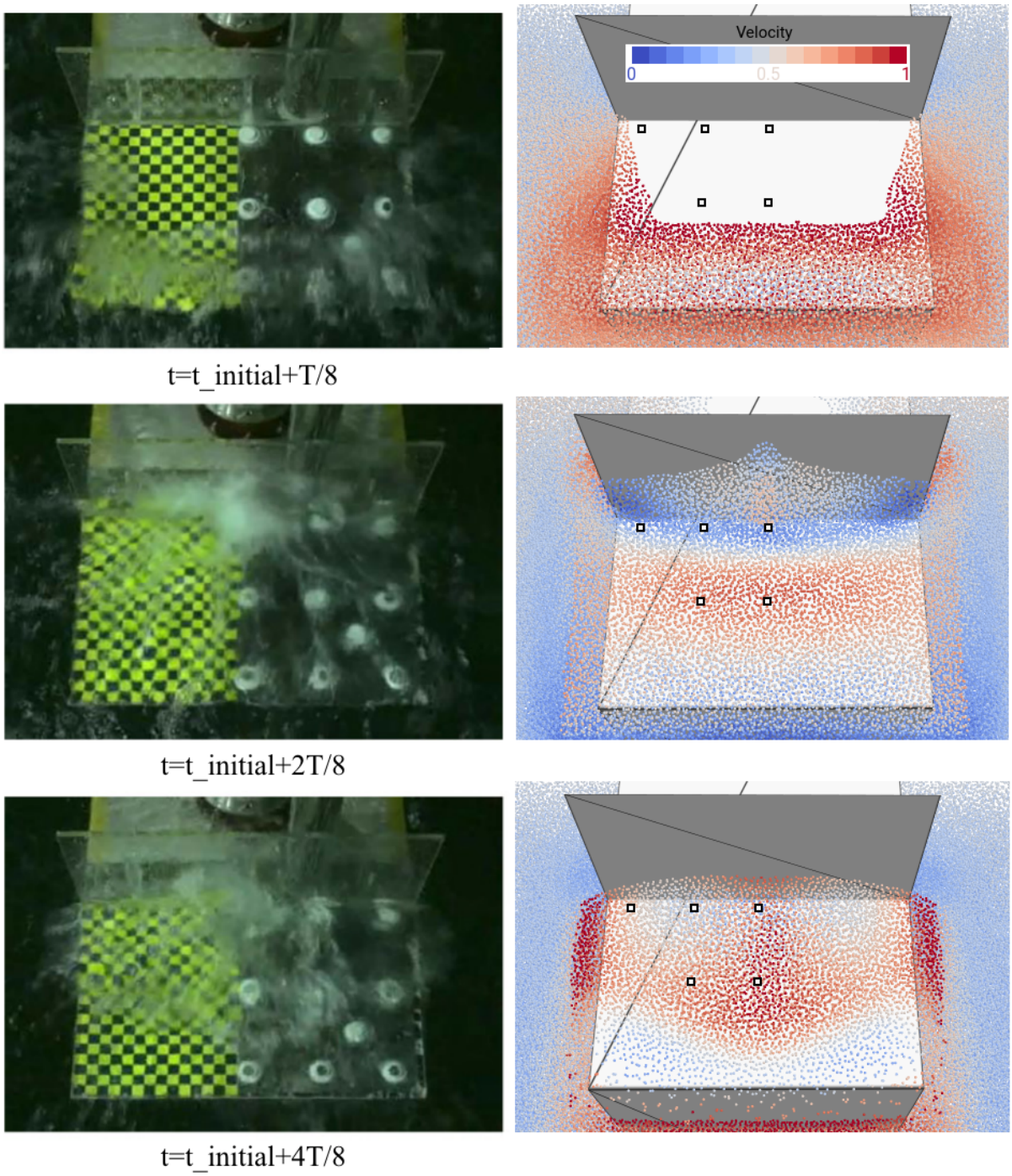

Figure 5.49.: Comparison of the experimental [67] (left column) and numerical (right column) green water behaviour on the deck for wave #9 at various time instants.

Literature is sparse when it comes to validating green water experiments by numerical means. A study published by Gatin *et al.* [30] is used for the comparison of the numerical results, in which the authors used unsteady RANS solver based on the FVM and VOF approach.

The domain size of the meshless solver was chosen to be small enough to save computa-

tional time, but to capture relevant nonlinearities. The domain extended 1.0 m in front of the model, 1.0 m back from the vertical wall, 0.5 m below the model, and 0.5 m from the sides of the model. The velocity known from the potential–flow solver was imposed at all domain boundaries, except for the downstream boundary, which imposed the open boundary condition. Multiple point spacing values were tested, and 12 mm was found to be adequate. A couple of screenshots in figure 5.48 show a wave run-up for larger point spacing $\Delta = 20$ mm.

The authors in [30] adjusted the time step during the simulations to maintain a maximum value of 0.75 for the CFL number, which corresponds to an average time step of 1 ms for their finest grid. The adaptive time stepping technique introduced in this thesis allowed time–step values that corresponded to classical CFL numbers larger than 1.0. For the sake of comparison, a constant time–step value of 3 ms was used for extracting the results of the study, which corresponds to a classical CFL number of 0.75 at time during the impact.

Firstly the behaviour of the computed green water events is investigated. The progress of a green water event may be seen in figure 5.49, which compares the experimental and numerical behaviour of flow on the deck for wave #9. The meshless points are coloured by the magnitude of their velocity vector. They are also rendered small so the deck can be visible. Typically, water ingress started at the fore end of the deck, and progressively water from the deck side merged. The merging built large momentum, which resulted in a violent impact against the wall. The water slided along the wall upwards, and then fell back on the deck, thus receding away from the wall and the deck. The numerical solver successfully reproduced the free surface evolution of the extremely nonlinear event, i.e. the ingress of water on the deck, piling against the wall during the impact, and falling of the water on the deck.

Table 5.6.: Comparison of average magnitudes of pressure peaks in a green–water event for wave #9 and relative deviations.

Experiment		Meshless simulation		FVM simulation	
Location	p, Pa	p, Pa	Rel. dev., %	p, Pa	Rel. dev., %
P11	2498	2403	–3.8	3697.7	+48.0
P12	1357	1615	+19.0	1877.3	+38.3
P13	977	1151	+17.8	1069.5	+9.5
P21	939	1033	–10.0	846.1	–9.9
P22	857	915	+6.7	719.6	–16.0

During the simulation run, pressure was registered at five relevant sensor locations using the meshless interpolation. The numerically obtained pressure signals of ten sequential green water events are plotted in figure 5.50 and compared to the experimental data and the results obtained by the FVM [30]. It should be noted that the plotted numerical

signals of 10 typical green water events are shifted in time in order to overlap with a part of the stable experimental signal for easier comparison. The pressure signals usually exhibits two peaks during the green water event. The first peak with a very short rising time is caused by the impact of the merged water front against the wall, and the second peak with a longer rising time is caused by the fall of the piled-up water. This resembles to typical sloshing events, some of which are simulated in section §5.5. The meshless solver managed to reproduce both peaks within a period of the green water event at all sensor locations. The pressure at location P11, where the impact pressure was most frequently observed, showed the largest uncertainty suggesting a strongly nonlinear and irregular feature of the green water impact [67]. The meshless simulation yielded somewhat more uniform peak magnitudes at location P11 than the experiment, i.e. the simulated magnitudes on impacts were in the range from 2900 Pa to 3500 Pa, and the magnitudes of the secondary peak were in the range from 1500 Pa to 1750 Pa. For comparison, the FVM has under-predicted the second peak in all cases, while the pressure peak at impact oscillated in the order of magnitude. In addition, the farther the pressure sensor was located from the wall and the centreline, that much miss-predicted the numerical signal shape was.

Lee *et al.* suggest that the average values of the sensors peak magnitudes are taken for validation of CFD solvers. The comparison of the average peak magnitudes is presented in table 5.6, and also graphed in figure 5.51. At the location P11, results of the numerical simulations in average have relative deviations less than 4% from the experiment, while results of the FVM have a discrepancy of 48% from the experiment, due to the uncertain pressure response during the initial impact of the wave front against the wall. Overall, the meshless simulation follows the experimental trend.

Table 5.7.: Comparison of average pressure impulses in a green–water event for wave #9 and relative deviations.

Experiment		Simulation		Gatin *et al.*	
Location	p, Pa	p, Pa	Rel. dev., %	p, Pa	Rel. dev., %
P11	938.1	842.5	–10.2	727.4	–22.5
P12	748.0	780.8	+4.4	632.9	–15.4
P13	642.8	668.0	+3.9	522.2	–18.8
P21	620.6	621.8	+0.2	452.8	–27.0
P22	550.5	634.2	+15.21	443.8	–19.4

However, the pressure–peak magnitude itself does not reveal enough information describing a green water event, or generally an impact event. The pressure impulse, or pressure integral over time, is a more relevant measure from the structural point of view. It was noticed that the results of experimental pressure integrals given in the appendix of [30] are lower up to 10% than those obtained by the author from the data shared by Rhee and Seok. Due to the consistency with the graphed pressure signals in figure 5.50, the integrals of those graphed pressure signals are considered in this validation study.

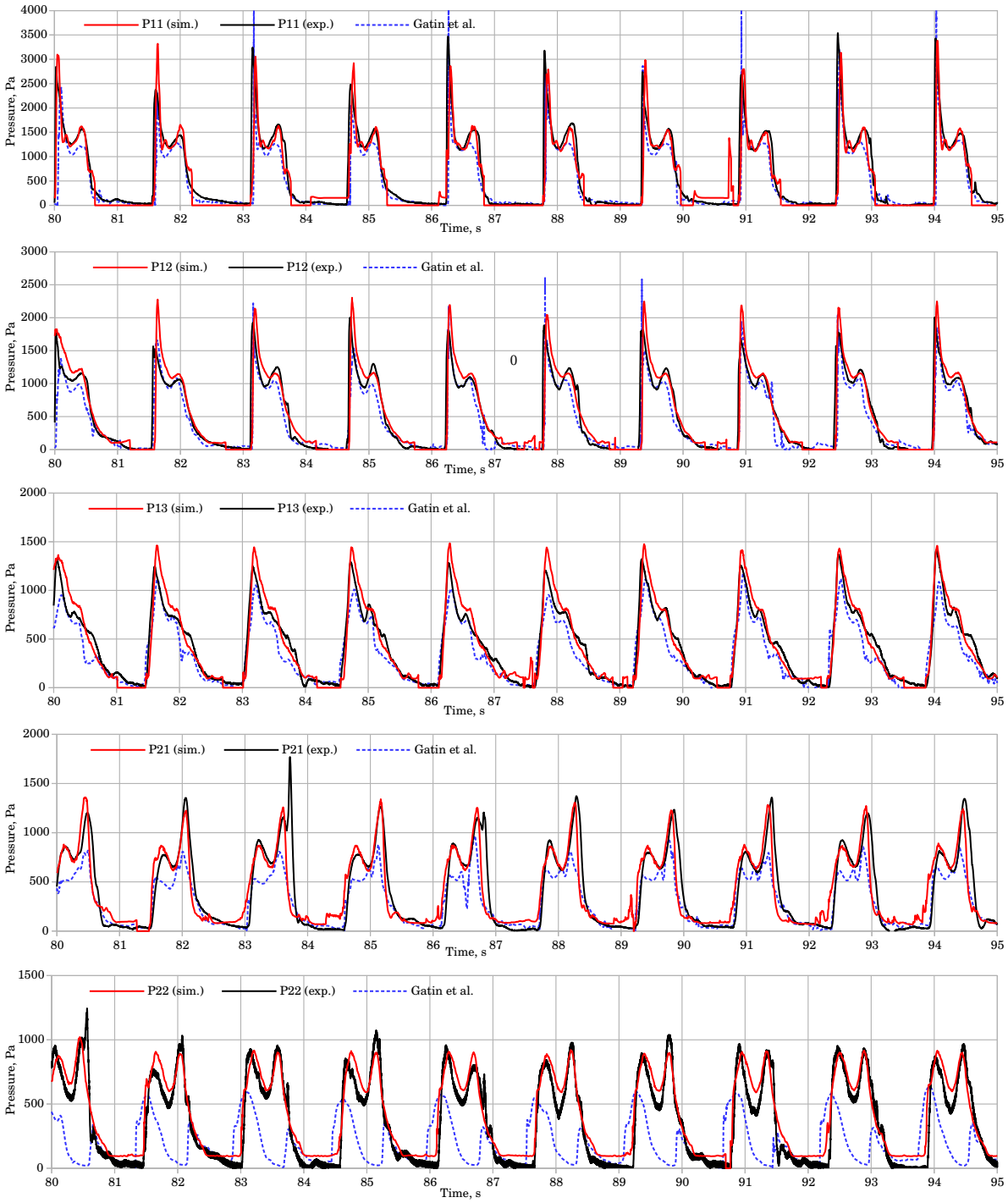

Figure 5.50.: Comparison of pressure readings obtained numerically and experimentally [67] for wave #9 at locations (top to bottom): P11, P12, P13, P21, P22.

The comparison of the pressure impulse is presented in table 5.7, and also graphed in figure 5.52. By comparing tables 5.6 and 5.7, it is evident that the peaks and impulses are not strongly correlated. For example, the impulse at P21 is accurately predicted, while the average pressure peak magnitude is under-predicted by 10%. In addition, the pressure impulse at P11 obtained by the FVM is under-predicted, while the peak magnitudes are over-predicted. The results of the meshless simulation at locations P12, P13, and P21

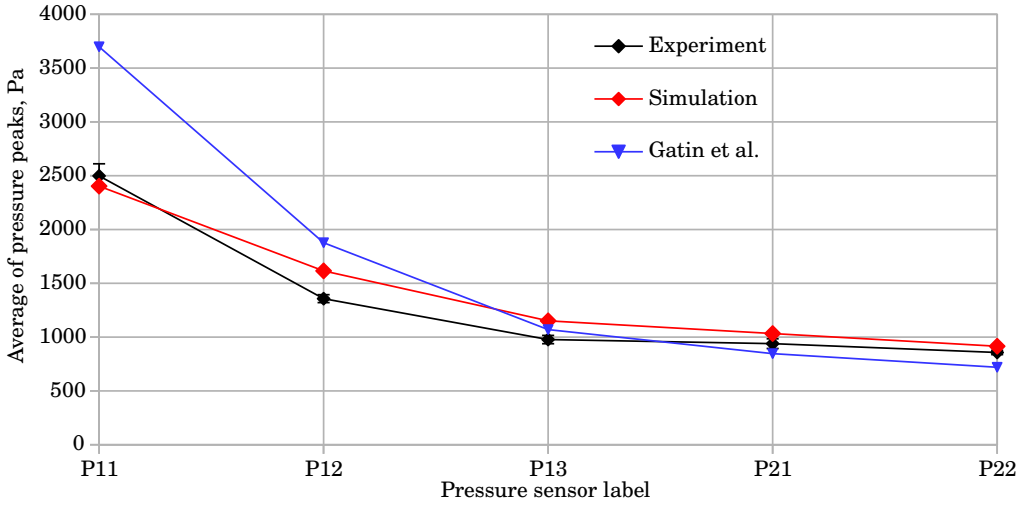

Figure 5.51.: Graphed comparison of average magnitudes of pressure peaks in a green–water event for wave #9.

are in extremely good agreement with the experimental results, while the results at P11 and P22 are slightly under- and over-predicted, respectively. Looking at the graph of the signal at location P11, it can be noticed that the simulation does not smooth-out the pressure while water is receding from the location of the sensor. Similarly, at P22 a single row of Lagrangian points remains there at all time, i.e. the deck stays wet. This is due to the relatively large point spacing value $\Delta = 12$ mm for such conditions. In either case, those points that remain at the deck influence the pressure signal only in between the impacts, which is irrelevant from the structural point of view, but may be relevant if the stability of the vessel is investigated.

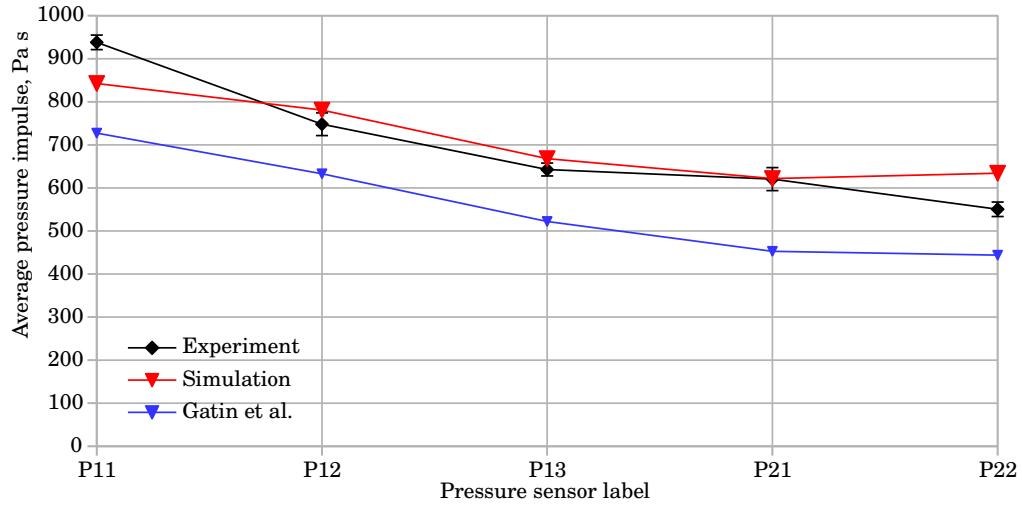

Figure 5.52.: Graphed comparison of average pressure impulses in a green–water event for wave #9.

The culprit of non-smooth ending of the pressure signal is that Lagrangian points are either there or not, therefore contributing or not contributing to the hydrostatic pressure,

i.e. wetting the deck. This may be alleviated by making the points spacing smaller, which would lead to many unnecessary points in the whole simulation domain. A current limitation of the implementation is that each point is considered to represent the same amount of fluid volume. In Eulerian methods, this is equivalent to using a uniform grid cell size throughout the domain. A solution to this limitation is to size down both point spacing and the domain volume, but not too excessively that would contaminate the results by too close boundaries imposing the flow. Future work will deal with implementing adaptively sized points spacing in order to allow for better spatial precision in areas where user specifies. On the other hand, one may choose to couple an FVM solver and the meshless solver in a way that the meshless domain captures only the deck and its structures. This discrepancy arising from the pressure in between impacts is not of a significant importance when analysing structural loads. Therefore, it can be concluded that the introduced meshless method and the domain decomposition technique can be used to accurately predict green water loads.

5.8. Heaving Vessel

The following numerical experiment is used to test the robustness of the implemented solver by simulating the FPSO model experiment by adding complexity of prescribed vessel movement. The vessel model oscillates in the vertical direction, i.e. it heaves with an amplitude of 50 mm. The amplitude of incoming waves is 112.5 mm, and the period is constrained to $T_W = 1.55$ s, but two heaving periods of the vessel were chosen in the simulations: $T_H = 0.75$ s and $T_H = 1.0$ s.

The numerically obtained pressure signals of four sequential green water events on the moving vessel are plotted in figure 5.53 and compared to the numerical results obtained for the fixed vessel. Since the value of the first chosen heaving period equals to almost half of the value of the waves period, it is expected that the pressure signals will repeat over time. Indeed, figure 5.53 shows that the signal is similar for each green water event, except for some discrepancy of secondary pressure peaks at P21. It is observable that this heaving period significantly amplifies magnitudes of the secondary pressure peak, and alleviates magnitudes of the primary pressure peak. This is due to the vertical acceleration of the deck that occurs while the piled-up water falls on the deck. The second chosen heaving period $T_H = 1.0$ is not in scale of the waves period. The pressure signal varies from event to event. Overall, it can be deduced that magnitudes of the primary pressure peak are significantly amplified, while magnitudes of the secondary pressure peak are only slightly modified.

It should be noted that computational performance of the solver employing moving geometry is the same as if the geometry is fixed. Methods based on conforming meshes

would have to perform mesh deformation and advection corrections each time step. In combination with physically moving geometry, Lagrangian fluid advection without a mesh simulates physics of green water events just as it is.

5.9. Discussion of the Results

The V&V study consisted of problems, which contain all relevant phenomenons that occur in green water events, in which the set of problems used for the verification had somewhat simpler set-ups than validation problems. For example, slamming includes an area of extremely high pressure that moves as the body enters the water, which generates a thin and strong jet. Dam break problems are opposite of slamming problems, since water moves and impacts a fixed wall, producing high pressure gradients and pile of water along the wall that falls. Sloshing problems include both moving walls and water, which can generate various nonlinear phenomenons, such as wave–breaking, splashing, spraying, etc. Therefore, it can be stated that the verification study has captured a range of problems of ship hydrodynamics under various conditions. The numerical results were in very good agreement with the experimental measurements, showing that the method can handle violent impacts of fluid onto walls, while accurately reproducing impact pressures. In addition, the solver managed to adequately reproduce expected flow kinematics, i.e. free surface evolution and velocity field from the experiments.

Moreover, a green–water event can comprise of all of separately tested phenomenons, i.e. a breaking wave on the moving deck impulsively impacts a moving wall while generating a jet, falling pile of water, spray, etc. The validation cases of isolated events of wetting the deck, and periodic green water events in regular seas, were successfully simulated in a short amount of time. The study showed that wave impact pressures and pressure impulses can be predicted with very good accuracy. Furthermore, a test with moving ship has shown the capability of the method to take all motions into account through the domain–decomposition approach.

There is an infinite amount of possible combinations of conditions formed by the 6–DOF vessel movement and incoming waves that can occur in harsh ocean environments. Potential flow methods can quickly approximate vessel motions in waves, but this V&V study has shown that unsteady CFD analyses should be made for questionable combinations of conditions. The introduced method has been shown to efficiently and robustly simulate local green water events and accurately predict hydrodynamic loads. A solver implemented based on the introduced method can simulate questionable conditions relatively quickly on the modern PC. The validation cases have shown that the domain may be small, taking into account only the space where nonlinearities occur. The number of points forming fluid in such small domains may be less than 200000. A modern high-end

GPU can compute one time step in less than 200 ms, i.e. it takes less than 1 microsecond per point per time step. Considering that the method is Lagrangian and can handle large classical CFL numbers, simulations can be made relatively quickly. To the author's knowledge, this is the fastest reported implicit Lagrangian method, which is still not fully optimised.

Results of the validation cases could benefit adaptive refinement techniques, which were not implemented in the solver. By choosing larger point spacing and adaptively refining the discretisation near high–pressure areas, the domain could be made larger with the same number of points. On another note, the meshless domain can be made as small as possible by taking flow input from another mesh–based, e.g. FVM, solver. In either case, these issues are not directly connected to the methodology, but to the implementation of the solver.

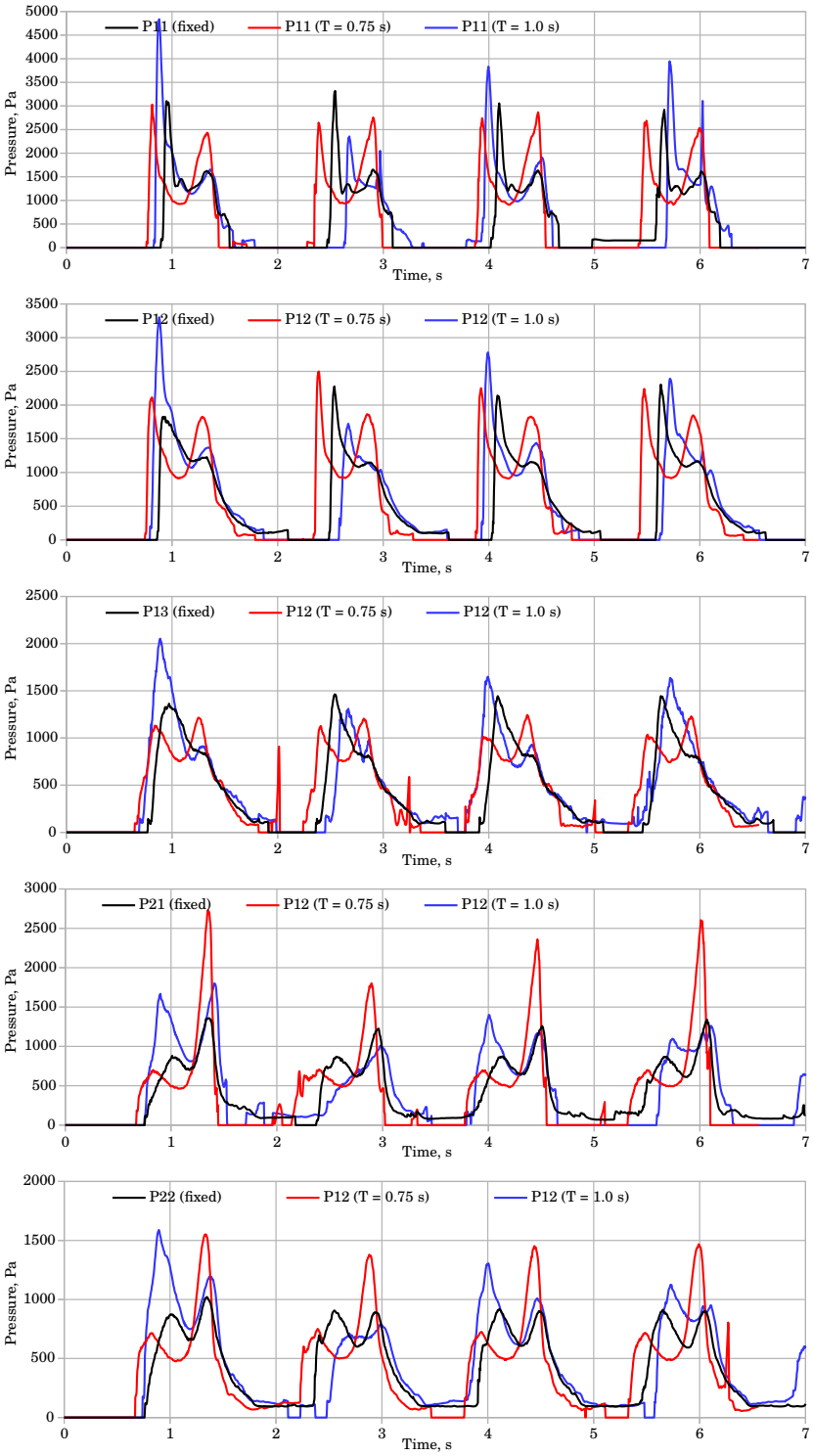

Figure 5.53.: Comparison of simulated pressure readings for wave #9, for the fixed and heaving models at sensor locations (top to bottom): P11, P12, P13, P21, P22.

6. Conclusions and Future Work

6.1. Conclusions

Large relative motions between a vessel and water can result in violent water dynamics so that the water flows onto the deck and reaches crucial equipment and other deck structures. Damages done by water impacts may be critical for small vessels, or may lead to loss of production time for large vessels. Predicting the effects of green water on the stability and structure of the vessel is a challenging problem, which needs to be assessed in the design process. Many numerical and experimental methods may be found in literature that try to solve these problems, but rarely such methods are designed specifically for the green–water problem. By analysing the limitations of currently used methods, a rationale was developed and the research resulted in a novel design tool, i.e. a methodology for incompressible flows suitable for simulating green water events. The introduced methodology can be described as: meshless, Lagrangian, volume–conservative, FD–based, second–order accurate, efficient and suitable for coupling.

As the Lagrangian description of the flow is suitable for violent free–surface flows, the mesh-free description of the fluid volume is assumed. The WLS–based second–order gradient in its generalised FD form works on highly irregular meshes and point clouds. The disadvantage was that the WLS did not specify a simple and accurate Laplacian, which is the main aspect for adequately solving a Poisson equation. In this thesis, the Laplacian is defined by extending the WLS method used for the gradient. The introduced Laplacian was shown to be efficient, having linear convergence rate even for extremely irregular point arrangements. The generalised second–order FD operators were employed to simulate incompressible fluid flow by solving the velocity–pressure formulation of the NSE in strong form. The combination of meshless fluid description and novel boundary conditions results in eliminating the meshing pre-processing step and enabling free advection of fluid that adjusts to boundaries at each time step. Furthermore, possible loss or gain in fluid volume during the Lagrangian advection is prevented by enforcing equidistant neighbours using the PBD method. It was shown that the volume is conserved at all time.

The implementation of the methodology was extensively verified and validated by simulating various problems that include violent fluid–structure interaction. The set of problems

used for the verification included problems with simpler set-ups. The novel Laplacians were verified on a set of approximation and Poisson problems that are artificially created. The lid–driven cavity problem was then simulated to verify the ability of the method to maintain unconditional stability within a confined volume, which is a challenging problem for Lagrangian methods. The first set of validation experiments were entries of bodies in still water, which included a wedge and a section of a ship bow. The pressure results were found to be in good agreement with the experimental measurements. The dam break experiments and sloshing experiments in rectangular and LNG–type tanks were used to validate that the method can handle violent impacts of fluid onto walls, while accurately reproducing impact pressures.

The simulation results were found to be in very good agreement with the experimental data from the literature. The numerical solver managed to adequately reproduce expected flow kinematics (free surface evolution and velocity field) and dynamics (pressures and forces) from the experiments. This confirmed the methodology to be ready for validation by simulating green water events. Moreover, it can be argued that the method has potential to become a general design tool when impacts between a structure and incompressible or nearly–incompressible fluid need to be predicted.

The final validation was performed by simulating isolated events of green water on a fixed structure, and regular waves wetting the deck of a FPSO model. Isolated green water events were simulated using the wet dam–break approach. The incoming waves with bores and plunging were successfully reproduced, as well as the force due to the shipping of water onto the deck. A fixed model of an FPSO was simulated on a regular wave–field by the domain decomposition approach. The study showed that wave impact pressures and pressure impulses can be predicted with very good accuracy. The FPSO was imposed with heaving motions to study the effect of vessel movement on the loads, and the applicability of the method on realistic scenarios where the vessel may move with 6-DOF.

Since the verification and validation study has captured a range of problems of marine hydrodynamics under various conditions, it may be concluded that the proposed method provides good estimates of the evolution of pressure in fluid–structure problems, which may be used for structural analyses.

Besides coupling to another flow solver, the flow solver implemented through the introduced methodology is ready to be coupled to a structural or a rigid–body solver. The flow solver handles geometry described by triangles and quadrilaterals, so the discrete structural model may be directly used in the fluid simulation, which makes the transfer of loads straightforward.

Finally, it should also be noted that the solver was implemented to run fully in parallel, which enables state-of-the-art simulations to be performed efficiently on personal com-

puters, proving the efficiency of the methodology. It was shown how detailed simulations are computed in a reasonable amount of time, faster than contemporary methods used for simulating same or similar problems.

To conclude the thesis:

Mesh-free FD approximations and the volume–conservative Lagrangian description of flow are employed to accurately solve the NSE and naturally handle free surface flows. This combination is appropriate for solver coupling and simulating FSI in harsh ocean environments.

6.2. Proposals for Future Work

The present methodology can further be improved in several directions. Some of the proposed improvements related to numerical features and physical modelling are listed as follows.

Domain Coupling Enhancements

Regular waves are rarely found in the environment, therefore, irregular waves are of great practical interest. First–order irregular waves are represented through the superposition of a set of linear regular waves with varying frequencies and amplitudes. The short–term prediction of irregular waves is described statistically, and is applicable to numerical wave generation. Future work will deal with modifying the implemented wave generation procedure to generate irregular waves of specified spectrum.

Besides one–way coupling, the method is mature for two–way coupling with another mesh–based flow solver. A Lagrangian solver can be reliably coupled to a Eulerian solver, while both solvers differ in the space and temporal discretisation [116]. It will be tested if the two–way coupled simulation can benefit the global vessel seakeeping problem, in which the Eulerian solver resolves the bulk flow and the Lagrangian solver resolves the regions where the complex free surface flow develops.

Two–Way Structural Coupling

As the introduced method generates boundary points each time step by projecting to nearby boundary geometry, it naturally handles deformations and movements of the geometry within the simulation. Without introducing any complexity, a partitioned fluid–structure interaction may be implemented through data exchange between the meshless solver and some chosen FEM solver. The discrete triangular or quadrilateral structural

meshes will be used directly in the fluid simulation, so the data exchange from the fluid solver to the structural solver is straightforward through the meshless interpolation, while the mesh elements and nodes movements are obtained from the structural solver. The ability of multi-physics coupling of the method will be exposed through some popular frameworks, such as preCICE [213] or MOOSE [214].

Adaptive Point–Cloud Refinement

The solver implementation should support variable spacing between points. A point virtually describing some amount of volume can be split into multiple smaller points where needed. On the other hand, points with smaller compact radius can be merged into a point with larger compact radius where the finer discretisation is not needed to represent flow structures. The list of criterions for splitting a point could include: distance from the body or free surface, overlapping with user–specified virtual bounding boxes, too excessive pressure or velocity gradient, etc. A criterion to merge group of split points can be a lack of pressure or velocity gradient. When merging a group of points, the resulting point is placed at the centre of gravity of the original particles. Splitting of a point into multiple points is performed by generating a new set of particles within the radius of the parent particle. Researchers of the SPH method have developed various techniques for the adaptive refinement, e.g. [215, 216], which are going to be considered for implementation.

Optimisations

The implemented algorithms should be thoroughly optimised for memory access. Cached and coalesced memory access reduce the computing time to an order of magnitude, because randomised or scattered global memory access can be more than 100 times slower. Since the meshless points are placed in a Cartesian grid, Morton or Z–curve hashing may be used for the sorting of allocated memory for the point cloud. The hashing assures that the data which Cartesian cells holds is close in space, but also close in the memory space. This improves caching. Furthermore, the shared memory model of accelerating devices will be also used, which offers extremely fast access to data.

The current implementation of the proposed methodology runs on a single node, i.e. one chosen device in the system, a CPU or a GPU. The base framework uses efficient node–level parallelism optimisations to get most out of the hardware it runs on. For some data–heavy problems that require large amount of simulating points, it would be convenient to split the problem into multiple problems and distribute the workload and memory among multiple computing nodes. A portable message–passing standard will be used for the communication between multiple instances of the solver that runs concurrently on

several homogeneous or heterogeneous nodes, i.e. multiple GPUs, CPUs, or CPU–GPU combinations.

When solving the PPE, i.e. the discretised linear system of equations, iterative Krylov subspace solvers use the Jacobi preconditioner to speed up the convergence. The preconditioner scales linear equations by the inverse of matrix diagonal coefficients, which is very effective for matrices having dominant diagonals. The parallel initialisation of preconditioners is an actively researched problem. In order to speed up and make the convergence more robust, parallely initiable preconditioners other than Jacobi's should be tested.

Flux–Based Solution

In the presented numerical approach, interactions between mesh-free points were treated with FD–like operators rather than assigned conserved quantities. Alternatively, one may attempt to utilise a Godunov approach by solving a hydrodynamic Riemann problem in a moving frame, between each pair of interacting particles, and thus obtain the fluxes from the solution [118]. Godunov methods have traditionally been used on Eulerian grids which introduce advection and angular momentum conservation problems. In Lagrangian context, the Riemann problem should be solved across faces distorting with the relative fluid flow, although the assumed motion of the face in the Riemann problem will not exactly match the real motion of the face [119]. Future work will include an attempt to implement a meshless Godunov approach and compare its performance and accuracy to those obtained by the proposed methodology.

Turbulence

For internal flows, and in general, LES allows the solution of more complex problems. Modelling of the Navier–Stokes equations by a meshless Lagrangian approach can be revisited from the point of view of Large Eddy Simulation (LES) [165]. The LES filtering procedure can be recast in a Lagrangian framework by defining a filter that moves with the positions of the fluid particles at the filtered velocity [217].

Bibliography

[1] P. Temarel, W. Bai, A. Bruns, Q. Derbanne, D. Dessi, S. Dhavalikar, N. Fonseca, T. Fukasawa, X. Gu, A. Nestegard, A. Papanikolaou, J. Parunov, K. H. Song and S. Wang. Prediction of wave-induced loads on ships: Progress and challenges. *Ocean Engineering*, 119:274–308, 2015. DOI: 10.1016/j.oceaneng.2016.03.030.

[2] S. Tan. Observations Made on Board of Dutch Merchant Ships. *International Shipbuilding Progress*, 16(181):259–276, 1969.

[3] B. Buchner. *Green water on ship-type offshore structures*. PhD thesis, Delft University of Technology, 2002, page 279.

[4] A. Colman. Report of the Re-opened Formal Investigation into the Loss of the MV "Derbyshire". Technical report, High Court of Justice, Norwich, United Kingdom, 2000, page 321.

[5] W. Morris, J. Millar and B. Buchner. Green Water Susceptibility of North Sea FPSO/FSUs. In *15th Conference on Floating Production Systems (FPS)*, page 23, London, 2000.

[6] G Ersdal, a Kvitrud, J. S. Chung, R. M. W. Frederking, H Saeki and W Koterayama. Green water on Norwegian production ships. *Proceedings of the 10th (2000) International Offshore and Polar Engineering Conference, Vol I*, 1:211–218, 2000.

[7] R. Newton. Wetness Related to Freeboard and Flare. In *Summer Meeting RINA*, Paper No. 3, 1959.

[8] R. Tasaki. On Shipment of Water in Head Waves. In *Tenth ITTC*, London, 1963.

[9] M. Ochi. Extreme Behaviour of a Ship in Rough Seas Slamming and Shipping of Green Water. In *Annual Meeting SNAME*, 1964.

[10] S. E. Hirdaris, W. Bai, D. Dessi, A. Ergin, X. Gu, O. A. Hermundstad, R. Huijsmans, K. Iijima, U. D. Nielsen, J. Parunov, N. Fonseca, A. Papanikolaou, K. Argyriadis and A. Incecik. Loads for use in the design of ships and offshore structures. *Ocean Engineering*, 78:131–174, 2014. DOI: 10.1016/j.oceaneng.2013.09.012.

[11] K. M. T. Kleefsman, G. E. Loots, A. E. P. Veldman, B. Buchner, T. Bunnik, E. Falkenberg, A. E. P. Veldman, B. Buchner, T. Bunnik and E. Falkenberg. The numerical simulation of green water loading including vessel motions and the incoming wave field. *ASME 2005 24th International Conference on Offshore Mechanics and Arctic Engineering*, (Omae):981–992, 2005. DOI: 10.1115/OMAE2005-67448.

[12] B. Buchner. The Impact of Green Water on FPSO Design. In *Offshore Technology Conference*, pages 45–57, 1995. DOI: 10.4043/7698-MS.

[13] B. Buchner and A. Voogt. The effect of bow flare angle on FPSO green water loading. In *Proceedings of Offshore Mechanics and Arctic Engineering (OMAE2000)*, page 8, New Orleans, 2000.

[14] D. L. Kriebel and T. H. Dawson. Distribution of Crest Amplitudes in Severe Seas With Breaking. *Journal of Offshore Mechanics and Arctic Engineering*, 115(1):9–15, 1993.

[15] C. Stansberg and S. Karlsen. Green Sea and Water Impact on FPSO in Steep Random Waves. In *Practical design of ships and other floating structures - Proceedings of the Eighth International Symposium on Practical Design of Ships and Other Floating Structures*, volume I, pages 593–601. Elsevier, 2001. DOI: 10.1016/B978-008043950-1/50075-2.

[16] Y. Ogawa, R. Matsunami, M. Minami, K. Tanizawa, M. Arai, A. Kumano and R. Miyake. Green Sea Loads on General Cargo Ship. In *Proceedings of 8th International Conference on the Stability of Ships and Ocean Vehicles*, pages 97–109, 2003.

[17] I. Watanabe, M. Ueno and H. Sawada. Effects of Bow Flare Shape to the Wave Loads of a container ship. *Journal of the Society of Naval Architects of Japan*, 1989(166):259–266, 1989. DOI: 10.2534/jjasnaoe1968.1989.166_259.

[18] J. O'Dea and D. A. Walden. The effect of bow shape and nonlinearities on prediction of large amplitude motions and deck wetness. In *Proceedings of the 15th Symposium on Naval Hydrodynamics*, pages 163–176, 1984.

[19] A. R. J. M. Lloyd. The Effect of Bow Shape on Deck Wetness in Head Seas. Technical report, Naval Systems Engineering Department, 1984, page 125.

[20] H. Vermeer. Prediction of the amount of shipping water. Technical report, Netherlands Maritime Institute, Delft, 1980.

[21] Ø Hellan, O. Hermundstad and C. Stansberg. Design Tool for Green Sea, Wave Impact, and Structural Response on Bow and Deck Structures. In *Offshore Technology Conference*, page 9. Offshore Technology Conference, 2001. DOI: 10.4043/13213-MS.

[22] X. P. Pham and K. S. Varyani. Evaluation of green water loads on high-speed containership using CFD. *Ocean Engineering*, 32(5-6):571–585, 2005. DOI: 10.1016/j.oceaneng.2004.10.009.

[23] K. S. Varyani and X. Pham. Whaleback forecastle for reducing green water loading on high-speed container vessels. *Ships and Offshore Structures*, 3(3):229–237, 2008. DOI: 10.1080/17445300802057407.

[24] C. A. Bellezi, L.-Y. Cheng and K Nishimoto. Particle based numerical analysis of green water on FPSO deck. In *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, page 8, Nantes, 2013. DOI: 10.1115/OMAE2013-11553.

[25] C. Bellezi, L.-Y. Cheng and K. Nishimoto. A numerical study of the effects of bow shape on green water phenomenon. In *Proceedings of the International Offshore and Polar Engineering Conference*, pages 734–745, 2013.

[26] G. Colicchio, M. Greco, C. Lugni and O. M. Faltinsen. Towards a fully 3D domain-decomposition strategy for water-on-deck phenomena. In *Journal of Hydrodynamics*, volume 22 of number 5 SUPPL. 1, pages 445–450. Elsevier, 2010. DOI: 10.1016/S1001-6058(09)60237-7.

[27] D. F. de Carvalho e Silva and R. R. Rossi. Green Water Loads Determination for FPSO Exposed to Beam Sea Conditions. In *ASME 2014 33rd International Conference on Ocean, Offshore and Arctic Engineering*, volume 8B, V08BT06A012. ASME, 2014. DOI: 10.1115/OMAE2014-23947.

[28] K. R. Drake. Wave profile characterisation of green water loading events from model test data. *Applied Ocean Research*, 23(4):187–193, 2001. DOI: 10.1016/S0141-1187(01)00021-9.

[29] O. M. Faltinsen, M Greco and M Landrini. Green Water Loading on a FPSO. *Journal of Offshore Mechanics and Arctic Engineering*, 124(2):97–103, 2002. DOI: 10.1115/1.1464128.

[30] I. Gatin, V. Vukčević, H. Jasak, J. Seo and S. H. Rhee. CFD verification and validation of green sea loads. *Ocean Engineering*, 148:500–515, 2018. DOI: 10.1016/j.oceaneng.2017.10.026.

[31] M. Gómez-Gesteira, D. Cerqueiro, C. Crespo and R. A. Dalrymple. Green water overtopping analyzed with a SPH model. *Ocean Engineering*, 32(2):223–238, 2005. DOI: 10.1016/j.oceaneng.2004.08.003.

[32] C. Gong, R. C. Zhu, G. P. Miao and J. Fan. A numerical method of the simulation of green water on the deck of a vessel. *Chuan Bo Li Xue/Journal of Ship Mechanics*, 18(5):524–531, 2014. DOI: 10.3969/j.issn.1007-7294.2014.05.006.

[33] M Greco, G Colicchio, C Lugni and O. M. Faltinsen. 3D domain decomposition for violent wave-ship interactions. *International Journal for Numerical Methods in Engineering*, 95(8):661–684, 2013. DOI: 10.1002/nme.4523.

[34] M. Greco, C. Lugni and O. M. Faltinsen. Can the water on deck influence the parametric roll of a FPSO? A numerical and experimental investigation. *European Journal of Mechanics - B/Fluids*, 47:188–201, 2014. DOI: 10.1016/j.euromechflu.2014.01.009.

[35] M. Greco and C. Lugni. 3-D seakeeping analysis with water on deck and slamming. Part 1: Numerical solver. *Journal of Fluids and Structures*, 33:127–147, 2012. DOI: 10.1016/j.jfluidstructs.2012.04.005.

[36] M. Greco, B. Bouscasse and C. Lugni. 3-D seakeeping analysis with water on deck and slamming. Part 2: Experiments and physical investigation. *Journal of Fluids and Structures*, 33:148–179, 2012. DOI: 10.1016/j.jfluidstructs.2012.05.009.

[37] B. Hamoudi and K. Varyani. Significant load and green water on deck of offshore units/vessels. *Ocean Engineering*, 25(8):715–731, 1998. DOI: 10.1016/S0029-8018(97)10005-1.

[38] T. E. Kendon, C Pakozdi, R. J. Baarholm, P. A. Berthelsen, C. T. Stansberg, S Enger and Asme. Wave-in-Deck Impact: Comparing CFD, Simple Methods, and Model Tests. *Proceedings of the Asme 29th International Conference on Ocean, Offshore and Arctic Engineering, 2010*, 4:495–509, 2010. DOI: 10.1115/OMAE2010-20860.

[39] M. Landrini, A. Colagrossi, M. Greco and M. P. Tulin. The fluid mechanics of splashing bow waves on ships: A hybrid BEM-SPH analysis. *Ocean Engineering*, 53:111–127, 2012. DOI: 10.1016/j.oceaneng.2012.06.027.

[40] D. Le Touzé, A. Marsh, G. Oger, P.-M. M. Guilcher, C. Khaddaj-Mallat, B. Alessandrini and P. Ferrant. SPH simulation of green water and ship flooding scenarios. *Journal of Hydrodynamics, Ser. B*, 22(5):231–236, 2010. DOI: 10.1016/S1001-6058(09)60199-2.

[41] G. N. Lee, K. H. Jung, C. Y. Jun, I. R. Park, Š. Malenica and Y. S. Chung. Experimental and numerica study of the behaviour and flow kinematics of the formation of green water on a rectangular structure. *Brodogradnja*, 67(3):133–145, 2017. DOI: 10.21278/brod67308.

[42] H. Lu, C. Yang and R. Löhner. Numerical studies of green water impact on fixed and moving bodies. *International Journal of Offshore and Polar Engineering*, 22(2):123–132, 2012.

[43] K. B. Nielsen and S. Mayer. Numerical prediction of green water incidents. *Ocean Engineering*, 31(3-4):363–399, 2004. DOI: 10.1016/j.oceaneng.2003.06.001.

[44] A. Östman, C. Pakozdi, L. Sileo, C.-T. Stansberg and D. F. de Carvalho e Silva. A Fully Nonlinear RANS-VOF Numerical Wavetank Applied in the Analysis of Green Water on FPSO in Waves. In *Volume 8B: Ocean Engineering*, volume 8B. ASME, 2014. DOI: 10.1115/OMAE2014-23927.

[45] C. Pakozdi, C.-T. Stansberg, P. Skjetne and W. Yang. Using a Simplified Smoothed Particle Hydrodynamics Model to Simulate Green Water on the Deck. In *Volume 1: Offshore Technology*, volume 1, page 645. ASME, 2012. DOI: 10.1115/OMAE2012-83847.

[46] H. Qin, W. Tang, Z. Hu and J. Guo. Structural response of deck structures on the green water event caused by freak waves. *Journal of Fluids and Structures*, 68:322–338, 2017. DOI: 10.1016/j.jfluidstructs.2016.11.009.

[47] Y. Ryu, K. A. Chang and R. Mercier. Runup and green water velocities due to breaking wave impinging and overtopping. *Experiments in Fluids*, 43(4):555–567, 2007. DOI: 10.1007/s00348-007-0332-0.

[48] Y. Ryu, K.-A. Chang and R. Mercier. Application of dam-break flow to green water prediction. *Applied Ocean Research*, 29(3):128–136, 2007. DOI: 10.1016/j.apor.2007.10.002.

[49] T. Schønberg and R. C. T. Rainey. A hydrodynamic model of Green Water incidents. *Applied Ocean Research*, 24(5):299–307, 2002. DOI: 10.1016/S0141-1187(03)00004-X.

[50] K. Shibata, S. Koshizuka and K. Tanizawa. Three-dimensional numerical analysis of shipping water onto a moving ship using a particle method. *Journal of Marine Science and Technology*, 14(2):214–227, 2009. DOI: 10.1007/s00773-009-0052-7.

[51] K. Shibata, S. Koshizuka, M. Sakai and K. Tanizawa. Lagrangian simulations of ship-wave interactions in rough seas. *Ocean Engineering*, 42:13–25, 2012. DOI: 10.1016/j.oceaneng.2012.01.016.

[52] D. F. Silva, A. L. Coutinho and P. T. Esperança. Green water loads on FPSOs exposed to beam and quartering seas, part I: Experimental tests. *Ocean Engineering*, 140:419–433, 2017. DOI: 10.1016/j.oceaneng.2017.05.005.

[53] D. F. Silva, P. T. Esperança and A. L. Coutinho. Green water loads on FPSOs exposed to beam and quartering seas, Part II: CFD simulations. *Ocean Engineering*, 140:434–452, 2017. DOI: 10.1016/j.oceaneng.2016.11.008.

[54] Y. K. Song, K. A. Chang, K. Ariyarathne and R. Mercier. Surface velocity and impact pressure of green water flow on a fixed model structure in a large wave basin. *Ocean Engineering*, 104:40–51, 2015. DOI: 10.1016/j.oceaneng.2015.04.085.

[55] S. Sun, W. Du and H. Li. Study on Green Water of Tumblehome Hull Using Dam-Break Flow and RANSE Models. *Polish Maritime Research*, S2(94):172–180, 2017. DOI: 10.1515/pomr-2017-0080.

[56] S. Wang, X. Wang, W. L. Woo and T. H. Seow. Study on green water prediction for FPSOs by a practical numerical approach. *Ocean Engineering*, 143:88–96, 2017. DOI: 10.1016/j.oceaneng.2017.07.052.

[57] M. Warmowska. Problem of water flow on deck. *Archives of Civil and Mechanical Engineering*, 7(4):5–14, 2007. DOI: 10.1016/S1644-9665(12)60221-0.

[58] L. Xiao, L. Tao, J. Yang and X. Li. An experimental investigation on wave runup along the broadside of a single point moored FPSO exposed to oblique waves. *Ocean Engineering*, 88:81–90, 2014. DOI: 10.1016/j.oceaneng.2014.06.009.

[59] J. Yamasaki, H. Miyata and A. Kanai. Finite-difference simulation of green water impact on fixed and moving bodies. *Journal of Marine Science and Technology*, 10(1):1–10, 2005. DOI: 10.1007/s00773-005-0194-1.

[60] O. Yilmaz, A. Incecik and J. C. Han. Simulation of green water flow on deck using non-linear dam breaking theory. *Ocean Engineering*, 30(5):601–610, 2003. DOI: 10.1016/S0029-8018(02)00042-2.

[61] R. Zhu, Z. Lin and G. Miao. Numerical simulation for green water occurrence. *Journal of Hydrodynamics, Ser. B*, 18(3):498–504, 2006. DOI: 10.1016/S1001-6058(06)60101-7.

[62] R. C. Zhu, G. P. Miao and Z. W. Lin. Numerical Research on FPSOs With Green Water Occurrence. *Journal of Ship Research*, 53(1):7–18, 2009.

[63] D. T. Cox and J. A. Ortega. Laboratory observations of green water overtopping a fixed deck. *Ocean Engineering*, 29(14):1827–1840, 2002. DOI: 10.1016/S0029-8018(02)00011-2.

[64] K Ariyarathne, K. A. Chang, R Mercier and Asme. Measurement of Green Water on a 3D Structure. *Omae 2009, Vol 6*, (2005):531–538, 2009. DOI: 10.1115/OMAE2009-79868.

[65] K Ariyarathne, K.-A. Chang and R Mercier. Green water impact pressure on a three-dimensional model structure. *Experiments in Fluids*, 53(6):1879–1894, 2012. DOI: 10.1007/s00348-012-1399-9.

[66] Y. Ryu and K.-A. Chang. Breaking wave impinging and greenwater on a two-dimensional offshore structure. In *Proceedings of 15th International Offshore and Polar Engineering Conference*, pages 660–665, 2005.

[67] H.-H. H. Lee, H.-J. J. Lim and S. H. Rhee. Experimental investigation of green water on deck for a CFD validation database. *Ocean Engineering*, 42:47–60, 2012. DOI: 10.1016/j.oceaneng.2011.12.026.

[68] B. Guo, L. F. Xiao and J. M. Yang. Analysis on Motions and Green Water of FPSOs in Shallow Water With Non-Collinear Environments. *29th International Conference on Ocean, Offshore and Arctic Engineering: Volume 4*:419–427, 2010. DOI: 10.1115/OMAE2010-20704.

[69] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *Journal of Computational Physics*, 225(1):300–321, 2007. DOI: 10.1016/j.jcp.2006.11.034.

[70] M. Mirzadeh, A. Guittet, C. Burstedde and F. Gibou. Parallel level-set methods on adaptive tree-based grids. *Journal of Computational Physics*, 322:345–364, 2016. DOI: 10.1016/j.jcp.2016.06.017.

[71] J.-S. Chen, M. Hillman and S.-W. Chi. Meshfree Methods: Progress Made after 20 Years. *Journal of Engineering Mechanics*, 143(4):04017001, 2017. DOI: 10.1061/(ASCE)EM.1943-7889.0001176.

[72] T.-p. Fries and H.-g. Matthias. Classification and overview of meshfree methods. Technical report, TU Braunschweig, 2004, page 64.

[73] S. Li and W. K. W. Liu. Meshfree and particle methods and their applications. *Applied Mechanics Reviews*, 55(1):1, 2002. DOI: 10.1115/1.1431547.

[74] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013, 1977. DOI: 10.1086/112164. arXiv: 9809069v1 [arXiv:gr-qc].

[75] R. a. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics-theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977. DOI: 10.1093/mnras/181.3.375. arXiv: arXiv:1011.1669v3.

[76] J. J. Monaghan. Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1):543–574, 1992. DOI: 10.1146/annurev.aa.30.090192.002551. arXiv: arXiv:1011.1669v3.

[77] J. Monaghan. Simulating Free Surface Flows with SPH. *Journal of Computational Physics*, 110(2):399–406, 1994. DOI: 10.1006/jcph.1994.1034.

[78] T. Tamai and S. Koshizuka. Least squares moving particle semi-implicit method. *Computational Particle Mechanics*, 1(3):277–305, 2014. DOI: 10.1007/s40571-014-0027-2.

[79] P. S. Jensen. Finite difference techniques for variable grids. *Computers and Structures*, 2(1-2):17–29, 1972. DOI: 10.1016/0045-7949(72)90020-X.

[80] J. Benito, F. Ureña and L. Gavete. Influence of several factors in the generalized finite difference method. *Applied Mathematical Modelling*, 25(12):1039–1053, 2001. DOI: 10.1016/S0307-904X(01)00029-4.

[81] S. Tiwari, J. J. Kuhnert and S. Tiwari. Finite Pointset Method Based on the Projection Method for Simulations of the Incompressible Navier-Stokes Equations BT - Meshfree Methods for Partial Differential Equations. In M. Griebel and M. A. Schweitzer, editors, *Meshfree Methods for Partial Differential Equations*, Berichte des Fraunhofer-Instituts für Techno- und Wirtschaftsmathematik (ITWM Report) - 30, pages 373–387. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. DOI: 10.1007/978-3-642-56103-0_26.

[82] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, 22(104):745–762, 1968. DOI: 10.1090/S0025-5718-1968-0242392-2.

[83] J. Kuhnert and S. Tiwari. Grid Free Method For Solving The Poisson Equation. Technical report, Fraunhofer (ITWM), 2001.

[84] S. Tiwari and J. Kuhnert. Modeling of two-phase flows with surface tension by finite pointset method (FPM). *Journal of Computational and Applied Mathematics*, 203(2 SPEC. ISS.):376–386, 2007. DOI: 10.1016/j.cam.2006.04.048.

[85] M. W. Evans and F. H. Harlow. The Particle-in-Cell Method for Hydrodynamic Calculations, 1957.

[86] J. U. Brackbill and H. M. Ruppel. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986. DOI: 10.1016/0021-9991(86)90211-1.

[87] O. J. Dellar and B. L. Jones. Discretising the linearised Navier-Stokes equations: A systems theory approach. In *2016 UKACC 11th International Conference on Control (CONTROL)*, pages 1–6, 2016. DOI: 10.1109/CONTROL.2016.7737634.

[88] J. Bašić, N. Degiuli and A. Werner. Simulation of Water Entry and Exit of a Circular Cylinder Using the ISPH Method. *Transactions of FAMENA*, 38(1):45–62, 2014.

[89] J. Guermond, P. Minev and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47):6011–6045, 2006. DOI: 10.1016/j.cma.2005.10.010.

[90] D. L. Brown, R. Cortez and M. L. Minion. Accurate Projection Methods for the Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 168(2):464–499, 2001. DOI: 10.1006/jcph.2001.6715.

[91] A. Vreman. The projection method for the incompressible Navier-Stokes equations: The pressure near a no-slip wall. *Journal of Computational Physics*, 263:353–374, 2014. DOI: 10.1016/j.jcp.2014.01.035.

[92] R. Témam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (I). *Archive for Rational Mechanics and Analysis*, 32(2):135–153, 1969. DOI: 10.1007/BF00247678.

[93] S. J. Cummins and M. Rudman. An SPH Projection Method. *Journal of Computational Physics*, 152(2):584–607, 1999. DOI: 10.1006/jcph.1999.6246.

[94] R. Rannacher. On Chorin's projection method for the incompressible Navier-Stokes equations. *The Navier-Stokes Equations II, Theory and Numerical Methods*:167–183, 1992. DOI: 10.1007/BFb0090341.

[95] S. M. Hosseini and J. J. Feng. Pressure boundary conditions for computing incompressible flows with SPH. *Journal of Computational Physics*, 230(19):7473–7487, 2011. DOI: 10.1016/j.jcp.2011.06.013.

[96] D. Shirokoff and R. Rosales. An efficient method for the incompressible Navier-Stokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary. *Journal of Computational Physics*, 230(23):8619–8646, 2011. DOI: 10.1016/j.jcp.2011.08.011.

[97] H. Johnston and J. G. Liu. Accurate, stable and efficient Navier-Stokes solvers based on explicit treatment of the pressure term. *Journal of Computational Physics*, 199(1):221–259, 2004. DOI: 10.1016/j.jcp.2004.02.009.

[98] S. A. Orszag, M Israeli and M. O. Deville. Boundary Conditions for Incompressible Flows. *Journal of Scientific Computing*, 1(1):75–111, 1986. DOI: 10.1007/BF01061454.

[99] W. D. Henshaw and N. A. Petersson. A Split-Step Scheme for the Incompressible Navier-Stokes Equations. In *Workshop on Numerical Simulations of Incompressible Flows*, pages 1–16, Half Moon Bay, CA, 2001.

[100] B. Van Haarlem, B. J. Boersma and F. M. Nieuwstadt. Direct numerical simulation of particle deposition onto a free-slip and no-slip surface. *Physics of Fluids*, 10(10):2608, 1998. DOI: 10.1063/1.869774.

[101] É. Lamballais and J. H. Silvestrini. Direct numerical simulation of interactions between a mixing layer and a wake around a cylinder. *Journal of Turbulence*, 2002. DOI: 10.1088/1468-5248/3/1/028.

[102] P. M. Gresho and R. L. Sani. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 7(10):1111–1145, 1987. DOI: 10.1002/fld.1650071008.

[103] R. L. Sani, J. Shen, O. Pironneau and P. M. Gresho. Pressure boundary condition for the time-dependent incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 50(6):673–682, 2006. DOI: 10.1002/fld.1062.

[104] J. Liu. Open and traction boundary conditions for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 228(19):7250–7267, 2009. DOI: 10.1016/j.jcp.2009.06.021.

[105] Z Sheng, M Thiriet and F Hecht. A high-order scheme for the incompressible Navier-Stokes equations with open boundary condition. *International Journal for Numerical Methods in Fluids*, 73(1):58–73, 2013. DOI: 10.1002/fld.3792.

[106] R. Fernandez-Feria and E. Sanmiguel-Rojas. An explicit projection method for solving incompressible flows driven by a pressure difference. *Computers and Fluids*, 2004. DOI: 10.1016/S0045-7930(03)00062-8.

[107] H Chanson. Analytical solutions of laminar and turbulent dam break wave. *River Flow*, (1955):465–474, 2006.

[108] I.-R. Park, K.-S. Kim, J. Kim and S.-H. Van. Numerical investigation of the effects of turbulence intensity on dam-break flows. *Ocean Engineering*, 42:176–187, 2012. DOI: 10.1016/j.oceaneng.2012.01.005.

[109] P. Arnold. Validation of FLOW-3D against a 3D Dam Breaking Problem. Technical report, Minerva Dynamics, High Street Bath, UK, 2014, page 15.

[110] I. M. Jánosi, D. Jan, K. G. Szabó and T. Tél. Turbulent drag reduction in dam-break flows. *Experiments in Fluids*, 37(2):219–229, 2004. DOI: 10.1007/s00348-004-0804-4.

[111] J. Bašić, N. Degiuli and D. Ban. Assessment of d'Alembert's paradox in panel methods by tangency correction. *Engineering Analysis with Boundary Elements*, 85:136–141, 2017. DOI: 10.1016/j.enganabound.2017.10.002.

[112] B. J. West, K. A. Brueckner, R. S. Janda, D. M. Milder and R. L. Milton. A new numerical method for surface hydrodynamics. *Journal of Geophysical Research: Oceans*, 92(C11):11803–11824, 1987. DOI: 10.1029/JC092iC11p11803.

[113] B. Le Méhauté. *An Introduction to Hydrodynamics and Water Waves*. 1976.

[114] X.-B. Chen. Hydrodynamics in Offshore and Naval Applications - Part I. In *Proceedings of the 6th International Conference on Hydrodynamics (ICHD '04)*, page 28, Perth, 2004.

[115] G. Ducrozet, F. Bonnefoy, D. Le Touzé and P. Ferrant. HOS-ocean: Open-source solver for nonlinear waves in open ocean based on High-Order Spectral method. *Computer Physics Communications*, 203:245–254, 2016. DOI: 10.1016/J.CPC.2016.02.017.

[116] S. Marrone, A. Di Mascio and D. Le Touzé. Coupling of Smoothed Particle Hydrodynamics with Finite Volume method for free-surface flows. *Journal of Computational Physics*, 310:161–180, 2016. DOI: 10.1016/J.JCP.2015.11.059.

[117] T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. 2011. DOI: 10.1002/9781119994138.

[118] E. Gaburov and K. Nitadori. Astrophysical weighted particle magnetohydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 414(1):129–154, 2011. DOI: 10.1111/j.1365-2966.2011.18313.x. arXiv: 1006.4159 [astro-ph.IM].

[119] P. F. Hopkins. A New Class of Accurate, Mesh-Free Hydrodynamic Simulation Methods. *Monthly Notices of the Royal Astronomical Society*, 450(1):53–110, 2015. DOI: 10.1093/mnras/stv195. arXiv: 1409.7395.

[120] J. P. Vila. On Particle Weighted Methods and Smooth Particle Hydrodynamics. *Mathematical Models and Methods in Applied Sciences*, 09(02):161–209, 1999. DOI: 10.1142/S0218202599000117.

[121] A. Katz and A. Jameson. A Meshless Volume Scheme. In *19th AIAA Computational Fluid Dynamics*, Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2009. DOI: doi:10.2514/6.2009-3534.

[122] N. Lanson and J. P. Vila. Renormalized Meshfree Schemes I: Consistency, Stability, and Hybrid Methods for Conservation Laws. *SIAM Journal on Numerical Analysis*, 46(4):1912–1934, 2008. DOI: 10.1137/S0036142903427718.

[123] N. Lanson and J. P. Vila. Renormalized Meshfree Schemes II: Convergence for Scalar Conservation Laws. *SIAM Journal on Numerical Analysis*, 46(4):1935–1964, 2008. DOI: 10.1137/S003614290444739X.

[124] N. Ivanova, S. Justham, X. Chen, O. De Marco, C. L. Fryer, E. Gaburov, H. Ge, E. Glebbeek, Z. Han, X. D. Li, G. Lu, T. Marsh, P. Podsiadlowski, A. Potter, N. Soker, R. Taam, T. M. Tauris, E. P. J. Van Den Heuvel and R. F. Webbink. Common envelope evolution: Where we stand and how we can move forward, 2013. DOI: 10.1007/s00159-013-0059-2. arXiv: 1209.4302.

[125] R. Fatehi and M. Manzari. Error estimation in smoothed particle hydrodynamics and a new scheme for second derivatives. *Computers & Mathematics with Applications*, 61(2):482–498, 2011. DOI: 10.1016/j.camwa.2010.11.028.

[126] T. Tamai, K. Murotani and S. Koshizuka. On the consistency and convergence of particle-based meshfree discretization schemes for the Laplace operator. *Computers & Fluids*, 142:79–85, 2017. DOI: 10.1016/j.compfluid.2016.02.012.

[127] M. Hirschler, P. Kunz, M. Huber, F. Hahn and U. Nieken. Open boundary conditions for ISPH and their application to micro-flow. *Journal of Computational Physics*, 307:614–633, 2016. DOI: 10.1016/j.jcp.2015.12.024.

[128] A. Skillen, S. Lind, P. K. Stansby and B. D. Rogers. Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised Fickian smoothing applied to body-water slam and efficient wave-body interaction. *Computer Methods in Applied Mechanics and Engineering*, 265:163–173, 2013. DOI: 10.1016/j.cma.2013.05.017.

[129] B. de Foy and W. Dawes. The design of improved smoothing operators for finite volume flow solvers on unstructured meshes. *International Journal for Numerical Methods in Fluids*, 36(8):903–923, 2001. DOI: 10.1002/fld.156.

[130] F. Juretic and D. Gosman. Error Analysis of the Finite-Volume Method with Respect to Mesh Type. *Numerical Heat Transfer Part B - Fundamentals*, 57:414–439, 2010. DOI: 10.1080/10407791003685155.

[131] H Owen and R Codina. A third-order velocity correction scheme obtained at the discrete level. *International Journal for Numerical Methods in Fluids*, 69(1):57–72, 2012. DOI: 10.1002/fld.2535.

[132] L. Brookshaw. A Method of Calculating Radiative Heat Diffusion in Particle Simulations. *Publications of the Astronomical Society of Australia*, 6(2):207–210, 1985. DOI: 10.1017/S1323358000018117.

[133] J. Monaghan and R. Gingold. Shock simulation by the particle method SPH. *Journal of Computational Physics*, 52(2):374–389, 1983. DOI: 10.1016/0021-9991(83)90036-0.

[134] Q. Ma, Y. Zhou and S. Yan. A review on approaches to solving Poisson's equation in projection-based meshless methods for modelling strongly nonlinear water waves. *Journal of Ocean Engineering and Marine Energy*, 2(3):279–299, 2016. DOI: 10.1007/s40722-016-0063-5.

[135] G. Chaussonnet, S. Braun, L. Wieth, R. Koch and H.-J. Bauer. Influence of particle disorder and smoothing length on SPH operator accuracy. In *10th International SPHERIC Workshop*, page 8, Parma, Italy, 2015.

[136] S. Koshizuka, A. Nobe and Y. Oka. Numerical analysis of breaking waves using the moving particle semi-implicit method. *International Journal for Numerical Methods in Fluids*, 26(7):751–769, 1998. DOI: 10.1002/(SICI)1097-0363(19980415)26:7<751::AID-FLD671>3.0.CO;2-C.

[137] H. Isshiki. Discrete differential operators on irregular nodes (DDIN). *International Journal for Numerical Methods in Engineering*, 88(12):1323–1343, 2011. DOI: 10.1002/nme.3225.

[138] S. Zhang, K. Fukuda, K. Morita and N. Shirakawa. Simulation of the incompressible flows with the MPS method. English. In *13th International Conference on Nuclear Engineering*, page 8, Beijing, China, 2005.

[139] S. Zhang, K. Morita, K. Fukuda and N. Shirakawa. An improved MPS method for numerical simulations of convective heat transfer problems. *International Journal for Numerical Methods in Fluids*, 51(1):31–47, 2006. DOI: 10.1002/fld.1106.

[140] K. Ng, Y. Hwang and T. Sheu. On the accuracy assessment of Laplacian models in MPS. *Computer Physics Communications*, 185(10):2412–2426, 2014. DOI: 10.1016/j.cpc.2014.05.012.

[141] H. Ikari, A. Khayyer and H. Gotoh. Corrected higher order Laplacian for enhancement of pressure calculation by projection-based particle methods with applications in ocean engineering. *Journal of Ocean Engineering and Marine Energy*, 1(4):361–376, 2015. DOI: 10.1007/s40722-015-0026-2.

[142] C Huang, J. M. Lei, M. B. Liu and X. Y. Peng. An improved KGF-SPH with a novel discrete scheme of Laplacian operator for viscous incompressible fluid flows. *International Journal for Numerical Methods in Fluids*, 81(6):377–396, 2016. DOI: 10.1002/fld.4191.

[143] J.-M. Lei and X.-Y. Peng. Improved kernel gradient free-smoothed particle hydrodynamics and its applications to heat transfer problems. *Chinese Physics B*, 25(2):020202, 2016. DOI: 10.1088/1674-1056/25/2/020202.

[144] Y. Lu, A.-k. Hu, Y.-c. Liu and C.-s. Han. A meshless method based on moving least squares for the simulation of free surface flows. *Journal of Zhejiang University SCIENCE A*, 17(2):130–143, 2016. DOI: 10.1631/jzus.A1500053.

[145] B. Schrader, S. Reboux and I. F. Sbalzarini. Discretization correction of general integral PSE Operators for particle methods. *Journal of Computational Physics*, 229(11):4159–4182, 2010. DOI: 10.1016/j.jcp.2010.02.004.

[146] J. D. Eldredge, A. Leonard and T. Colonius. A General Deterministic Treatment of Derivatives in Particle Methods. *Journal of Computational Physics*, 180(2):686–709, 2002. DOI: 10.1006/jcph.2002.7112.

[147] V. Bayona, M. Moscoso, M. Carretero and M. Kindelan. RBF-FD formulas and convergence properties. *Journal of Computational Physics*, 229(22):8281–8295, 2010. DOI: 10.1016/j.jcp.2010.07.008.

[148] O. Davydov and D. T. Oanh. On the optimal shape parameter for Gaussian radial basis function finite difference approximation of the Poisson equation. *Computers & Mathematics with Applications*, 62(5):2143–2161, 2011. DOI: 10.1016/j.camwa.2011.06.037.

[149] S. Khorasanizade and J. M. Sousa. An innovative open boundary treatment for incompressible SPH. *International Journal for Numerical Methods in Fluids*, 2016. DOI: 10.1002/fld.4074.

[150] M. Ferrand, D. R. Laurence, B. D. Rogers, D. Violeau and C. Kassiotis. Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *International Journal for Numerical Methods in Fluids*, 71(4):446–472, 2013. DOI: 10.1002/fld.3666.

[151] A. Mayrhofer, B. D. Rogers, D. Violeau and M. Ferrand. Investigation of wall bounded flows using SPH and the unified semi-analytical wall boundary conditions. *Computer Physics Communications*, 184(11):2515–2527, 2013. DOI: 10.1016/j.cpc.2013.07.004. arXiv: 1304.3692.

[152] A. Leroy, D. Violeau, M. Ferrand and C. Kassiotis. Unified semi-analytical wall boundary conditions applied to 2-D incompressible SPH. *Journal of Computational Physics*, 261:106–129, 2014. DOI: 10.1016/j.jcp.2013.12.035.

[153] S. Marrone, A. Colagrossi, D. Le Touzé and G. Graziani. Fast free-surface detection and level-set function definition in SPH solvers. *Journal of Computational Physics*, 229(10):3652–3663, 2010. DOI: 10.1016/j.jcp.2010.01.019.

[154] A. Barecasco, H. Terissa and C. F. Naa. Simple free-surface detection in two and three-dimensional SPH solver. *Recent Development in Computational Science*:1–10, 2013. arXiv: 1309.4290.

[155] H. Johnston and J.-G. Liu. Finite Difference Schemes for Incompressible Flow Based on Local Pressure Boundary Conditions. *Journal of Computational Physics*, 180(1):120–154, 2002. DOI: 10.1006/jcph.2002.7079.

[156] B. Seibold. Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods. *Numerical Linear Algebra with Applications*, 17:433–451, 2010. DOI: 10.1002/nla.710. arXiv: 0905.3005.

[157] Y. Saad. Iterative Methods for Sparse Linear Systems, Second Edition. *Philadelphia, Pennsylvania: SIAM*, 2003. DOI: 10.2277/0898715342.

[158] L.-Y. Cheng, R. A. Amaro Junior and C. A. Bellezi. Wave Impact Loads by MPS Method with an Improved Pressure Source Term. In *Annual Meeting of JASNAOE 2018*, page 5, Chiba, Japan, 2018.

[159] L. Demkowicz, A. Karafiat and T. Liszka. On some convergence results for FDM with irregular mesh. *Computer Methods in Applied Mechanics and Engineering*, 42(3):343–355, 1984. DOI: 10.1016/0045-7825(84)90013-6.

[160] T. J. Liszka, C. A. Duarte and W. W. Tworzydlo. hp-Meshless cloud method. *Computer Methods in Applied Mechanics and Engineering*, 1996. DOI: 10.1016/S0045-7825(96)01086-9. arXiv: arXiv:1403.8129v1.

[161] R. S. Varga. *Matrix Iterative Analysis*. Springer, 1999, page 358. DOI: 10.1007/978-3-642-05156-2.

[162] W. D. Henshaw, H. O. Kreiss and L. G. Reyna. A fourth-order-accurate difference approximation for the incompressible Navier-Stokes equations. *Computers and Fluids*, 23(4):575–593, 1994. DOI: 10.1016/0045-7930(94)90053-1.

[163] W. D. Henshaw. A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids, 1994. DOI: 10.1006/jcph.1994.1114.

[164] W. D. Henshaw. Cgins Reference Manual: An Overture Solver for the Incompressible Navier-Stokes Equations on Composite Overlapping Grids. Technical report, Lawrence Livermore National Laboratory, Livermore, CA, 2012, page 80.

[165] A. Vidal Urbina. *Meshless Direct Numerical Simulation of Turbulent Incompressible Flows*. PhD thesis, University of Central Florida, 2015.

[166] M. Muller, B. Heidelberger, M. Hennix and J. Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007. DOI: 10.1016/j.jvcir.2007.01.005.

[167] M. Macklin and M. Müller. Position based fluids. *ACM Transactions on Graphics*, 32(4):1, 2013. DOI: 10.1145/2461912.2461984.

[168] P. Suchde and J. Kuhnert. Point Cloud Movement For Fully Lagrangian Meshfree Methods. *arXiv*, 1704.00618:12, 2017. arXiv: 1704.00618.

[169] R. Xu, P. Stansby and D. Laurence. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703–6725, 2009. DOI: 10.1016/j.jcp.2009.05.032.

[170] S. Lind, R. Xu, P. Stansby and B. Rogers. Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. *Journal of Computational Physics*, 231(4):1499–1523, 2012. DOI: 10.1016/j.jcp.2011.10.027.

[171] M. A. Kopera and F. X. Giraldo. Mass conservation of the unified continuous and discontinuous element-based Galerkin methods on dynamically adaptive grids with application to atmospheric simulations. *Journal of Computational Physics*, 297:90–103, 2015. DOI: 10.1016/J.JCP.2015.05.010.

[172] A. Tafuni, J. Domínguez, R. Vacondio and A. Crespo. A versatile algorithm for the treatment of open boundary conditions in Smoothed particle hydrodynamics GPU models. *Computer Methods in Applied Mechanics and Engineering*, 342:604–624, 2018. DOI: 10.1016/J.CMA.2018.08.004.

[173] M. Ferrand, A. Joly, C. Kassiotis, D. Violeau, A. Leroy, F. X. Morel and B. D. Rogers. Unsteady open boundaries for SPH using semi-analytical conditions and Riemann solver in 2D. *Computer Physics Communications*, 2017. DOI: 10.1016/j.cpc.2016.09.009.

[174] R Perić and M Abdel-Maksoud. Reliable damping of free-surface waves in numerical simulations. *Ship Technology Research*, 63(1):1–13, 2016. DOI: 10.1080/09377255.2015.1119921.

[175] D. R. Fuhrman, P. A. Madsen and H. B. Bingham. Numerical simulation of lowest-order short-crested wave instabilities. *Journal of Fluid Mechanics*, 563:415–441, 2006. DOI: 10.1017/S0022112006001236.

[176] S. Mayer, A. Garapon and L. S. Sørensen. A Fractional Step Method for Unsteady Free-surface Fow with Applications to Non-linear Wave Dynamics. *International Journal for Numerical Methods in Fluids*, 28(2):293–315, 1998. DOI: 10.1002/(SICI)1097-0363(19980815)28:2<293::AID-FLD719>3.0.CO;2-1.

[177] M. Müller, D. Charypar and M. Gross. Particle-Based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2003.

[178] M. A. Miquel, A. Kamath, M. Alagan Chella, R. Archetti and H. Bihs. Analysis of Different Methods for Wave Generation and Absorption in a CFD-Based Numerical Wave Tank, 2018. DOI: 10.3390/jmse6020073.

[179] J. Choi and S. B. Yoon. Numerical simulations using momentum source wave-maker applied to RANS equation model. *Coastal Engineering*, 2009. DOI: 10.1016/j.coastaleng.2009.06.009.

[180] D. Barbieri, V. Cardellini and S. Filippone. Fast Uniform Grid Construction on GPGPUs Using Atomic Operations. In *Proceedings of the International Conference on Parallel Computing (ParCo)*, pages 295–304, 2013. DOI: 10.3233/978-1-61499-381-0-295.

[181] H. A. van der Vorst. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992. DOI: 10.1137/0913035.

[182] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986. DOI: 10.1137/0907058.

[183] R. W. Freund. A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems. *SIAM Journal on Scientific Computing*, 14(2):470–482, 1993. DOI: 10.1137/0914029.

[184] T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto and C. H. Tong. A Quasi-Minimal Residual Variant of the Bi-CGSTAB Algorithm for Nonsymmetric Systems. *SIAM Journal on Scientific Computing*, 15(2):338–347, 1994. DOI: 10.1137/0915023.

[185] H. Anzt, M. Gates, J. Dongarra, M. Kreutzer, G. Wellein and M. Köhler. Preconditioned Krylov solvers on GPUs. *Parallel Computing*, 68:32–44, 2017. DOI: 10.1016/j.parco.2017.05.006.

[186] W. J. Schroeder and K. M. Martin. The visualization toolkit. In *Visualization Handbook*, pages 593–614. 2005. DOI: 10.1016/B978-012387582-2/50032-0.

[187] H. Carter Edwards, C. R. Trott and D. Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74(12):3202–3216, 2014. DOI: 10.1016/j.jpdc.2014.07.003.

[188] R. Franke. A critical comparison of some methods for interpolation of scattered data. Technical report, 1980, page 379.

[189] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396, 1995. DOI: 10.1007/BF02123482.

[190] F. Gibou, R. P. Fedkiw, L.-T. Cheng and M. Kang. A Second-Order-Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *Journal of Computational Physics*, 176(1):205–227, 2002. DOI: 10.1006/jcph.2001.6977.

[191] J. Papac, A. Helgadottir, C. Ratsch and F. Gibou. A level set approach for diffusion and stefan-type problems with robin boundary conditions on quadtree/octree adaptive cartesian grids. *Journal of Computational Physics*, 233(1):241–261, 2013. DOI: 10.1016/j.jcp.2012.08.038.

[192] A. Guittet, M. Lepilliez, S. Tanguy and F. Gibou. Solving elliptic problems with discontinuities on irregular domains - the Voronoi Interface Method. *Journal of Computational Physics*, 298:747–765, 2015. DOI: 10.1016/j.jcp.2015.06.026.

[193] C. H. Marchi, R. Suero and L. K. Araki. The lid-driven square cavity flow: numerical solution with a 1024 x 1024 grid. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 31(3):186–198, 2009. DOI: 10.1590/S1678-58782009000300004.

[194] U. Ghia, K. N. Ghia and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982. DOI: 10.1016/0021-9991(82)90058-4.

[195] B Buchner, T. Bunnik, G Fekken and A. E. P. Veldman. A Numerical Study on Wave Run Up on an FPSO Bow. In *Proceedings of OMAE2001:The 20th International Conference on Offshore Mechanics and Arctic Engineering*, page 7, Rio de Janeiro, 2001.

[196] ISOPE-IHC. Comparative Study on Water Impact Problem for Ship Section & Wedge Drops. Technical report, International Ocean and Polar Engineering Conference International Hydrodynamics Committee, Rhodes, Greece, 2016, page 37.

[197] M. Jalalisendi, S. Zhao and M. Porfiri. Shallow water entry: modeling and experiments. *Journal of Engineering Mathematics*, 104(1):131–156, 2017. DOI: 10.1007/s10665-016-9877-3.

[198] C. Monroy, S. Seng, L. Diebold, A. Benhamou and S. Malenica. A Comparative Study of the Generalized Wagner Model and a Free-Surface RANS Solver for Water Entry Problems. *International Journal of Offshore and Polar Engineering*, 27(2):135–143, 2017. DOI: 10.17736/ijope.2017.jc690.

[199] K. Kleefsman, G. Fekken, A. Veldman, B. Iwanowski and B. Buchner. A Volume-of-Fluid based simulation method for wave impact problems. *Journal of Computational Physics*, 206(1):363–393, 2005. DOI: 10.1016/j.jcp.2004.12.007.

[200] C. Hu and M. Kashiwagi. A CIP-based method for numerical simulations of violent free-surface flows. *Journal of Marine Science and Technology*, 9(4):143–157, 2004. DOI: 10.1007/s00773-004-0180-z.

[201] M. Arai, L.-Y. Cheng, A. Kumano and T. Miyamoto. A Technique for Stable Numerical Computation of Hydrodynamic Impact Pressure in Sloshing Simulation. *Journal of the Society of Naval Architects of Japan*, 2002(191):299–307, 2002. DOI: 10.2534/jjasnaoe1968.2002.191_299.

[202] C. Mokrani and S. Abadie. Conditions for peak pressure stability in VOF simulations of dam break flow impact. *Journal of Fluids and Structures*, 62:86–103, 2016. DOI: 10.1016/j.jfluidstructs.2015.12.007.

[203] H. Ozmen-Cagatay and S. Kocaman. Dam-break flow in the presence of obstacle: Experiment and CFD simulation. *Engineering Applications of Computational Fluid Mechanics*, 5(4):541–552, 2011. DOI: 10.1080/19942060.2011.11015393.

[204] L. Lobovský, E. Botia-Vera, F. Castellana, J. Mas-Soler and A. Souto-Iglesias. Experimental investigation of dynamic pressure loads during dam break. *Journal of Fluids and Structures*, 48:407–434, 2014. DOI: 10.1016/j.jfluidstructs.2014.03.009. arXiv: arXiv:1308.0115v1.

[205] a. S. Iglesias, E. B. Vera and G Bulian. Repeatability and Two-Dimensionality of Model Scale Sloshing Impacts. In *The Twenty-second International Offshore and Polar Engineering Conference*, page 8, 2012.

[206] T. W. Yung, Z. Ding, H. He and R. E. Sandström. LNG sloshing: Characteristics and scaling laws. *International Journal of Offshore and Polar Engineering*, 19(4):264–270, 2009.

[207] A. C. Khezzar, A Seibi and D Goharzadeh. Water Sloshing in Rectangular Tanks. An Experimental Investigation and Numerical Simulation. *International Journal of Engineering*, 3(2):174–184, 2009.

[208] Z. R. Kishev, C. Hu and M. Kashiwagi. Numerical simulation of violent sloshing by a CIP-based method. *Journal of Marine Science and Technology*, 11(2):111–122, 2006. DOI: 10.1007/s00773-006-0216-7.

[209] T. Bunnik and R. Huijsmans. Large-scale LNG sloshing model tests. In *International Journal of Offshore and Polar Engineering*, pages 1893–1899, 2009.

[210] J. V. Hernández-Fontes, M. A. Vitola, M. C. Silva, P. d. T. T. Esperança and S. H. Sphaier. On the Generation of Isolated Green Water Events Using Wet Dam-Break. *Journal of Offshore Mechanics and Arctic Engineering*, 140(5):051101, 2018. DOI: 10.1115/1.4040050.

[211] P. K. Stansby, A Chegini and T. C. D. Barnes. The initial stages of dam-break flow. *Journal of Fluid Mechanics*, 374:407–424, 1988.

[212] J. V. Hernández-Fontes, M. A. Vitola, P. d. T. T. Esperança and S. H. Sphaier. Assessing shipping water vertical loads on a fixed structure by convolution model and wet dam-break tests. *Applied Ocean Research*, 82:63–73, 2019. DOI: 10.1016/J.APOR.2018.10.022.

[213] H. J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev and B. Uekermann. preCICE – A fully parallel library for multi-physics surface coupling. *Computers and Fluids*, 141:250–258, 2016. DOI: 10.1016/j.compfluid.2016.04.003. arXiv: 0402594v3 [arXiv:cond-mat].

[214] D. Gaston, C. Newman, G. Hansen and D. Lebrun-Grandié. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768–1778, 2009. DOI: 10.1016/j.nucengdes.2009.05.021. arXiv: arXiv:1011.1669v3.

[215] Q. Xiong, B. Li and J. Xu. GPU-accelerated adaptive particle splitting and merging in SPH. *Computer Physics Communications*, 184(7):1701–1707, 2013. DOI: 10.1016/j.cpc.2013.02.021.

[216] W. Hong, D. H. House and J. Keyser. Adaptive particles for incompressible fluid simulation. In *Visual Computer*, volume 24 of number 7-9, pages 535–543, 2008. DOI: 10.1007/s00371-008-0234-z.

[217] A. Di Mascio, M. Antuono, A. Colagrossi and S. Marrone. Smoothed particle hydrodynamics method from a large eddy simulation perspective. *Physics of Fluids*, 29(3), 2017. DOI: 10.1063/1.4978274.

[218] P Sagaut and Y.-T. Lee. Large Eddy Simulation for Incompressible Flows: An Introduction. Scientific Computation Series. *Applied Mechanics Reviews*, 55(6):B115, 2002. DOI: 10.1115/1.1508154.

[219] C. Yang, H. Zhang and H. Yao. Numerical study of MPS method with large eddy simulation for fluid solid coupling problem. *Journal of Physics: Conference Series*, 814(1):12004, 2017.

[220] M. B. Liu, G. R. Liu and K. Y. Lam. Constructing smoothing functions in smoothed particle hydrodynamics with applications. *Journal of Computational and Applied Mathematics*, 2003. DOI: 10.1016/S0377-0427(02)00869-5.

[221] G. R. Johnson, R. A. Stryk and S. R. Beissel. SPH for high velocity impact computations. *Computer Methods in Applied Mechanics and Engineering*, 1996. DOI: 10.1016/S0045-7825(96)01089-4.

[222] A. Ghai, C. Lu and X. Jiao. A Comparison of Preconditioned Krylov Subspace Methods for Large-Scale Nonsymmetric Linear Systems:38, 2016. arXiv: 1607.00351.

[223] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002. DOI: 10.1006/jcph.2002.7176.

[224] P. Diehl and M. Schweitzer. Efficient Neighbor Search for Particle Methods on GPUs. In *Meshfree Methods for Partial Differential Equations VII*. Volume 100, Lecture Notes in Computational Science and Engineering. Springer, 2014.

[225] R. Hoetzlein. Fast Fixed-Radius Nearest Neighbors: Interactive Million-Particle Fluids. In *GPU Technology Conference*, Santa Clara, CA, 2014.

[226] W. Kusumawinahyu, W. Karjanto and G. Klopman. Linear theory for single and double flap wavemakers. *Journal of the Indonesian Mathematical Society*, 12(1):41–57, 2006. arXiv: 1703.09445.

[227] O. Nwogu. Alternative Form of Boussinesq Equations for Nearshore Wave Propagation. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 1993. DOI: 10.1061/(ASCE)0733-950X(1993)119:6(618).

Appendices

A. Remarks on the Boundary Condition for the Pressure

Enforcing the incompressibility constraint through the use of the PPE has been a main focus in the development of numerical methods for incompressible flow. There has been a long debate among researchers concerning proper boundary conditions for the PPE equation. One of the topics was whether it is appropriate to use the normal or tangential component of the momentum equation on the boundary as a boundary condition, or some other type of boundary condition; see e.g. [102, 103, 163, 155, 98, 89, 90].

Gresho and Sani [102] were the first to propose an equivalence theorem regarding the PPE for the incompressible Navier–Stokes equations, although without proving it. They claimed that if the Navier–Stokes *momentum equation is solved simultaneously with the PPE equation* whose boundary condition is the Neumann boundary condition obtained by applying the *normal component of the momentum equation on the boundary* on which the normal component of velocity is specified as a Dirichlet boundary condition, *then the solution* of velocity and pressure would be *exactly the same* as if the primitive equations (with the usual divergence-free constraint) were solved instead. Unfortunately, their particular PPE formulation does not incorporate explicit boundary conditions that can be used to recover the pressure from the velocity.

Sani *et al.* [103] investigated the claim to actually prove the theorem for some specific conditions. Additionally, like the primitive equations that require no boundary condition for the pressure, the new results establish the same requirement when the PPE equation approach is employed.

Finally, it is worth noting that even though researchers are familiar with the listed facts, it is not uncommon to use the simplest pressure boundary condition $\nabla p \cdot \boldsymbol{n} = 0$, which makes easy to attain a stable scheme. For a laminar boundary layer flow with a relatively high Reynolds number, it can be a good approximation since $\nabla p \cdot \boldsymbol{n} = \mathcal{O}\left(Rn^{-1/2}\right)$ tends to zero as $Rn \to \infty$. However, the flow around some body always experiences deceleration and detachment, where the simplified boundary condition will always yield significant discrepation, as validated for flow around a circular cylinder [99]. This is also the case for violent flows analysed in this thesis.

B. Large Eddy Simulation

There are several common ways of reducing the number of degrees of freedom in the numerical solution [218]:

- calculating the statistical average of the solution directly (RANS), which is used mostly in engineering calculations,
- calculating certain low-frequency modes in time and the average field (URANS, Semi–Deterministic Simulation (SCS), Very Large Eddy Simulation (VLES) and Coherent–Structure Capturing (CSC)),
- projecting the solution on the ad-hoc function basis and retaining only a minimum number of modes, to get a dynamic system with fewer degrees of freedom (Proper–Orthogonal Decomposition (POD)),
- calculating the low-frequency modes in space directly (Large Eddy Simulation (LES)).

LES modelling represents eddies down to a certain scale, and uses a sub-grid to represent smaller eddies that occur below the threshold scale. Unsteady applications investigated with Lagrangian methods often employ LES, e.g. [165, 217, 219]. The sub-grid properties are included through a statistical model [218], and the theoretical scale separation is formalized in the form of a frequency low-pass filter. A LES model is incorporated into the momentum equation of the incompressible NSE by taking turbulence stress into account:

$$\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f} + \frac{1}{\rho}\nabla\boldsymbol{\tau}, \tag{.1}$$

where $\boldsymbol{\tau}$ is the sub-point scale (SPS) turbulence stress tensor, which is analogous to the sub-grid scale (SGS) in grid-based CFD methods. The eddy viscosity assumption is often used to model the SGS/SPS turbulence stress as:

$$\tau_{ij} = 2\rho\left(\nu_t S_{ij} - \frac{1}{3}\delta_{ij}k\right), \tag{.2}$$

where S_{ij} is SPS strain tensor, k is the turbulence kinetic energy, δ_{ij} is the Kronecker–delta function and ν_t is the eddy viscosity, e.g. which can be based on Smogorinsky SGS model. The eddy viscosity is defined as:

$$\nu_t = (\Delta C_s)^2\sqrt{2\boldsymbol{S}:\boldsymbol{S}}, \tag{.3}$$

where C_s is the Smagorinsky constant, and $\boldsymbol{S}:\boldsymbol{S}$ is the double–inner product that in Einstein notation is written as $S_{ij}S_{ij}$.

C. Weighting Functions

Most of the mesh-free methods do not incorporate a topology map. Compact–support methods employ arbitrary shaped domains around each point in the point cloud where the exchange of the information between close points is made. It is assumed that points outside of the domain do not influence the point under consideration. Therefore, the strength of the interaction between points is based on the distance criteria. The group of points that is located in the compact domain may be referred to as the neighbourhood, and the points are neighbours of the point under consideration. The geometrical shape of the compact domain is most often a sphere, and rarely an ellipse or a rectangular cuboid. Non-sphere domains have the advantage to enclose neighbours in a chosen relevant direction from the considered point. This is not as important, because even when choosing neighbours within a spherical domain, neighbours can be picked and even weighted based on criteria other than the distance from the considered point. Like in most other compact–support meshless methods, the present method employs spherical shapes of the compact domain. For a spherical compact domain, the weighting function depends on the distance and compact radius, $W(r,\, h)$, which can be written in the univariate form by using the ratio of the distance and sphere radius, $W(q)$, where $q = r/h$.

Due to the nature of spatial operators in the SPH method, the performance of an SPH model is critically dependent on the choice of the weighting function, sometimes referred to as the smoothing kernel. The shape of the smoothing kernel must satisfy various conditions in order to yield stable and successful simulations. The kernel must be a smooth and monotonically decreasing function that satisfies the Dirac delta function condition as $h \to 0$, and has normalised area below its curve. The SPH derivatives take directly the derivatives of the smoothing kernel into account, which makes the SPH method sensitive to the choice of the smoothing kernel.

On the other hand, the weighting function in the present method assigns weights to each neighbour point in the compact domain, which are used in the WLS fitting procedure described in section §3.2.1. The values of weights do not directly contribute to the evaluation of spatial operators, but the combination of all of the weights in the neighbourhood determines the outcome of the evaluation of spatial operators. The classical LS form is obtained if the weighting function is taken as $W(q) = \text{const}$. It may be regarded as certain that weighting functions benefit from monotonically decreasing shapes, since farther neighbours should have less influence on the point under consideration.

Many researchers have dealt with introducing compactly supported radial functions of small degrees, e.g. [189, 220], and some of those were modified to be used in the SPH method. Some of the most popular and efficient compactly supported radial functions used in meshless methods are listed in table C.1. Note that the absolute scale of the

listed functions is irrelevant for weighting in the LS or Shepard's procedure. The listed weighting functions are graphed in figure C.1. Properties of the functions are given in table C.2, which describes fullness of the functions through reaching 1% of weighting, and reaching 99.9% of the area under the curve. Figure C.2 shows how the bell–shaped weighting functions would be shaped if they are horizontally scaled by the ratio of their area and the area of Wendland's C2 weighting function. The curves are similar, which confirms that Wendland's C2 function is a reasonable choice for the weighting implemented within the solver.

Table C.1.: Compactly supported weighting functions.

#	Name	$W(q)$	Degree	Author
1	Linear	$1-q$	Linear	n/a
2	Quadratic	q^2-2q+1	Quadratic	Johnson *et al.* [221]
3	Spiky	$(1-q)^3$	Cubic	Desbrun [177]
4	Lucy's	$(1-q)^3(3q+1)$	Quartic	Lucy [74]
5	Liu's	$-\frac{1}{4}(1-q)^2(15q^2-8q-4)$	Quartic	Liu *et al.* [220]
6	Wendland's C2	$(1-q)^4(4q+1)$	Quintic	Wendland [189]
7	Poly6	$(1-q^2)^4$	Sextic	Müller [177]
8	Wendland's C4	$(1-q)^6\left(\frac{35}{3}q^2+6q+1\right)$	Octic	Wendland [189]

Table C.2.: Some properties of compactly supported weighting functions.

#	Name	q for $W=0.01$	q for $\int_0^q W\,dx$ reaching 99.9% of the area	Area
1	Linear	0.990	0.968	0.500
2	Quadratic	0.900	0.900	0.333
3	Spiky	0.784	0.822	0.250
4	Lucy's	0.859	0.855	0.400
5	Liu's	0.792	0.780	0.375
6	Wendland's C2	0.778	0.792	0.333
7	Poly6	0.827	0.823	0.406
8	Wendland's C4	0.687	0.710	0.296

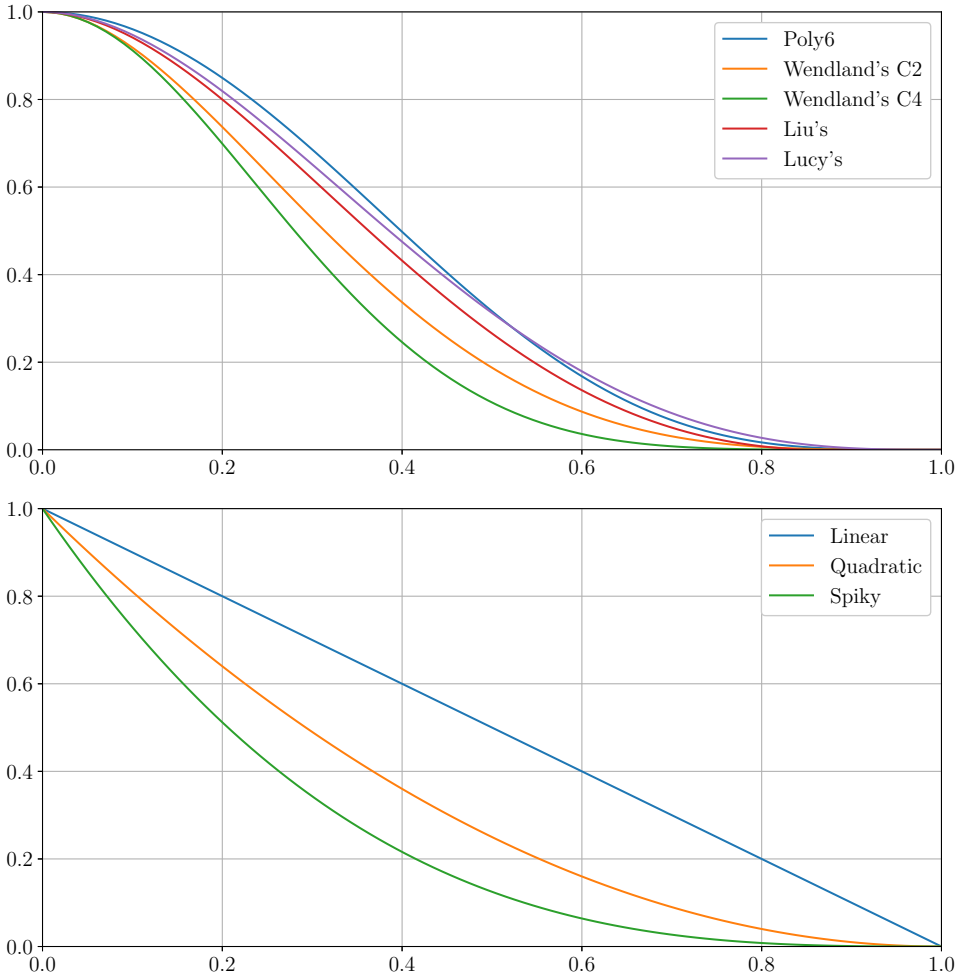

Figure C.1.: Comparison of the weighting function shapes. The top image shows bell–shaped functions, and the bottom image shows other shapes.

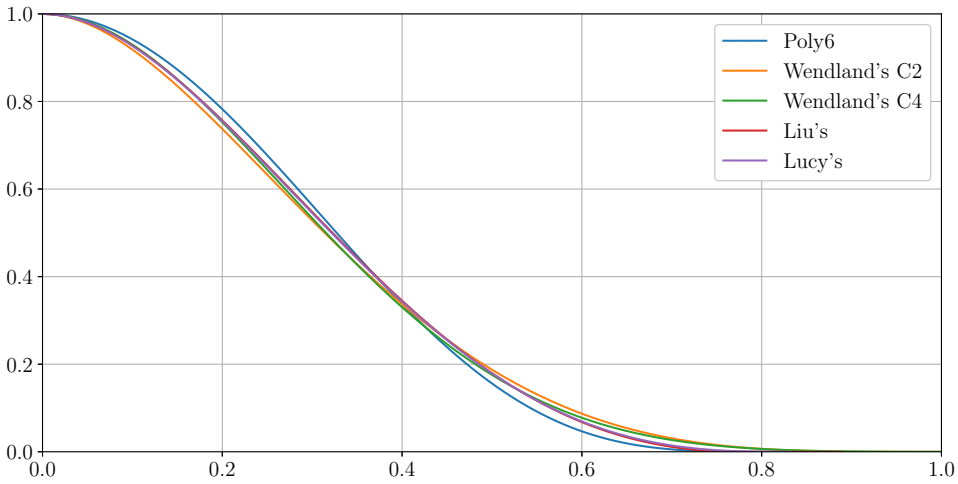

Figure C.2.: Bell–shaped weighting functions horizontally scaled by the ratio of their area and the area of Wendland's C2 weighting function.

D. Krylov Subspace Methods

Krylov subspace (KSP) methods are often used when solving a linear system:

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the large, sparse and non-symmetric $n \times n$ matrix, and $\mathbf{b} \in \mathbb{R}^n$ is the given right-hand-side vector, and $\mathbf{x} \in \mathbb{R}^n$ is the vector of unknowns. Let $\mathbf{x}_0$ be some initial guess to the solution vector, and $\boldsymbol{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ is the initial residual vector. A KSP method incrementally finds approximate solutions within k-th Krylov subspace:

$$\mathcal{K}_k\left(\mathbf{A},\, \mathbf{v}\right) = \text{span}\left\{\mathbf{v},\, \mathbf{A}\mathbf{v},\, \mathbf{A}^2\mathbf{v}, \ldots,\, \mathbf{A}^{k-1}\mathbf{v}\right\}.$$

To construct the basis of the subspace $\mathcal{K}_k\left(\mathbf{A},\, \mathbf{v}\right)$, most commonly used procedures are the (restarted) Arnoldi iteration, and the bi-Lanczos iteration (Lanczos or tridiagonal biorthogonalization) [157]. The most used Arnoldi-based method Generalised Minimal Residual (GMRES) [182] stores basis vectors, and its computational cost is increases with the iteration count. Large memory footprint remains even with the restarting of the solver after some iterations count. Bi-Lanczos iterations need two matrix-vector products, retain three-term recurrence, have fixed performance, but can suffer from breakdown. Rule of the thumb is: if the Arnoldi-based method converges before a restart is needed, it may be the most effective method; if the bi-Lanczos method converges before any breakdown, it is typically more robust than the restarted Arnoldi-based methods [222].

Bi-conjugate gradient stabilised (BiCGStab) method [181] is the most used bi-Lanczos iteration method, since it doesn't require transposing of the coefficient matrix ($\mathbf{A}^{\mathrm{T}}$), and is very robust even without the preconditioning. It was proposed in an attempt to curb the excesses of Conjugate Gradient Squared (CGS) method while retaining the advantages of more rapid convergence and the non-use of transposes. However, it does not guarantee monotonically decreasing residuals. The conventional implementation of the method is shown in algorithm D.1. The bottleneck of the BiCGStab algorithm are two sparse-matrix/vector multiplications per iteration, $\mathbf{v}_i = \mathbf{A}\boldsymbol{K}^{-1}\boldsymbol{p}_i$ and $\boldsymbol{t} = \mathbf{A}\boldsymbol{K}^{-1}\boldsymbol{s}$.

In order to improve the performance and convergence of KSP methods, a preconditioner should be used. A preconditioner is a matrix of transformation, $\mathbf{M}$, whose inverse approximates the inverse of $\mathbf{A}$, i.e. $\mathbf{M}^{-1} \approx \mathbf{A}^{-1}$. Generally, the preconditioned system is easier to solve than the original system. Preconditioning is the most important element when developing efficient iterative solvers for challenging large-scale problems. It has intensively been researched, although an optimal general-purpose preconditioner is unlikely to exist [223]. A preconditioner matrix transforms the initial system to a similar system that should be easier to solve iteratively by improving the spectral properties of the coefficient

Algorithm D.1 Original right–preconditioned BiCGStab.

Require: i_{MAX} // iterations limit
Require: r_{MAX} // relative residual limit
Require: $\mathbf{x}_0$ // e.g. solution from the previous step
1: $\boldsymbol{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
2: $\mathbf{p}_0 = \mathbf{r}_0$ // arbitrary vector such that $\mathbf{p}_0 \cdot \mathbf{r}_0 \neq 0$
3: $\rho_0 = \alpha = \omega = 1$
4: **for** $i = 1, 2, \ldots, i_{MAX}$ **do**
5: $\quad \rho_i = \mathbf{r}_0 \cdot \mathbf{r}_{i-1}$
6: $\quad \beta = (\rho_i / \rho_{i-1})\,(\alpha/\omega)$
7: $\quad \mathbf{p}_i = \mathbf{r}_{i-1} + \beta\,(\mathbf{p}_{i-1} - \omega \mathbf{s}_{i-1})$
8: $\quad \hat{\boldsymbol{p}} = \mathbf{M}^{-1}\mathbf{p}_i$ // precondition
9: $\quad \mathbf{s} = \mathbf{A}\hat{\boldsymbol{p}}$ // matrix-vector product
10: $\quad \alpha = \rho_i / (\mathbf{r}_0 \cdot \mathbf{s})$
11: $\quad \mathbf{q} = \mathbf{r}_{i-1} - \alpha \mathbf{s}_i$
12: $\quad \hat{\boldsymbol{q}} = \mathbf{M}^{-1}\mathbf{q}$ // precondition
13: $\quad \mathbf{y} = \boldsymbol{A}\hat{\boldsymbol{q}}$ // matrix-vector product
14: $\quad \omega = (\mathbf{q} \cdot \mathbf{y}) / (\mathbf{y} \cdot \mathbf{y})$
15: $\quad \mathbf{x}_i = \mathbf{x}_{i-1} + \alpha \hat{\boldsymbol{p}}_i + \omega \hat{\boldsymbol{q}}$
16: $\quad \mathbf{r}_i = \mathbf{q}_i - \omega \mathbf{y}$
17: $\quad$ **if** $\|\mathbf{r}_i\| / \|\mathbf{b}\| < r_{MAX}$ **then**
$\quad\quad$ **return** // $\mathbf{x}_i$ is accurate enough
18: $\quad$ **end if**
19: **end for**

matrix $\mathbf{A}$. The right–preconditioner matrix transforms the linear system into:

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}^{-1}\mathbf{y},$$

which is often used in methods where the residual is being minimised. The transformed equation has the same solution as the initial one. A preconditioner should be easy and fast to construct and benefit to the convergence of the method. In other words, the solving time with the additional cost of constructing and using the preconditioner in an iterative method should be smaller then the solving time for the target tolerance. Anzt *et al.* [185] investigated the effect of Jacobi and ILU preconditioning of GPU-accelerated Krylov solvers (BiCGStab, CGS, QMR, and IDR(s)). They concluded that solvers typically benefit from using the Jacobi preconditioner, i.e. embarrassingly parallel diagonal scaling. Incomplete factorization preconditioning improves the robustness of the solvers, but it is difficult to parallelize, which can become a bottleneck thus increasing the overall solver run-time. For these reasons, the Jacobi preconditioning was chosen to improve the solver convergence and efficiency. The optimised version of the BiCGStab, which merges operations and does not store a matrix, is shown in algorithm D.2. The same optimisation techniques in the shown algorithm were also applied within implementations of CG, TFQMR and QMRCGStab solvers.

Algorithm D.2 Optimised matrix-free implementation of the BiCGStab method with a diagonal preconditioner.

1: $\boldsymbol{m} = \boldsymbol{M}^{-1}(\boldsymbol{A})$ // parallel matrix-free preconditioner
2: **for** $i = 1, 2, \ldots, i_{MAX}$ **do**
3: $\quad \boldsymbol{s} = \boldsymbol{A}\hat{\boldsymbol{p}}$ // parallel matrix-free SPMV product
4: $\quad \alpha = \rho_i / (\boldsymbol{r}_0 \cdot \boldsymbol{s})$
5: $\quad$ **for** $j = 1, 2, \ldots, n$ **in parallel do**
6: $\quad\quad q_j = r_j - \alpha s_j$
7: $\quad\quad \hat{q}_j = m_j q_j$ // precondition
8: $\quad$ **end for**
9: $\quad \boldsymbol{y} = \boldsymbol{A}\hat{\boldsymbol{q}}$ // parallel matrix-free SPMV product
10: $\quad \omega = (\boldsymbol{q} \cdot \boldsymbol{y}) / (\boldsymbol{y} \cdot \boldsymbol{y})$
11: $\quad$ **for** $j = 1, 2, \ldots, n$ **in parallel do**
12: $\quad\quad x_j \mathrel{+}= \alpha \hat{p}_j + \omega \hat{q}_j$
13: $\quad\quad r_j = q_j - \omega y_j$
14: $\quad$ **end for**
15: $\quad$ **if** $\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|} < r_{MAX}$ **then**
$\quad\quad$ **return** // $\boldsymbol{x}_i$ is accurate enough
16: $\quad$ **end if**
17: $\quad \rho_i = \boldsymbol{r}_0 \cdot \boldsymbol{r}$
18: $\quad \beta = (\rho_i / \rho_{i-1})(\alpha / \omega)$
19: $\quad$ **for** $j = 1, 2, \ldots, n$ **in parallel do**
20: $\quad\quad p_j = r_j + \beta (p_j - \omega s_j)$
21: $\quad\quad \hat{p}_j = m_j p_j$ // precondition
22: $\quad$ **end for**
23: **end for**

E. Nearest-Neighbour Search

Compact–support meshless methods usually rely on the distance criteria within the compact domain. In other words, since the point cloud does not incorporate a topology map, a point picks its neighbours within the range of specified compact domain size. A compact domain is most often a sphere, and rarely an ellipse or a rectangular cuboid. Non-sphere domains have the advantage to enclose neighbours in a chosen relevant direction from the considered point. This is not as important, because even when choosing neighbours within a spherical domain, neighbours can be picked and even weighted based on criteria other than the distance from the considered point.

In any case, points in a point cloud need to find nearest neighbours as fast as possible to eliminate the bottleneck of the neighbour traversal, which is a operation present in each step of the solving procedure (and sometimes executed multiple times). To tackle this computational efficiency issue, most researchers of meshless methods employ background grids, which help the nearest-neighbours search. Background grids are usually uniform Cartesian grids, which cells are squares in 2D or cubes in 3D. Non-uniform grids may

contain cells with sizes different in each direction.

Background Grid Insert

In this thesis, atomic functions are used to build up a uniform grid in the form of a set of linked lists [180]. The grid is represented in computer memory by two arrays:

- `head` array (one entry per cell), which stores the index of the first point in list of cell points,
- `next` array (one entry per point), which stores the index of the next point in the list of cell points.

With this configuration, the basic steps to achieve a concurrent insert of a point identified with `point_ID`, in a cell identified with `cell_ID`, are as follows:

```
cell_ID = cell_from_point_coordinate(point_ID)
next[point_ID] = atomic_exchange(&head[cell_ID], point_ID)
```

where **`atomic_exchange`** is the "atomic exchange" function implemented on all modern parallel architectures, which enables concurrent write (of `point_ID`) on a memory location (`&head[cell_ID]`) after retrieving the current value at the location (the result of the function). The retrieved value in this case is used to build linked list, i.e. the former first item in the list becomes the second one, and the new item becomes the first item on insert. The beauty of the method is the simplicity and the performance of elements insertion in the grid [180], without the need of sorting algorithms, e.g. as in [224, 225]. The drawback of the method is that the identifiers of the elements in the same linked list are not sequential in memory. This can be assessed by sorting the data in optimal sequential manner from time to time, e.g. after solving 50 time-steps.

Neighbour Search

After the points are inserted in the background grid, it can be used to find the nearest neighbours of a point under consideration.

```
cells_list = cells_around_point_coordinate(point_ID, compact_radius)
for_each (cell_ID in cells_list):
  neighbour_ID = head[cell_ID]
  while (neighbour_ID >= 0):
    if (distance(point_ID, neighbour_ID) < compact_radius):
      # calculate stuff between the point and its neighbour
    neighbour_ID = next[point_ID]
```

F. Fluid Rendering Shaders

The OpenGL Shading Language (GLSL) is the principal shading language for OpenGL. An OpenGL shader is a user-defined program designed to run on some stage of a graphics processor. Its purpose is to execute one of the programmable stages of the rendering pipeline. The stages are:

- The vertex shader is the programmable shader stage in the rendering pipeline that handles the processing of individual vertices. Each meshless point is a vertex in the shader.
- A geometry shader is the programmable shader stage that governs the processing of primitives. In this case the geometry shader makes a quadrilateral primitive, which centre is the vertex location. The quadrilateral is always facing the camera.
- A fragment shader is the programmable shader stage that will process a fragment generated by the rasterisation into a set of colours and a single depth value. For each sample of the pixels covered by the quadrilateral, a "fragment" is generated. Fragment shaders create the illusion of a sphere on the flat quadrilateral.

The GLSL code snippets of the vertex, geometry and fragment shaders used to visualise meshless points as spheres are listed in the following text, respectively. The code can process millions of meshless points in real-time on modern GPUs, outputting smooth visuals with more than 30 frames per second (FPS).

```
#version 330 core
#extension GL_EXT_gpu_shader4 :  enable
// input:
layout(std140) uniform colormap { vec4 colormap_values[16]; }; // colormap
LUT
uniform vec2 range; // {min, 1/(max-min)}
uniform vec4 clip_plane_near;
uniform vec4 clip_plane_far;
layout(location = 0) in vec4 pos;
// output:
out vec3 sphere_color;
out float gl_ClipDistance[2];
// kernel:
void main() {
  int idx = clamp(int((pos.w - range.x) * range.y * 16), 0, 15);
  sphere_color = colormap_values[idx].xyz;
  vec4 pos4 = vec4(pos.xyz, 1.0);
  gl_Position = pos4;
  gl_ClipDistance[0] = dot(clip_plane_near, pos4);
  gl_ClipDistance[1] = dot(clip_plane_far, pos4);
}
```

```
#version 330 core
#extension GL_EXT_geometry_shader4 :  enable
// input:
layout(points) in;
uniform mat4 mv_matrix;
uniform mat4 p_matrix;
uniform float sphere_radius;
in vec3 sphere_color[];
// output:
layout(triangle_strip, max_vertices=4) out;
flat out vec3 color;
smooth out vec2 texcoord;
flat out vec4 eye_position;
// kernel:
void main() {
  eye_position = mv_matrix * gl_in[0].gl_Position;
  color = sphere_color[0];
  gl_ClipDistance[0] = gl_in[0].gl_ClipDistance[0];
  gl_ClipDistance[1] = gl_in[0].gl_ClipDistance[1];
// generate four vertices of the quad:
texcoord = vec2(-1.0, -1.0);
  gl_Position = eye_position;
  gl_Position.xy += vec2(-sphere_radius, -sphere_radius);
  gl_Position = p_matrix * gl_Position;
EmitVertex();
texcoord = vec2(-1.0, 1.0);
  gl_Position = eye_position;
  gl_Position.xy += vec2(-sphere_radius, sphere_radius);
  gl_Position = p_matrix * gl_Position;
EmitVertex();
texcoord = vec2(1.0, -1.0);
  gl_Position = eye_position;
  gl_Position.xy += vec2(sphere_radius, -sphere_radius);
  gl_Position = p_matrix * gl_Position;
EmitVertex();
texcoord = vec2(1.0, 1.0);
  gl_Position = eye_position;
  gl_Position.xy += vec2(sphere_radius, sphere_radius);
  gl_Position = p_matrix * gl_Position;
EmitVertex();
EndPrimitive();
}
```

```
#version 330 core
// input:
uniform mat4 p_matrix;
uniform float sphere_radius;
uniform vec3 light_dir;
uniform float ambient_shade;
flat in vec3 color;
smooth in vec2 texcoord;
flat in vec4 eye_position;
// output:
out vec4 out_color;
// kernel:
void main() {
  float x = texcoord.x;
  float y = texcoord.y;
  float z_sq = 1.0 - x*x - y*y;
  if (z_sq <= 0.0) discard;
  float z = sqrt(z_sq);
  vec4 pos = eye_position;
  pos.z += sphere_radius * z;
  pos = p_matrix * pos;
  gl_FragDepth = 0.5 * (pos.z / pos.w) + 0.5;
  vec3 normal = vec3(x, y, z);
  float diffuse = clamp(dot(normal, mix(light_dir, normal, ambient_shade)),
0.1, 1.0);
  out_color = vec4(diffuse * color, 1.0);
}
```

G. Status of Hardware

The many-core revolution in computational hardware can be characterized by increasing thread counts, decreasing memory per thread, and architecture specific performance constraints for memory access patterns. GPUs have recently become extremely powerful HPC devices used to perform general purpose calculations. Generally, accelerators and co-processors are great for providing HPC in terms of floating point operations per second (FLOP/sec). When talking about parallelism, one frequently talks about the number of cores, e.g. 22 CPU cores in a highest-end CPU sound like a lot, but are few compared to the 3584 processing units in a GPU. The comparison of theoretical peak performances for single precision arithmetic, plotted in figure G.3, shows a five– to fifteen–fold margin when comparing high-end CPUs with high-end GPUs over time. It should be noted that the difference in performance peaks between single and double precision arithmetic of current GPUs can be up to a factor of 32, whereas for CPUs is up to a factor of 2. Comparison of theoretical peak memory bandwidth is graphed in figure G.4. In contrast to raw performance, the advantage of GPUs and Xeon Phi over a single CPU socket in terms of peak memory performance has increased from about three-fold in 2007 to ten-fold in 2016. The graphs in figures G.3 and G.4 are obtained from Karl's Rupp survey on hardware characteristics progress over time, given at `https://github.com/karlrupp/cpu-gpu-mic-comparison`. In conclusion, massively parallel architectures (AMD and NVIDIA GPUs, Intel Xeon Phis) currently provide similar peak performance, so a preference of one architecture over another should be based on development effort, maintainability, and portability aspects rather than minor differences in peak performance.

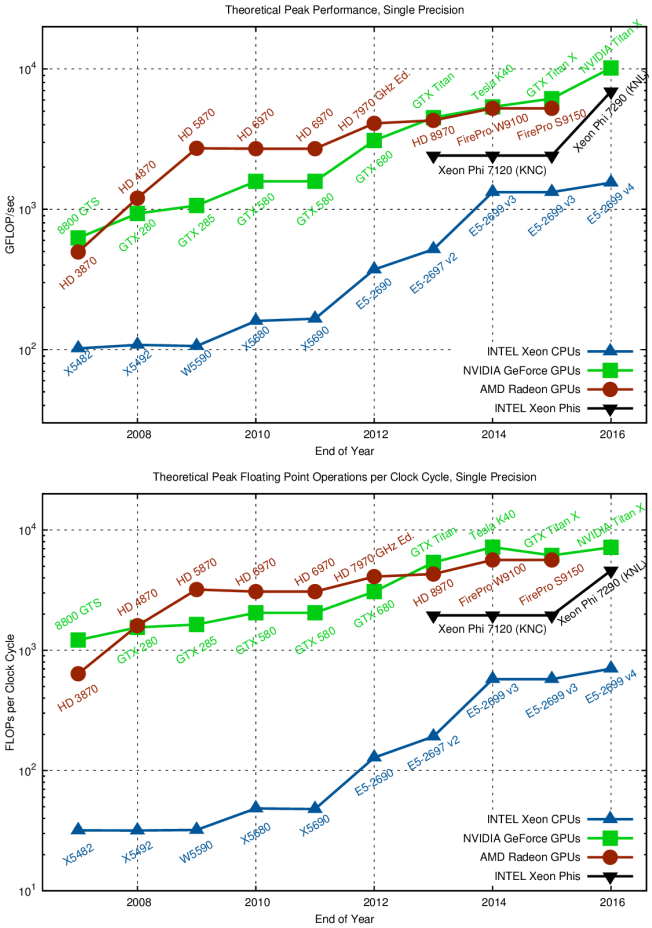

Figure G.3.: Comparison of modern hardware theoretical peak performance (top image) and theoretical peak floating-point operations per clock cycle (right image).

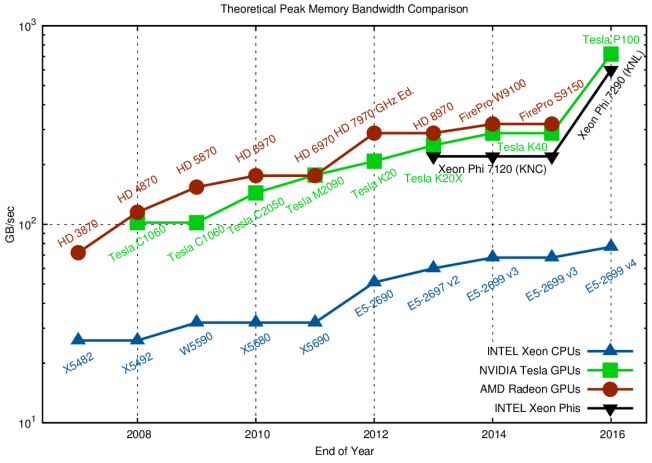

Figure G.4.: Comparison of modern hardware theoretical peak memory bandwidth.

H. Flap Wavemaking Theory

By finding the ratio between the wave height H and the wavemaker stroke s, both the direct and the inverse problems are solved at the same time [226]. The direct problem means that by prescribing the wavemaker stroke, the wave height far from the wavemaker follows as an outcome. The inverse problem means that if a certain wave height is desired far away from the wavemaker, the stroke needed as an input to the wavemaker can be calculated. By using the linear-wave theory, an explicit relation between the wave height of generated wave was in the far field and the stroke of the wavemaker can be found. This relation depends explicitly on the water depth, the hinge position, and the wave frequency, shown in figure H.5. The linearized equation model for the boundary value problem is given as:

$$\begin{aligned}
\nabla^2\phi = 0, -h \leq z \leq 0,\\
\frac{\partial\phi}{\partial z} = 0, \text{ at } z = -h,\\
\frac{\partial\eta}{\partial t}\frac{\partial}{\partial z}, \text{ at } z = 0,\\
\eta + \frac{1}{g}\frac{\partial\phi}{\partial t} = 0, \text{ at } z = 0,\\
\frac{\partial\phi}{\partial x}\frac{\partial s\left(z,\, t\right)}{\partial t}, \text{ at } z = 0.
\end{aligned}$$

The lateral boundary motion $s\left(z,\, t\right)$ for a single-flap wavemaker and sinusoidal flap is defined as:

$$s\left(z,\, t\right) = \frac{1}{2}S\left(z\right)\sin\left(\omega t\right),$$

which describes the wavemaker motion with maximum stroke $S\left(z\right)$ at a specific height, and wavemaker frequency ω. According to the solution of the problem presented by Kusumawinahyu *et al.* [226], the ratio of the wave height and the stroke at initial free surface is given by the following expression:

$$\frac{H}{S\left(0\right)} = 4\left(\frac{\sinh\left(kh\right)}{kd}\right)\frac{\cosh\left[k\left(h-d\right)\right] + kd\sinh\left(kh\right) - \cosh\left(kh\right)}{2kh + \sinh\left(2kh\right)},$$

and when the hinge of the flap is located at the bottom of the wave tank, i.e. $d = h$, $\cosh 0 = 1$, the ratio becomes:

$$\frac{H}{S\left(0\right)} = 4\left(\frac{\sinh\left(kh\right)}{kd}\right)\frac{1 + kd\sinh\left(kh\right) - \cosh\left(kh\right)}{2kh + \sinh\left(2kh\right)}.$$

When the H/S ratio is obtained, maximum flap deflection angle is obtained:

$$\alpha_{MAX} = \arctan\left(\frac{S\,(0)}{d}\right),$$

and the rotational motion of the flap is simply simulated with a sinusoidal motion:

$$\alpha\,(t) = \alpha_{MAX}\sin\,(\omega t)\,.$$

Usually, Galvin's simple theory gives a good approximation to the wave height to stroke ratio for small values of kh, but the presented solution should be used for larger values of kh [226]. The authors also apply the linear theory to a wave-maker made of two connected flaps, which is useful in generating waves in a wider range of frequencies.

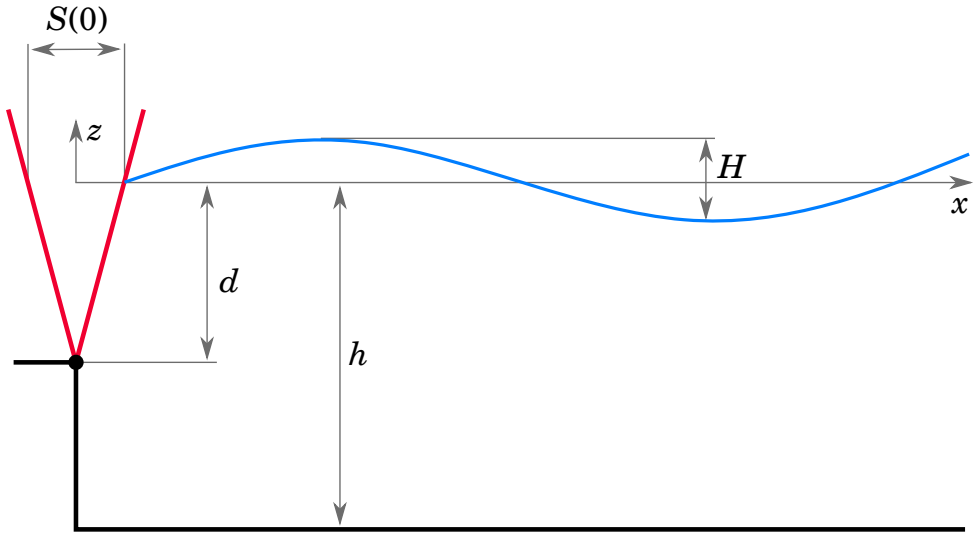

Figure H.5.: Sketch of a two-dimensional wave tank with a single-flap wavemaker.

I. Momentum-Source Wavemaking

Waves can be generated by adding source to the momentum equation within some virtual volume. The components of the momentum source vector $\boldsymbol{s} = \{s_x,\, s_y,\, s_z\}$ for a linear monochromatic wave can be defined as explained in [179]:

$$\begin{aligned} s_x &= -2\,\beta\,g\,x\,\exp\left\{-\beta\,x^2\right\}\,\frac{D}{\omega}\,\sin\left(k_y y-\omega t\right), \\ s_y &= g\,\exp\left\{-\beta\,x^2\right\}\,\frac{D}{\omega}\,\cos\left(k_y y-\omega t\right), \\ s_z &= 0, \end{aligned}$$

where $\beta = 20/w^2$, w is the source function width Classically, ω is the wave frequency, and $k_y = k\,\sin\theta$, where k is the wave number and θ is the angle between the wave direction and x axis. The source function amplitude, D, is obtained from the following expression:

$$D = \frac{H\left(\omega^2-\lambda_1\,g\,k^4\,d^3\right)\cos\theta}{\omega\,I\,k\left[1-\lambda\left(k\,d\right)^2\right]},$$

where H is the wave height, $I = \sqrt{\pi/\beta}\,\exp\{-k^2/4\beta\}$, and d is the initial water depth. For the extended Boussinesq equations of Nwogu [227], the following holds:

$$\begin{aligned} \lambda &= \frac{z_a}{d}\left(\frac{z_a}{2d}+1\right) = -0.38955, \\ \lambda_1 &= \lambda + 1/3 = -0.0562, \end{aligned}$$

where $z_a = -0.53d$ is chosen for a good agreement between the linear and the exact dispersion relation for a wide range of water depths. Using a linear combination of monochromatic waves, the momentum source function for random waves can be written as:

$$\begin{aligned} s_x &= -2\,\beta\,g\,x\,\exp\left\{-\beta\,x^2\right\}\sum_j\frac{D_j}{\omega_j}\,\sin\left(\omega_j t+\varepsilon_j\right), \\ s_y &= g\,\exp\left\{-\beta\,x^2\right\}\sum_j\frac{D_j}{\omega_j}\,\cos\left(\omega_j t+\varepsilon_j\right), \end{aligned}$$

where ε_j is the random phase, and j is the frequency component of a random wave spectrum.

J. Simulation Input

General parameters of the input file are as follows:

name Short name of the simulation, dictating how the exported files will be named. The default value is the name of the input filename.

description Optional textual description of the simulation.

dimensions Number of dimensions, 2 or 3. Must be specified.

precision Machine precision of numbers, "single" for 32-bit precision and "double" for 64-bit precision. The default value is "double".

device Device on which to perform the computation, "cpu" or "gpu". If not set, the solver automatically chooses the best device.

gravity Constant external acceleration. The default value is [0, –9.81] for 2D and [0, 0, –9.81] for 3D problems.

fluid Fluid parameters:

- **density** Density of the fluid. Must be specified.
- **viscosity** Kinematic viscosity of the fluid. Must be specified.
- **flood_points** Point from where to start filling the fluid until reaching the boundaries, specified as [[x_1, y_1, z_1], [x_2, y_2, z_2], ...]

pressure_solver Pressure solver parameters:

- **solver** Type of the iterative solver: "bicgstab", "qmrcgstab", "qmr", "gmres", "cg". The default value is "qmrcgstab".
- **max_iterations** Number of maximum allowed iterations of the solver. The default value is 700.
- **max_error** Maximum relative residual. The default value is 1e–6.

integration The integration of the velocity and position.

- **velocity** The class of velocity integration technique: 1, 2 or 3, which corresponds to BDF1, BDF2 and BDFA, respectively. The default value is 2.
- **position** The order of position integration: 1 or 2. The default value is 2.

domain Domain parameters:

- **min** The minimum location of the domain bounding box. Optionally specified for better performance.
- **max** The maximum location of the domain bounding box. Optionally specified for better performance.

resolution Discretisation parameters:

spacing Spacing between adjacent points. Must be specified.

compact_radius Radius of the compact sphere around points, where neighbours are found. The default value is 2.4.

reordering PBD parameters:

iterations Number of iterations for the PBD. The default value is 7.

radius Radius of the compact sphere around points, where neighbours are found for the PBD. The default value is 1.6.

compression Allowed compression factor between neighbours. The default value is 1.0. Values lower than 1.0 do not allow compression (and even force some amount of expansion), and values higher than 1.0 allow some amount of compression.

patches Information on patches forming bodies:

"patch_name" User-chosen name of the patch:

type Type of the patch. Can be "free_surface" or "solid".

geometry Geometry representation: "polyline", "triangle", "quad", "box" or "mesh".

file If the type is "mesh", relative file path should be specified.

points If the type is "polyline" as 2D primitive, or "triangle" or "quad" as 3D primitive, the array of points forming the primitive should be specified.

velocity Imposing motion by defining list of velocities in time as: [[t_t, [u_1, v_1, w_1]], ...]

oscillation Imposing oscillating motion:

period Period of the oscillation.

direction Direction vector of the oscillation, magnitude defines the amplitude.

axis Axis vector around which the oscillating rotation happens, magnitude defines the amplitude.

damping The damping factor.

probes An array of point probes that are fixed to the patch, with their locations specified as: [[x_1, y_1, z_1], [x_2, y_2, z_2], ...]

tank Specify external open boundaries:

min Coordinates of the minimum point of the tank box, [x_{MIN}, y_{MIN}, z_{MIN}].

min Coordinates of the maximum point of the tank box, $[x_{MAX}, y_{MAX}, z_{MAX}]$.

wave_probes An array of points from where to measure the height of the free surface, specified as [$[x_1, y_1, z_1]$, $[x_2, y_2, z_2]$, ...]

probes An array of non-moveable probes, with their locations specified as: [$[x_1, y_1, z_1]$, $[x_2, y_2, z_2]$, ...]

An example of the input file used to simulate the experiment described in section 5.5.3:

```
"description":  "Sloshing in a swaying LNG tank",
"dimensions":  2,
"precision":  "single",
"device":  "gpu",
"gravity":  [0, -9.81],
"pressure_solver":  {
  "solver":  "qmrcgstab",
  "max_iterations":  200,
  "max_error":  1e-6
},
"integration":  {
  "velocity":  2,
  "position":  2
},
"domain":  {
  "min":  [-2.2, -0.1],
  "max":  [2.2, 3.0]
},
"resolution":  {
  "spacing":  0.01,
  "compact_radius":  2.8
},
"fluid":  {
  "density":  999,
  "viscosity":  1e-6,
  "flood_point":  [0.0, 0.1]
},
"patches":  {
  "water":  {
    "type":  "free_surface",
    "geometry":  "polyline",
    "points":  [ [-10, 0.438], [10, 0.438] ]
  },
  "tank":  { "type":  "solid",
    "geometry":  "polyline",
    "points":  [ [1.51, 0.0], [1.948, 0.438], [1.948, 1.859], [1.11, 2.697], [-1.11,
2.697], [-1.948, 1.859], [-1.948, 0.438], [-1.51, 0.0], [1.51, 0.0] ],
    "oscillation":  { "period":  3.25, "direction":  [-0.1, 0] },
    "probes":  [ [1.656, 0.152] ]
  }
}
```

K. Post-processing with ParaView

The current implementation of the proposed methodology exports simulation results to VTK file format. Exported results may be imported into ParaView post-processing software. Formally, each point is a cell with one vertex, so the cells do not have topological information. To work with mesh–based post-processing features, the vertex cloud can be triangulated using the Delaunay triangulation filter. After the mesh is obtained, ParaView can use all relevant analysis options, such as section cutting, contouring, drawing streamlines, etc. For example, figure K.6 shows a screenshot captured while running the cavity–flow problem within the GUI that was rendered in a real–time manner. Figure K.7 shows a screenshot of a ParaView instance that loaded the same time instant of the cavity–flow problem, performed a Delaunay triangulation, and plotted a couple of cuts through the fluid domain.

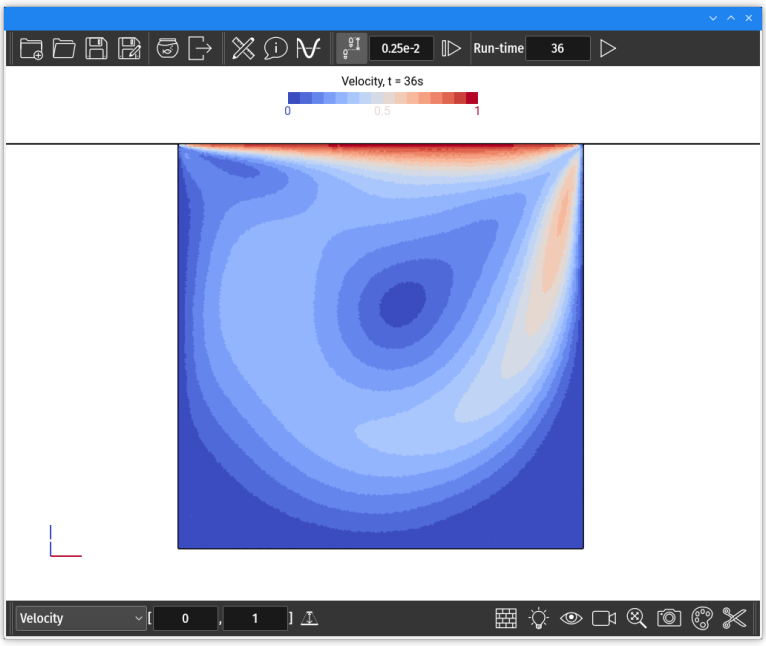

Figure K.6.: A screenshot of the real-time GUI simulating the lid-driven cavity test case.

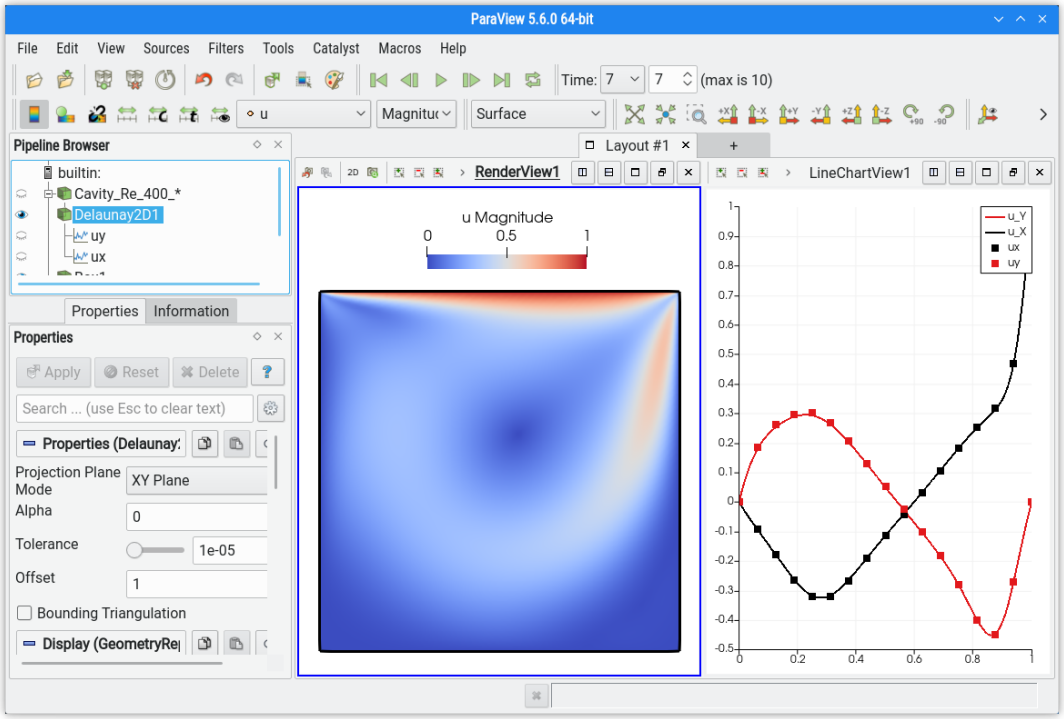

Figure K.7.: A screenshot of the ParaView setup for analysing the lid-driven cavity test case.

Curriculum Vitae

Josip Bašić was born in Split, Croatia on 17th June 1988, where he finished mathematical and natural sciences high school. In 2010 he finished the undergraduate studies in naval architecture at Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, as the best student in class, for which he got a Dean's award. The same year he enrolled in graduate studies in naval architecture at Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, and became a scholarship holder of Croatian Register of Shipping. After obtaining a master's degree in 2012, he worked for a couple of years at AVL as a developer of CFD tools for automotive industry. From October 2014 he has been working as a teaching and research assistant at Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, at Department of Mechanical Engineering and Naval Architecture. In 2017 he was awarded for the best project among research assistants at Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split. He was also awarded as the best student of postgraduate doctoral studies in Mechanical Engineering, Naval Architecture, Aeronautical Engineering, Metallurgical Engineering in 2017.