Fluid Flow in a Tank Coupled to Ship Motions

Tormod Ravnanger Landet

Master Thesis

Norwegian University of Science and Technology
Faculty of Engineering Science and Technology
Department of Marine Technology

June 2008

Abstract

How well can coupled ship motions and sloshing be simulated by simple numerical methods? The SOLA algorithm for solving the Navier–Stokes equation and an extensions to this algorithm for modelling a free surface by means of the Volume Of Fluid (VOF) method is implemented and tested. The VOF solver is then coupled to the equation of motion of a ship in waves and the coupled system is analysed to find the effect of sloshing on the ship motions. The results are discussed and compared to results from literature and experiments.

Keywords: Sloshing, SOLA, VOF, Ship Motions, Coupling, Retardation

Preface

This report constitutes the mandatory master thesis at the Department of Marine Technology at the Norwegian University of Science and Technology. The master thesis represents one semester's workload, but the thesis also builds on the work done for the project thesis written in the fall of 2007.

I would like to thank my adviser Professor Bjørnar Pettersen and co–adviser Associate Professor Håvard Holm at the Department of Marine Technology for their help in understanding the physical problem, finding literature and example code, and evaluating the results from the simulations. I would also like to thank Olav Rognebakke at DNV for helping draft the thesis scope, getting me started with a collection of articles, and for encouragement and help along the way.

Tormod Ravnanger Landet
June 17, 2008

Scope of Work

The scope of this master thesis is included below in Norwegian.

HOVEDOPPGAVE I MARIN HYDRODYNAMIKK

VÅR 2008

FOR

STUD. TECHN. Tormod Ravnanger Landet

Væskebevegelse i tank koplet til skipsbevegelser

Koplingen mellom skipsbevegelser og væskebevegelser i skipets tanker er en interessant utvidelse av dagens beregningsprogrammer for skipsbevegelser. Problemstillingen kan være aktuell for store gasstankere og FPSOer.

Kandidaten skal med utgangspunkt i volume-of-fluid (VOF) metoden lage en kode og med denne gjøre detaljerte studier av væskebevegelse i en tank med fri overflate. Antagelser og forutsetninger som gjøres og som kan ha betydning for resultatene eller skape begrensninger, skal dokumenteres.

Tanken er en del av et skip. I utgangspunktet antas det enkle skipsbevegelser med amplitude og frekvens som eksiterer væsken i tanken. Det komplete problemet tank/skip implementeres i tidsplanet. I den grad tiden tillater det kan virkningen av forskjellige tankstørrelser i forhold til skipets størrelse undersøkes. Parametere som skal studeres spesielt avtales med veileder.

Kandidaten skal i besvarelsen legge frem sitt personlige bidrag til løsning av de problemer som oppgaven stiller. Påstander og konklusjoner som legges frem, skal underbygges med matematiske utledninger og logiske resonnementer der de forskjellige trinn tydelig fremgår. I besvarelsen skal det klart fremgå hva som er kandidatens eget arbeid, og hva som er tatt fra andre kilder. Kandidaten skal utnytte de muligheter som finnes til å skaffe seg relevant litteratur for det problemområdet kandidaten skal bearbeide. Besvarelsen skal være oversiktlig og gi en klar fremstilling av resultater og vurderinger. Det er viktig at teksten er velskrevet og klart redigert med tabeller og figurer. Besvarelsen skal gjøres så kortfattet som mulig, men skrives i klart språk.

Besvarelsen skal inneholde oppgaveteksten, forord, innholdsfortegnelse, sammendrag, hoveddel, konklusjon med anbefalinger for videre arbeide, symbolliste, referanser og eventuelle vedlegg. Alle figurer, tabeller og ligninger skal nummereres. Det forutsettes at Institutt for marin teknikk, NTNU, fritt kan benytte seg av resultatene i sitt forskningsarbeid, da med referanse til studentens besvarelse.

Besvarelsen leveres 9. juni 2008 i tre eksemplarer.

Faglig veileder: Førsteamanuensis Håvard Holm

Bjørnar Pettersen
Professor

Contents

List of Symbols

Fluid flow in a tank

u	Horizontal velocity
v	Vertical velocity
Ψ	Pressure
ρ	Density
p	Ratio of fluid pressure to density, $p = \Psi/\rho$
μ	Dynamic viscosity
ν	Kinematic viscosity, $\nu = \mu/\rho$
h	The filling height in the tank
w	The width of the tank
β	Surface angle with the horizontal axis (x–axis)
C	The VOF colour function
s	The fraction of a cell boundary that is wetted by fluid

Coupled motion

ω	Incoming wave angular frequency of oscillation
$x(t)$	Motion of the ship hull
M	Mass of the ship excluding the fluid in the tank
$A(\omega)$	Added mass coefficient of the ship hull
$B(\omega)$	Damping coefficient of the ship hull motion
C	Restoring force coefficient of the ship hull
$f(t)$	Excitation force
A^∞	Infinite frequency added mass
B^∞	Infinite frequency damping
$a(\omega)$	Frequency dependent part of $A(\omega)$, $a(\omega) = A(\omega) - A^\infty$
$b(\omega)$	Frequency dependent part of $B(\omega)$, $b(\omega) = B(\omega) - B^\infty$
$h(t)$	Retardation function

Chapter 1

Introduction

Sloshing is the motion of a fluid with a free surface in a tank. Sloshing can lead to high dynamic pressures on the tank walls and roof due to large fluid motions near resonance and the low damping of the fluid motion in tanks with smooth walls. The tank may be large and located on a ship. The resultant forces from the tank wall pressures will have a large impact on the motion of the ship. The interaction between the ship motions and the resultant forces from sloshing has to be accounted for to accurately calculate the motions of ships with partially filled tanks.

This master thesis will demonstrate a way of numerically simulating the motion of fluid in a partially filled tank that is subjected to external accelerations. A simple, two dimensional model is presented that can handle mild sloshing, but no overturning of the free surface, fluid spray or mixing of fluid and air. This model uses a free surface tracking technique known as the Volume Of Fluid (VOF) method. The VOF method can in principle handle overturning, mixing and spray. It can also be extended to three dimensions. These features and extensions are beyond the scope of this thesis.

A computer program has been implemented based on the presented theory. This report will compare the results from calculations made using this program to results from literature and model tests. The accuracy in both phase, direction and amplitude of the resultant forces from the simulations are analysed to see if the numerical models give results that can be used in design of ships with partially filled tanks.

Chapter 2

Solving the Incompressible Navier–Stokes Equations

The first step towards simulating sloshing is to accurately model the fluid. The motion and pressure of the fluid in the tank will be found by solving the Navier–Stokes equation. The internal fluid will be assumed to be incompressible; an assumption which is good for fluids such as water. Hirt, Nichols & Romero (1975) developed a simple solution algorithm called SOLA for solving the Navier–Stokes equations for incompressible Newtonian fluids enclosed in a rectangular, two dimensional domain. Their method was a simplified version of the algorithm used in the Marker and Cell (MAC) method by Harlow & Welch (1965).

MAC is a method for simulating free surface flows by tracking marker particles through the flow field. The original SOLA algorithm could not deal with free surfaces, but several extensions to the algorithm have been published which can handle this. An extension of SOLA to handle free surfaces by means of a Volume of Fluid (VOF) method is described in chapter 3.

The Navier–Stokes equations can be written as follows for incompressible and two dimensional flow (White 2006):

$$\frac{\mathrm{D}u_i}{\mathrm{D}t} = -\frac{1}{\rho}\nabla\Psi + f_i + \nu\nabla^2 u_i \ , \qquad i = 1, 2 \tag{2.1}$$

The flow field variables are the velocities $u = u_1$ and $v = u_2$, and the pressure Ψ. The fluid density is ρ and the kinematic viscosity is ν. The external body forces are denoted f. The pressure will be normalised by the density so that $p = \Psi/\rho$ to simplify the following equations.

Finding the analytical solution to this partial differential equation for general geometries and arbitrary external forces is for most practical purposes impossible. The Navier–Stokes equation is very complex due to the coupling

of the velocities u_i with the pressure p. This makes the usage of numerical mathematics and computer solutions the only way to solve the equations for practical use cases. (White 2006)

When solving the Navier–Stokes equations one has to make sure that the solution satisfies the continuity criterion and that all the boundary conditions are imposed correctly on the field variables. The continuity criterion ensures that no fluid is lost or gained through the calculations. In two dimensions for an incompressible and Newtonian fluid it can be written

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.2}$$

2.1 The Momentum Equation

The Navier-Stokes momentum equation (2.1) written out for the x-component in 2D gives

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial \Psi}{\partial x} + f_x + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

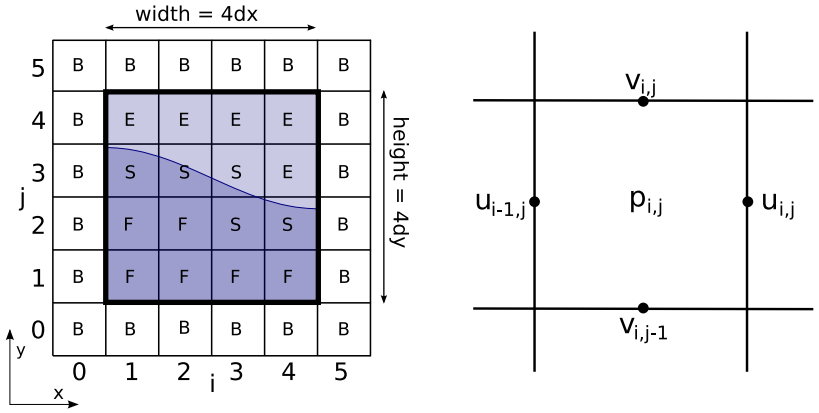
This can be rewritten by substituting $p = \Psi/\rho$ and by using continuity and the product rule of calculus. This transformation gives the equation on a differential form that is completely conservative of mass and momentum (Harlow & Welch 1965). The result is the following equation which will later be discretised

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + f_x + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{2.3}$$

2.2 Discretisation

The basic SOLA solution method is as follows: discretise the domain, i.e. the tank, by using a staggered grid (see fig. 2.1). The velocities are calculated by using the lowest order approximations of the derivatives based on the past velocity and pressure fields. These first approximations, called the guess velocities, are *not* correct as many of the approximations are only correct to the first order, $\mathcal{O}(\delta x_i)$. In addition, the interaction of pressures and velocities is not accounted for. This leads to a velocity field that does not satisfy the continuity criterion (2.2).

A staggered grid is a grid where the unknowns in each cell are located at different points in space. The SOLA algorithm calculates the pressures and the two velocity components at different locations. The use of staggered grids



(a) Gridded domain (b) Staggered grid cell

Figure 2.1 – The discretisation of the domain in the SOLA method. The labels are used for the free surface calculations below. The labels denote cells that are **F**ull, **E**mpty, **S**urface and **B**oundary.

in the solution of the Navier-Stokes equation is common because it avoids non-physical pressure oscillations which are common in normal grids where all unknowns are calculated in the corners of each cell (Langtangen 1999, p. 441).

The discretised continuity equation (2.2) is simply

$$\frac{1}{\delta x}\left(u_{i,j} - u_{i-1,j}\right) + \frac{1}{\delta y}\left(v_{i,j} - v_{i,j-1}\right) = 0 \tag{2.4}$$

The discretised equation of motion (2.3) in the x-direction becomes (similar in the y-direction):

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \delta t\left[\frac{1}{\delta x}\left(p_{i,j}^{n} - p_{i+1,j}^{n}\right) + f_x - \mathrm{FUX} - \mathrm{FUY} + \mathrm{VISX}\right] \tag{2.5}$$

Where the convective and viscous fluxes are defined as:

$$\begin{aligned}
\mathrm{FUX} = \frac{1}{4\delta x}\Big[& \left(u_{i,j} + u_{i+1,j}\right)^2 + \alpha\left|u_{i,j} + u_{i+1,j}\right| \cdot \\
& \left(u_{i,j} - u_{i+1,j}\right) - \left(u_{i-1,j} + u_{i,j}\right)^2 \\
& - \alpha\left|u_{i-1,j} + u_{i,j}\right|\left(u_{i-1,j} - u_{i,j}\right)\Big] \\
\mathrm{FUY} = \frac{1}{4\delta y}\Big[& \left(v_{i,j} + v_{i+1,j}\right)\left(u_{i,j} + u_{i+1,j}\right) \\
& + \alpha\left|v_{i,j} + v_{i+1,j}\right|\left(u_{i,j} - u_{i,j+1}\right) \\
& - \left(v_{i,j-1} + v_{i+1,j-1}\right)\left(u_{i,j-1} + u_{i,j}\right)
\end{aligned}$$

$$
\begin{aligned}
&\quad - \alpha \left| v_{i,j-1} + v_{i+1,j-1} \right| \left(u_{i,j-1} - u_{i,j} \right) \Big] \\
\text{VISX} = \quad & \nu \Big[\frac{1}{\delta x^2} \left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \right) \\
& + \frac{1}{\delta y^2} \left(u_{i,j+1} - 2u_{i,j} + u_{i,j-1} \right) \Big]
\end{aligned}
$$

Quantities without superscripts are evaluated at time $n\delta t$. The number α denotes the amount of upstream differencing. It is zero for central space differencing and one for full upstream differencing. The central form is not numerically stable. The upstream (donor cell) differencing is stable provided no fluid particle can pass through an entire cell in one timestep, δt (Hirt et al. 1975). In general α should be chosen such that

$$
1 \geq \alpha > \max \left\{ \left| \frac{u\delta t}{\delta x} \right|, \left| \frac{v\delta t}{\delta y} \right| \right\} \tag{2.6}
$$

The value $\alpha = 0.6$ has been used for the calculations in section 2.7 and further.

2.3 Iterative Pressure Calculations

The calculated guess velocities (u, v) from eq. (2.5) do *not* satisfy the continuity criterion (2.2). Hirt et al. (1975) describes a method for ensuring continuity by iteratively changing the pressures until all cells have divergences lower than a certain small number ϵ. The fluid is "pushed around" between the grid cells by changing the cell pressures and updating the inter cell velocities until continuity is established. The fluid pressure is never calculated directly, but only changed iteratively in this approach.

The continuity criterion is then

$$
\nabla \vec{u} = D < \epsilon \tag{2.7}
$$

This criterion is satisfied by substituting updated velocities

$$
\begin{aligned}
u^*_{i,j} &= u_{i,j} + \frac{\delta t \delta p}{\delta x} \\
u^*_{i-1,j} &= u_{i-1,j} - \frac{\delta t \delta p}{\delta x}
\end{aligned} \tag{2.8}
$$

(similar for v) into the continuity equation (2.4). The pressure change needed to make D equal zero is then

$$
\delta p = -D \Big/ \left[2\delta t \left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} \right) \right] \tag{2.9}
$$

After updating all cell pressures and velocities in an iterative way, the criterion (2.7) will eventually be satisfied if the initial guess velocities with applied boundary conditions are "close enough" to the real divergence free solution. Convergence is not guaranteed, so a limit has to be put on how many pressure correction iterations are allowed before giving up to avoid an infinite runtime.

The rate of convergence can in some cases be accelerated by a method called over–relaxation. A constant over–relaxation parameter is introduced and the calculated pressure change is multiplied by this factor in equation (2.9). The parameter must never be larger than 2.0, or else the iteration will be unstable. A value of 1.8 is recommended in the original SOLA article and that is what has been used for the calculations in section 2.7 and onwards.

2.4 Boundary Conditions

The SOLA algorithm introduces a layer of boundary cells outside the physical geometry. This is done to avoid changing the numerical schema near the boundaries. The boundary cells are *not* a part of the physical boundary. They are fictitious fluid cells outside the domain. The boundary cells are here denoted B while the fluid cells next to the boundary cells are denoted F (see Fig. 2.1). The velocities normal and parallel to the boundary are denoted $u_\perp$ and $u_\parallel$ respectively. The boundary itself is stationary at all times. All external motions on the tank are transferred to the tanks stationary coordinate system by changing the force field f in (2.1).

The condition of free-slip is:

$$\begin{aligned} u_{\perp,B} &= u_{\perp,F} = 0 && \text{(no flow through the wall)} \\ u_{\parallel,B} &= u_{\parallel,F} && \text{(no tangential velocity gradient)} \end{aligned}$$

The condition of no-slip is:

$$\begin{aligned} u_{\perp,B} &= u_{\perp,F} = 0 && \text{(no flow through the wall)} \\ u_{\parallel,B} &= -u_{\parallel,F} && \text{(no tangential velocity at the wall)} \end{aligned}$$

The pressure boundary condition on the walls were not given by Hirt et al. so the condition has been taken from Harlow & Welch (1965). It ensures that the pressure change across the wall is zero except for the effect of external gravity fields. It reads:

$$\frac{\partial p}{\partial x_i} = \delta x_i \; f_i \; , \qquad i = 1, 2 \tag{2.10}$$

2.5 Numerical Stability

The solution algorithm assumes that no fluid particle can pass through more than one cell in the time increment δt. For a calculation timestep with global maximum velocities u_M and v_M the criterion for δt is

$$\delta t < \min\left\{\frac{\delta x}{|u_M|}, \frac{\delta y}{|v_M|}\right\} \tag{2.11}$$

A stability criterion when analysing a problem with non-zero kinematic viscosity, ν, is that momentum must not diffuse more than approximately one cell in one timestep:

$$\nu\delta t < \frac{1}{2}\frac{\delta x^2\ \delta y^2}{\delta x^2 + \delta y^2} \tag{2.12}$$

These equations does along the stability demands on the upstream differencing factor and the over–relaxation factor compromise the criteria necessary for numerical stability of the SOLA algorithm.

2.6 The Implemented Navier–Stokes Solver

The steps detailed above can be briefly described by the pseudo–code in the two following routines. SOLA is the main program which calls Pressure-Iteration and the other routines that are straight forward implementations of the steps in the SOLA algorithm. All the routines have an upper bound of $\mathcal{O}(n)$ on the runtime where n is the total number of grid cells, except the pressure iterations if no upper limit is put on the allowable number of iteration. A limit of 1000 iterations was used for the calculations in section 2.7 and further. This results in an upper bound on the total runtime of $\mathcal{O}(t_{end}\, n)$, where t_{end} is proportional to the number of simulated timesteps.

```
SOLA
    Read-Input()
    Impose-Boundary-Conditions(u, v, p)
    for t ← 0 to t_end
        do
            Compute-Initial-Guess-Velocities(u, u_old, v, v_old, p)
            Impose-Boundary-Conditions(u, v, p)
            Pressure-Iteration(u, v, p)
            Print-Output(u, v, p)
            u_old ← u
            v_old ← v
```

PRESSURE-ITERATION(u, v, p)

while $D_{max} > \epsilon$ **and** $iter_{count} < iter_{max}$

do

UPDATE-PRESSURES-AND-VELOCITIES(u, v, p)

IMPOSE-BOUNDARY-CONDITIONS(u, v, p)

FIND-$D_{max}(u, v)$

For more information on the code that implements the above algorithm see appendix A.

2.7 Tests of the Navier–Stokes Solver

A test of the basic solver without a free surface has been undertaken to verify the correctness of the implementation of the numerical algorithm. The ability to solve the simpler problem of fluid motion in a full tank correctly is vital to solving the more complex problem of a tank with a free surface.

The chosen test case was the flow in a quadratic cavity with a prescribed horizontal fluid velocity at the top, the so called *Lid Driven Cavity Flow*. This is a purely viscous shear driven case, so it is not very relevant for sloshing simulation. There is no free surface, and the flow is driven by a constant horizontal velocity in the uppermost cells. This test case was chosen because it is well researched and results for comparison are readily available in literature.

The Reynolds number studied was $Re = U_{top}L/\nu = 1000$ defined by the side length and the prescribed velocity. A side length of one meter and a kinematic viscosity of 10^{-4} was chosen giving a velocity at the top of 0.1 m/s. The no–slip condition was used at the walls. Five different grid sizes from 25x25 to 400x400 elements were tested.

The horizontal velocities in a centred, vertical, cut were measured after the simulation had reached an approximately steady state after 300 seconds of simulation time. As can be seen from figure 2.2, the solution approaches the solution by Ghia, Ghia & Shin.

To find out whether or not the solution is converging, and if so how fast, the largest negative velocity from Ghia, Ghia & Shin (1982) in figure 2.2 was taken as a reference and the corresponding velocities from the SOLA analysis were used to calculate the relative error. Figure 2.3 shows that the solution approaches the values from Ghia et al.. More elements were not tested as the run with the maximum number of elements (400x400) took more than 37 hours to run compared to 9 hours for the 200x200 case. The run time is an approximate linear function of the number of elements, so a 800x800 grid problem would take almost a week to run.

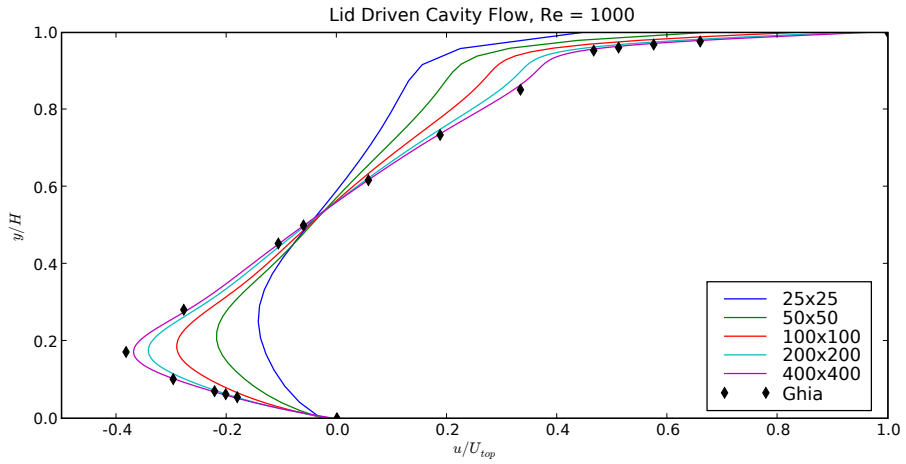


Figure 2.2 – Lid driven flow, horizontal velocities in a centred vertical cut

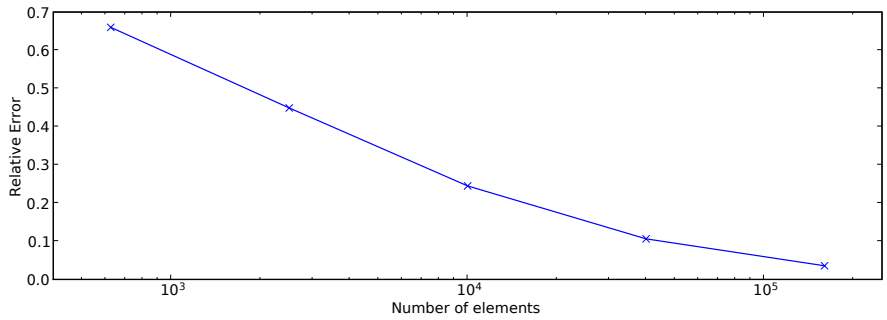


Figure 2.3 – Lid driven flow, convergence plot

The results when an approximately stationary condition had been achieved can be seen in figure 2.4. The scalar field in the background is the normalised pressure and the vectors show the velocity field.

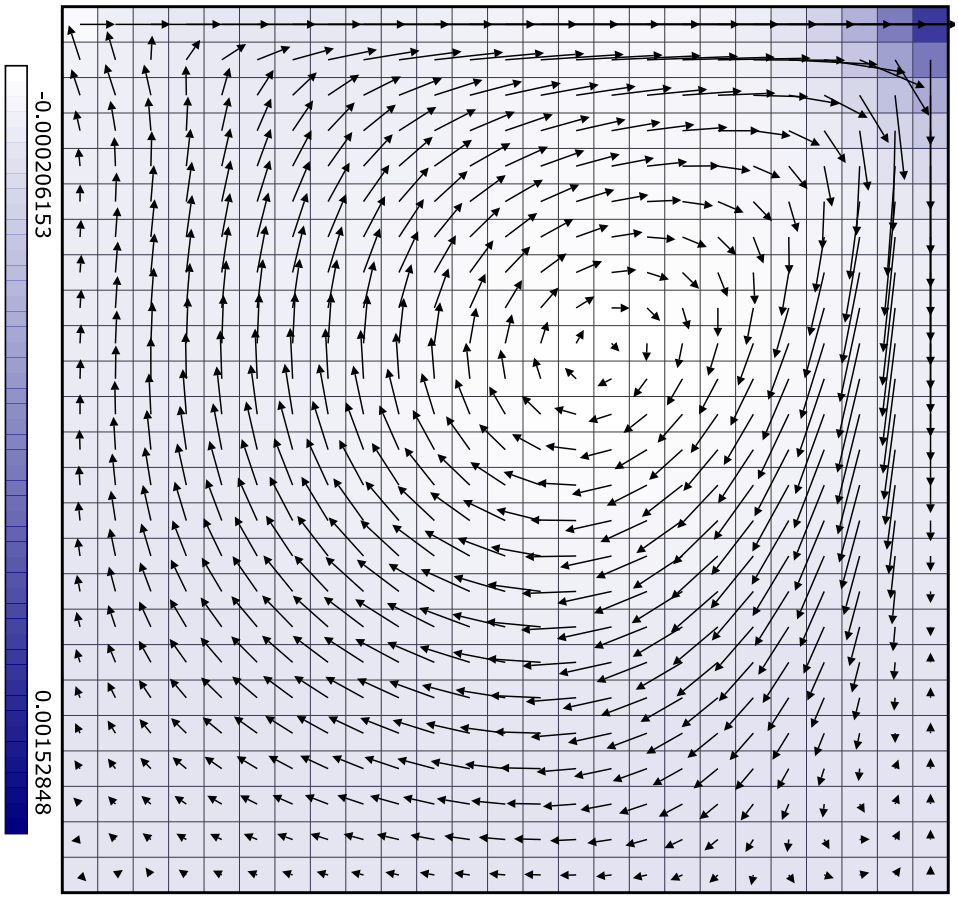


Figure 2.4 – Lid driven cavity flow at Re = 1000. The scalar field in the background is the normalised pressure. The grid size is 25x25 elements.

Chapter 3

Sloshing Simulation with the Volume of Fluid Method

Two additions have to be made to the SOLA algorithm in order to model the fluid motion in a partially filled tank. The interface between the fluid and the air above must be described in space and time and the correct boundary conditions must be imposed on the cells near the free surface.

Some simplifications of the problem are made in the following. The properties of the air flow above the free surface are neglected and zero atmospheric pressure is assumed. These simplifications are valid because the atmospheric pressure will be acting on both the inside and outside of the tank, so the resultant force on the tank wall is zero. The density of the air is assumed to be much lower than the density of the fluid, so the effect of the air flow on the fluid flow will be negligible.

In the Volume Of Fluid method (VOF) the free surface is tracked by introducing a new field variable C, the colour function. This function is unity at a point in the fluid and zero at a point in the void. C is introduced on the staggered grid in such a way that it denotes the volume fraction of fluid in each cell. The equation governing the colour function is that the total volume fraction of fluid and void should remain the same. The advection equation for C is then (Rudman 1997):

$$\frac{\mathrm{D}C}{\mathrm{D}t} = 0 \tag{3.1}$$

The free surface location within each cell is not given by the cell's C–value, so a method for tracking the fluid location in each cell must be applied to avoid unphysical diffusion. The main problem with tracking the time development of C, and the reason why we cannot simply discretise (3.1), is that C is a step function with a discontinuity at the free surface. The most basic method of

tracking the time development of C would be to find the upwind volume flux between cells, calculated from the cell boundary velocity multiplied by the volume fraction of fluid in the upwind cell to find the flux of C. This would lead to blurring of the surface and numerical mixing of fluid and void. An example of this can be seen in figure 3.1.

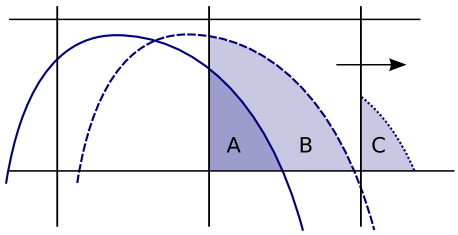
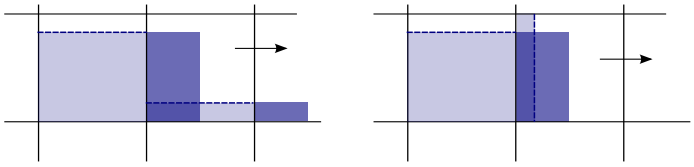


Figure 3.1 – An example of numerical diffusion. The initial state is A, B is the correct state after the next timestep (only void is fluxed to the next cell), C shows what will happen when using a flux algorithm which does not account for the fluid location in the each cell. The total flux is here half the cell volume.

Hirt & Nichols (1981) introduced the Volume Of Fluid method by publishing what they called the donor–acceptor VOF method. This method handles the free surface by tracking the fluid in rectangular sub–volumes in each cell. The sub–volumes may extend from any of the four cell boundaries and must span the entire boundary. The amount and location of fluid is then used when calculating fluxes. This will in most cases result a better estimate of the fluid fluxes, but will also give some numerical diffusion as can be seen in figure 3.2.



(a) Numerical diffusion (b) No numerical diffusion

Figure 3.2 – Two discretisations of figure 3.1. The fluid is fluxed according to the volume flux and the orientation of fluid in the cell. The dark areas are the volumes that will be fluxed according to Hirt & Nichols.

The donor–acceptor method was soon found to be a bit too simple and not accurate enough for many test cases. Several ways of computing the flux of C have been proposed. A rather simple geometrical method developed by Youngs (1982) has been shown to be among the best in several comparisons (Rudman 1997, Kleefsman 2005).

3.1 Youngs' VOF

Youngs' VOF method (Youngs 1982) considers all eight neighbouring cells to find the orientation and slope of the free surface in each cell. The surface is approximated by a straight line and will often be continuous between grid cells, but this is not guaranteed. The surface can have 16 different basic configurations in a cell, but they can all be represented by rotating the configurations in figure 3.3. These four configurations correspond to four different ways of computing the fluid fluxes. The main difference between the configurations is which of the walls that are wetted.

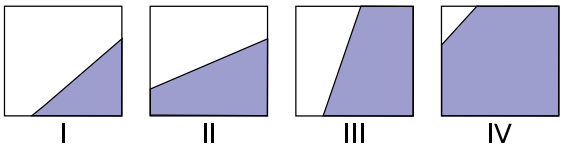


Figure 3.3 – The four possible interface reconstructions for Youngs' VOF (Rudman 1997)

Youngs' method works by first computing the basic diffusive upwind fluxes for all cells containing fluid and then computing the outwards fluxes based on the straight line approximation of the fluid/void interface for the cells containing the free surface ($0 < C < 1$).

Rudman (1997) recommends a stencil by Kothe, Mjolsness & Torrey (1991) for finding the slope of the surface, β:

$$\begin{aligned} n_{x,i,j} &= \frac{1}{\delta x}\left(C_{i+1,j+1} + 2C_{i+1,j} + C_{i+1,j-1} - C_{i-1,j+1} - 2C_{i-1,j} - C_{i-1,j-1}\right) \\ n_{y,i,j} &= \frac{1}{\delta y}\left(C_{i+1,j+1} + 2C_{i,j+1} + C_{i-1,j+1} - C_{i+1,j-1} - 2C_{i,j-1} - C_{i-1,j-1}\right) \end{aligned} \tag{3.2}$$

which gives the surface angle with the x-axis

$$\beta = \tan^{-1}\left(\frac{-n_x}{n_y}\right) \tag{3.3}$$

Rudman (1997) provides a thorough walkthrough of how the fluxes near the surface are computed in Youngs' method. The fraction of each boundary that is wetted by fluid, s, is computed from the surface normal β, the fractional volume of fluid C, and the relevant case from figure 3.3. The outwards flux is then found from the surface fraction, the surface normal and the outwards velocities. Inward directed velocities are ignored. An example of a computed flux volume can be seen in the shaded part of figure 3.5. Note that the free

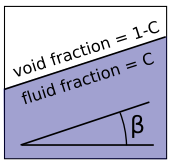


Figure 3.4 – Surface angle

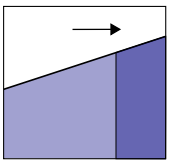


Figure 3.5 – Calculated flux volume

surface angle is taken into account in addition to the wetted part of the grid cell's right boundary.

The full set of equations for computing the surface cell flux is replicated in table 3.1. The subscripts t, r, b and l in the table denote the top, right, bottom and left edge respectively. Note that some changes have been made compared to the table in Rudman (1997). Some of the equations were found to be erroneous when all the equations in the table were recalculated to find the source of a large observed numerical damping. The equations given in table 3.1 are all calculated from simple geometrical relations and have a much lower observed numerical damping.

There are two main ways of performing the colour function fluxing when the inter cell fluxes have been calculated. The choices are the multidimensional and the direction split flux methods. The direction split method differs from the multidimensional in that it reconstructs the surface configuration once for each axis direction. Alternating for each timestep, the surface is reconstructed and fluxed first in one direction, the colour function field is updated, and then the VOF algorithm runs a second time to compute the fluxes in the other direction. Rudman (1997) recommends using a direction split approach to VOF calculations for better accuracy. Rider & Kothe (1998) prefers the multidimensional method for lower computational cost and because they find it has better symmetry preserving properties. For this report Rudman (1997) has been followed for the choice of the direction split method just as for the formulae to compute the surface normal. Both of these choices differ from the original algorithm by Youngs (1982).

Table 3.1 – Flux calculation for Youngs' method. Table from Rudman (1997) with modifications marked by *. Outward velocities are positive and flux calculations are not done for inward–pointing (negative) velocities.

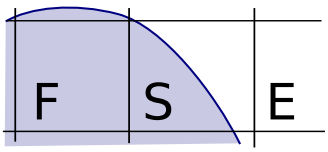
	Case I	Case II	Case III	Case IV
s_t	0	0	$C - \frac{1}{2}\cot\alpha$	$*1 - [2(1-C)\cot\alpha]^{\frac{1}{2}}$
s_r	$(2C\tan\alpha)^{\frac{1}{2}}$	$C + \frac{1}{2}\tan\alpha$	1	1
s_b	$(2C\cot\alpha)^{\frac{1}{2}}$	1	$C + \frac{1}{2}\cot\alpha$	1
s_l	0	$C - \frac{1}{2}\tan\alpha$	0	$*1 - [2(1-C)\tan\alpha]^{\frac{1}{2}}$
if $U_t > 0$	if $U_t\delta t \le (1-s_r)\delta y$ $F_t = 0$ else $F_t = \frac{1}{2}[U_t\delta t - (1-s_r)\delta y]^2\cot\beta$	if $U_t\delta t \le (1-s_r)\delta y$ $F_t = 0$ else if $U_t\delta t \le (1-s_l)\delta y$ $F_t = \frac{1}{2}[U_t\delta t - (1-s_r)\delta y]^2\cot\beta$ else $F_t = U_t\delta t\delta x - (1-C)\delta x\delta y$	$F_t = U_t\delta t(s_t\delta x + \frac{1}{2}U_t\delta t\cot\beta)$	if $U_t\delta t \ge (1-s_l)\delta y$ $F_t = U_t\delta t\delta x - (1-C)\delta x\delta y$ else $F_t = U_t\delta t(s_t\delta x + \frac{1}{2}U_t\delta t\cot\beta)$
if $U_r > 0$	if $U_r\delta t \ge s_b\delta x$ $F_r = C\delta x\delta y$ else $*F_r = U_r\delta t(s_r\delta y - \frac{1}{2}U_r\delta t\tan\beta)$	$F_r = U_r\delta t(s_r\delta y - \frac{1}{2}U_r\delta t\tan\beta)$	if $U_r\delta t \le s_t\delta x$ $F_r = U_r\delta t\delta y$ else if $U_r\delta t \le s_b\delta x$ $F_r = U_r\delta t\delta y - \frac{1}{2}(U_r\delta t - s_t\delta x)^2\tan\beta$ else $F_r = C\delta x\delta y$	if $U_r\delta t \le s_t\delta x$ $F_r = U_r\delta t\delta y$ else $F_r = U_r\delta t\delta y - \frac{1}{2}(U_r\delta t - s_t\delta x)^2\tan\beta$
if $U_b > 0$	if $U_b\delta t \ge s_r\delta y$ $F_b = C\delta x\delta y$ else $*F_b = U_b\delta t(s_b\delta x - \frac{1}{2}U_b\delta t\cot\beta)$	if $U_b\delta t \le s_l\delta y$ $F_b = U_b\delta t\delta x$ else if $U_b\delta t \le s_r\delta y$ $F_b = U_b\delta t\delta x - \frac{1}{2}(U_b\delta t - s_l\delta y)^2\cot\beta$ else $F_b = C\delta x\delta y$	$F_b = U_b\delta t(s_b\delta x + \frac{1}{2}U_b\delta t\cot\beta)$	if $U_b\delta t \le s_l\delta y$ $F_b = U_b\delta t\delta x$ else $F_b = U_b\delta t\delta x - \frac{1}{2}(U_b\delta t - s_l\delta y)^2\cot\beta$
if $U_l > 0$	if $U_l \le (1-s_b)\delta x$ $F_l = 0$ else $F_l = \frac{1}{2}[U_l\delta t - (1-s_b)\delta x]^2\tan\beta$	$F_l = U_l\delta t(s_l\delta y + \frac{1}{2}U_l\delta t\tan\beta)$	*if $U_l\delta t \le (1-s_b)\delta x$ $F_l = 0$ *else if $U_l\delta t \le (1-s_t)\delta x$ $F_l = \frac{1}{2}[U_l\delta t - (1-s_b)\delta x]^2\tan\beta$ else $F_l = U_l\delta t\delta y - (1-C)\delta x\delta y$	if $U_l\delta t \ge (1-s_t)\delta x$ $F_l = U_l\delta t\delta y - (1-C)\delta x\delta y$ else $F_l = U_l\delta t(s_l\delta y + \frac{1}{2}U_l\delta t\tan\beta)$

3.2 Boundary Conditions

To accommodate a free surface in the tank the pressure iteration has to be changed to prescribe atmospheric pressure at the free surface. In this thesis the atmospheric pressure is set to zero. The pressure in the cells containing the surface is computed from the pressure at the surface and the pressure in the cell below by assuming a linear variation. The interpolation neighbour can in general be any of the four neighbouring cells, and not necessarily the cell in the direction of gravity. This would have allowed for simulating overturning surfaces. This approach was tested but the simpler version was chosen due to better observed stability of the computations.

One important note with regards to the pressure iterations is that the full set of boundary conditions detailed below are only applied once, just after the guess velocities are calculated. Only the wall boundary conditions are applied after each pressure iteration. A full set of boundary conditions that did not cause oscillations and divergence in interaction with the pressure iteration algorithm has not been found.

Boundary conditions must also be imposed on the velocity field. There are five different types of velocities at the free surface when grouping the velocities at the cell faces by the degree of filling in the bordering cells. The velocities in the following are named according to figure 3.6.



To/From	Full	Surface	Empty
Full	FF	FS	
Surface	FS	SS	SE
Empty		SE	EE

Figure 3.6 – Cell naming scheme and the corresponding cell face velocities

The first class of velocities are those that can be calculated by the momentum equation. These velocities are FF, SS and FS. The second class of velocities are the SE velocities. The choice of how to calculate this class of velocities will have a big impact on the stability and correctness of the results. The last class are the fictitious EE velocities that have to be established in order to use the same numerical scheme for the calculation of the velocities near the surface as in the rest of the domain. The need for determining these velocities is purely an artefact of the Navier–Stokes finite volume solver. (Chen, Johnson & Raad 1995, Kleefsman 2005)

In the original Marker and Cell (MAC) method by Harlow & Welch (1965) the free surface normal velocities were calculated by demanding conservation of mass. Chen et al. (1995) also used mass conservation in their improved free surface velocity boundary conditions for the MAC method. Kleefsman

(2005) writes that the conservation of mass method is not very accurate for simulation of surface waves. The problem lays in the assumption of mass conservation which does not apply to surface cells with varying degrees of filling. Kleefsman recommends a mixed method which uses both conservation of mass and extrapolation of internal velocities to calculate the FS velocities. For simplicity, and because no big problems were reported for sloshing simulations by Gerrits (2001), the condition of mass conservation was chosen for implementation.

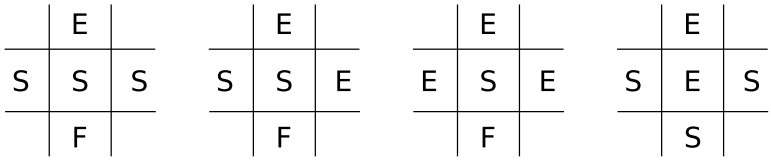


Figure 3.7 – Four possible label configurations around a surface cell. The indicated cell is the centre cell for all cases except the rightmost where it is the bottom cell.

For the leftmost configuration in figure 3.7 the following equation is used for preserving mass. The vertical velocity is chosen such that the divergence is zero.

$$v_{i,jt_i} = v_{i,jt_i-1} - \frac{\delta y}{\delta x}(u_{i,jt_i} - u_{i-1,jt_i}) \tag{3.4}$$

For the cases in the middle the top SE velocity is set equal to the bottom FS velocity. The right horizontal SE velocity is set equal to the velocity on the left side. This obviously preserves continuity. For the rightmost case the top SE velocity is left as is. After some trial and error it was found that not treating this case different from its neighbours on the sides was the best solution.

Two methods have been tested for the EE velocities normal to the surface as no description of these were found in the studied literature. The first method was extrapolation from the SE velocities to preserve a smooth field for the velocity solver. This lead to quickly diverging solutions and was abandoned. The second method was setting all EE velocities normal to the surface to zero. This lead to simulations that seemed to give accurate results, so this method was chosen. When studying the original SOLA–SURF code implementing a surface height function approach the same method was found, albeit the code only gave this solution for small surface displacements. When the surface retracted from a cell the velocities calculated for the fluid that previously occupied the cell were left in place (Hirt et al. 1975). No reason for this behaviour was described, so the implemented code explicitly sets the EE normal velocities to zero.

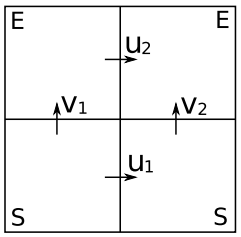


Figure 3.8 – Velocity labels near a parallel EE velocity

The EE velocities parallel to the free surface are calculated from balancing the forces at the surface and demanding that the void inflicts no tangential stresses in the fluid (Gerrits 2001):

$$\mu \left(\frac{\partial u_{\perp}}{\partial x_{\parallel}} + \frac{\partial u_{\parallel}}{\partial x_{\perp}} \right) = 0 \tag{3.5}$$

where $\mu = \rho\nu$ is the dynamic viscosity of the fluid. This gives, in two dimensions

$$\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = 0 \tag{3.6}$$

This equation can be discretised on the grid in figure 3.8 giving

$$\frac{u_2 + u_1}{\delta y} + \frac{v_2 - v1}{\delta x} = 0 \tag{3.7}$$

One final change in the boundary conditions is the special treatment of the roof. The surface may hit the roof of the tank, and if it does it should be allowed to detach without sticking, which it will without special treatment due to the condition of no flow through the tank boundaries. The only treatment of roof impact done in this thesis is a small change to the boundary conditions on the roof to allow for velocities that are pointing into the tank, while still setting outwards pointing velocities to zero. This will allow for simulating roof impact in a few cases where the free surface rises to just touch the roof before descending again. Heavy impacts and fast water upraise in jets along the walls will still cause the solver to fail as it causes an overturning free surface.

3.3 The Implemented VOF Solver

The only change to the overall structure of the calculations when adding a free surface is the inclusion of an Update-Surface-Position(u, v) routine after the Pressure-Iteration step in the pseudo code in section 2.6.

In addition the boundary condition and the pressure iteration routines are altered.

The flux algorithm to update the free surface position works on the assumption that the velocity field is completely divergence free. This assumption is never fully satisfied because the pressure iteration routine will never result in a truly divergence free velocity field unless an infinite number of iterations are performed. The result of this is that errors will be introduced over time. These errors can lead to small pockets of void artificially forming in the interior of the fluid.

The solver does not implement support for mixing of fluid and void, so void cells in the interior of the fluid is a stop condition of the solver. As a band aid to this problem all interior fluid cells are forced to remain at a filling level of unity at all times. This will over time increase the amount of fluid in the tank. Other small errors will also affect the average filling level. To counter this, a routine is run after the surface position is updated to correct the amount of fluid in the tank by subtracting or adding an equal amount of fluid to each column of cells and thus restoring the tank to its original average filling level. Since these corrections are very small for each time step it is believed that they have a low influence on the correctness of the solver.

The boundary condition code needs to be able to work no matter how the surface is oriented. A rotation mechanism is implemented to facilitate this. A rotated view of the pressure, colour and velocity fields can be extracted by an algorithm that is straight forward for the values that are defined in the cell centres and a bit more complicated for the values that are defined at cell edges. This would have enabled the solver to handle arbitrary surface orientations, but due to problems with the implementation of the generic pressure boundary conditions, and the choice to limit the validity of these in section 3.2, no overturning is supported.

Just as for the basic Navier–Stokes solver, the time complexity of the VOF solver is $\mathcal{O}(t_{end}\, n)$ where n is the number of elements and t_{end} the total simulated time. For more information on the implementation see appendix A.

3.4 Tests of the VOF Solver

The two major parts of the free surface extension to SOLA are the VOF flux algorithm and the boundary conditions on the free surface. The flux algorithm is the easiest to test since it can be tested independently of the rest of the solver code. Realistic tests of the boundary conditions involve both the flux algorithm and the basic Navier–Stokes equation solver. Many micro tests of the boundary condition code have been performed, but it is hard to predict the behaviour and stability of the solver as a whole without extensive realistic testing.

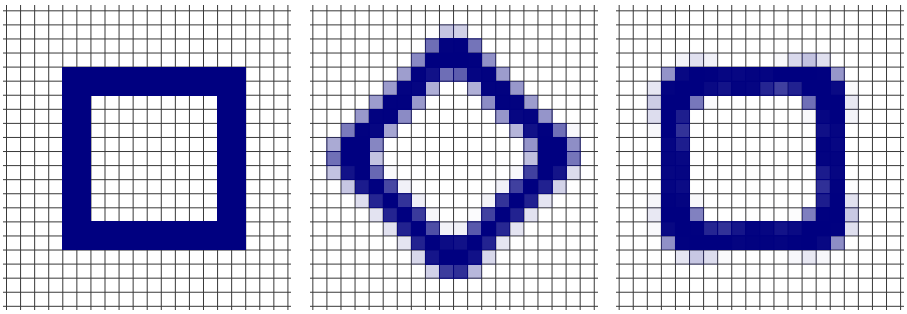
Many tests have been run while developing the VOF solver code and they are not all written about below. In general it can be said that the major challenge is the choice of boundary conditions at the free surface. The results below are all generated with the choices from section 3.2, but some tests have been run with more success with other choices during the development of the code. No generic choice of boundary conditions has been found during the development of the VOF solver that was robust with regards to geometry, discretisation and excitation.

3.4.1 Testing the VOF Flux Algorithm

The VOF flux algorithm has been tested by prescribing analytical velocity and pressure fields. The tests were performed without any boundary conditions or pressure iterations. The pressure is always zero and the prescribed velocity field is chosen such that it is divergence free.

Basic convection and shear tests have been undertaken to verify that the algorithm can track a complex surface through the regular grid. The most complex test that has been run is a case with a square fluid mass with a cutout in a potential vortex velocity field. The results for one revolution of the square can be seen in figure 3.9.

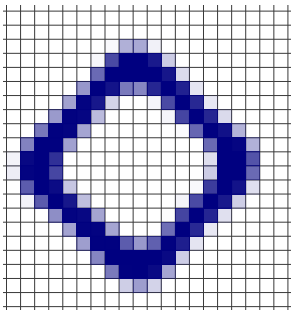
From the figures it can be seen that the flux algorithm manages to track the square rather well trough the vortex revolutions. Some blurring of the edges occurs as the square rotates, but the original form can be easily recognised. Two modifications to the flux algorithm used for the sloshing simulations have been made to get these results. No correction is made to the filling level as this makes no sense in this context. Also, the internal fluid cells (F-cells) are not forced to be at $C = 1.0$ after each flux calculation.



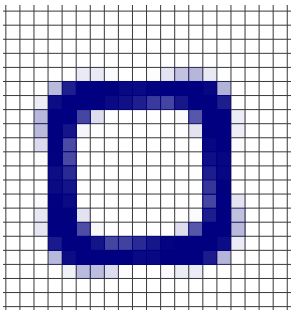
(a) $\theta = 0$

(b) $\theta = \pi/4$

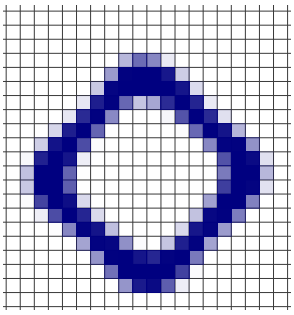
(c) $\theta = \pi/2$



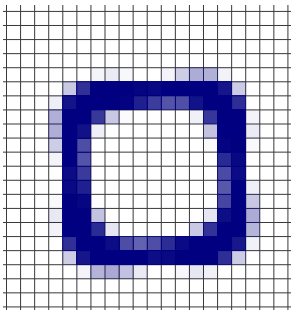
(d) $\theta = 3\pi/4$



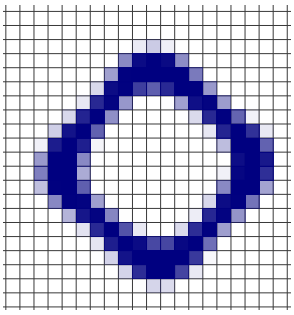
(e) $\theta = \pi$



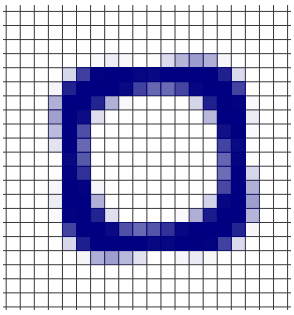
(f) $\theta = 5\pi/4$



(g) $\theta = 6\pi/4$



(h) $\theta = 7\pi/4$



(i) $\theta = 2\pi$

Figure 3.9 – Fluxing of a quadratic fluid shape in a potential flow vortex. The rotation is θ and $\delta t = \pi/100$

3.4.2 Testing a Tilting Tank

One of the problems with free surface boundary conditions is that they may introduce energy into the flow and set of unphysical motion in the tank. This has been a problem with some choices of boundary conditions, and in some cases still is a problem. To test this, a tank is tilted such that the surface is at an angle to the x–axis. The expected result is that the free surface forms a straight line normal to the direction of gravity, and that there is almost no motion in the tank as the tilting is done slowly to avoid triggering the first sloshing mode.

The test case below is a tank that is 1.73 meters wide and has a filling level of 0.5 meters. The external force field is slowly changed in a smooth, quarter sine, motion so that f_x goes from 0.0 to -3.4 m/s^2. The vertical gravity is constant at $f_y = -9.81$ m/s^2. The discretisation is such that $\delta x \approx \delta y \approx 0.02$ m. This same case failed with unphysical motion being induced for a tank of width 37.6 cm and a filling of 18.6 cm (the same dimensions as used below for testing the resonant motion). The motion caused a vortex to form in the fluid with an increasing velocity at the surface.

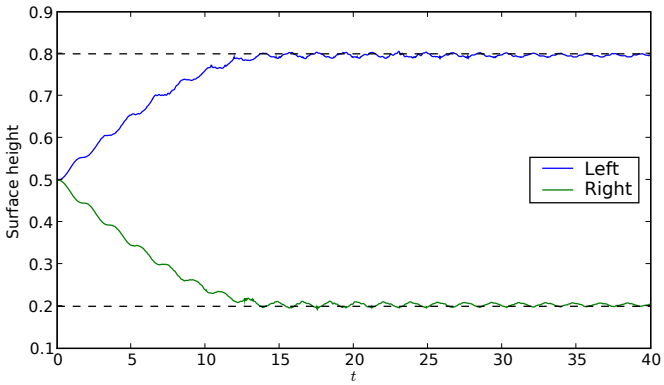


Figure 3.10 – The surface displacements at the walls of the tank with the analytical solution in dashed lines.

As can be seen from figure 3.10, the surface oscillates around the expected level at both walls. No unphysical velocities such as vortexes appear on the screenshot from a visualisation in GLview in figure 3.11. The velocities in the tank are all low. The velocity at the surface is almost zero as expected.

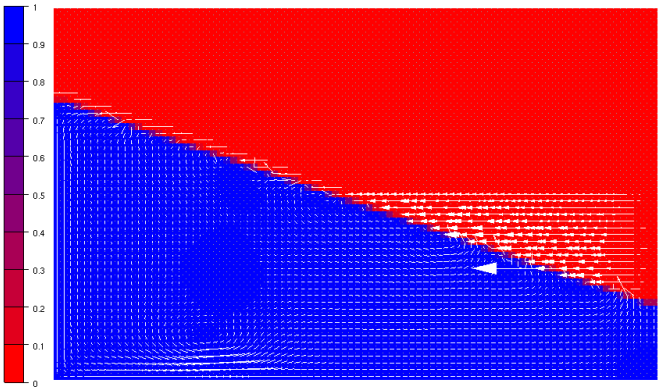


Figure 3.11 – The final state of the simulation of the tilting tank from Ceetron GLview. The scalar field is the colour function; the vector field is the velocity. Note that the one large velocity that can be seen is situated in the void. It is a leftover from when the surface passed through the cell and has no effect on the solution.

3.4.3 Testing the Resonant Sloshing Modes

Fluid motion in a rectangular tank has several resonant modes of motion. The first half sine and three quarter sine modes can be seen in figure 3.12. The resonant modes of motion will have a large impact on the motions of fluid in a tank for excitation periods well away from resonance. The Eigen–frequencies of the fluid motion can be found from (3.8) where n is the mode number, w is the tank width, h is the average filling height and g is the acceleration of gravity. (Solaas 1995)

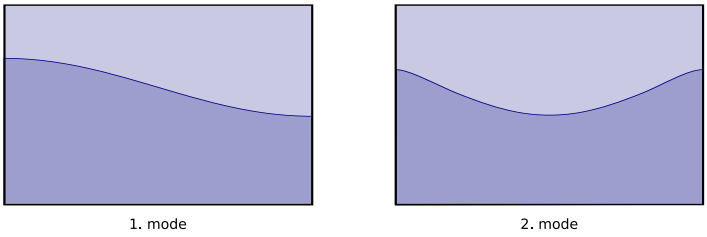


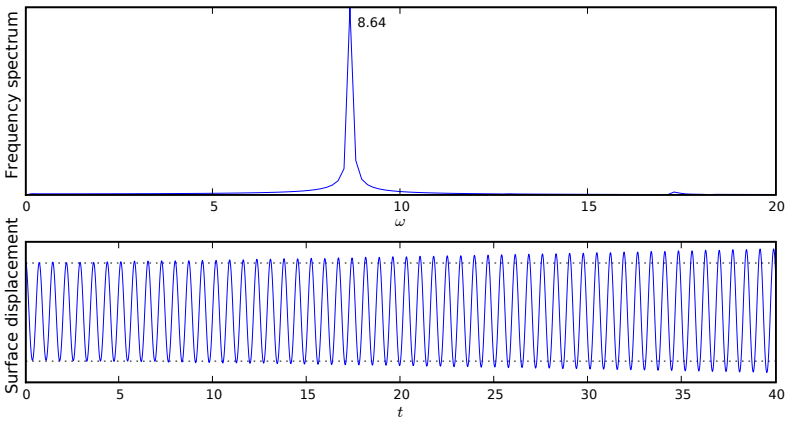
Figure 3.12 – The two first sloshing modes

$$\omega_n = \sqrt{\frac{n\pi}{w} g \tanh(\frac{n\pi}{w} h)} \tag{3.8}$$

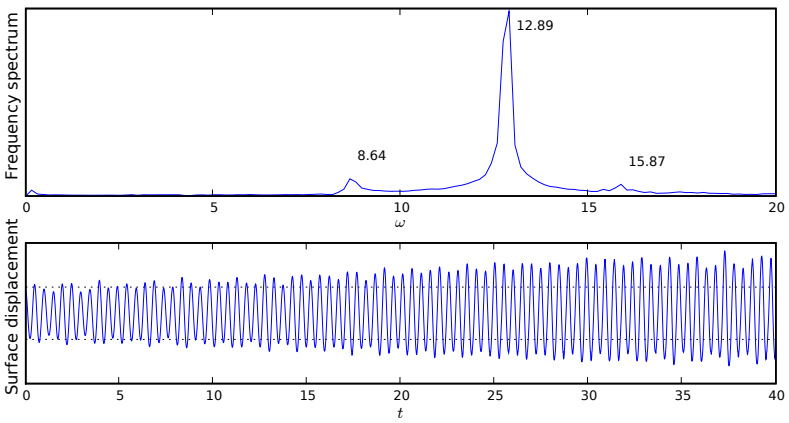
A simulation to find the resonant frequency of the first two sloshing modes was done by starting with no velocities, free surface configurations as in figure 3.12 and hydrostatic pressure accounting for the surface displacement.

The width of the tank was 37.6 cm and the average filling height was 18.6 cm. The initial surface displacement was 3% of the filling height. The chosen fluid was water. The reason for these choices were that they are used in the next chapter to compare with model tests done by Rognebakke (2002).

Identical results as those calculated by starting out of static equilibrium were had by starting with an impulse acceleration at $t = 0$ and the surface configuration initially at equilibrium. The spectrum and time series for this were very similar to the results from the simulation of the first sloshing mode that can be seen in figure 3.13(a).



(a) 1st mode



(b) 2nd mode

Figure 3.13 – The frequency spectrum of the surface elevation near the left wall from simulations of the first two sloshing modes. The frequencies of the peaks are written on the graphs.

As can be seen from figures 3.13(a) and 3.13(b), the frequency of the simulation results fit very well with the analytical solutions. The analytical frequencies are 8.66 and 12.78 rad/s for the first and second mode respectively. The results for the first mode show a better fit than the results for the second. This is also true for the amplitudes which should stay at 3% of the tank filling height as there is no damping in the system.

Negative damping can be seen from the figures as the amplitudes of the surface displacement increases over time. This is particularly evident on the second mode. The reason for this negative damping remains unknown, but it seems to have little influence when the tank is driven by excitations that do not trigger resonant sloshing.

Chapter 4

Coupled Simulation of Sloshing and Ship Motions

4.1 Ship Motions

The rigid body motions of a ship hull in waves can be modelled by a set of rather simple differential equations under the hypothesis that the motions are linear with respect to the incident wave. This means that the principle of superposition applies and that the results from multiple waves can be added together to form the total response to an irregular wave. The assumption of linearity is rather good, and many tests have been performed that seem to validate it for moderate wave conditions and forward speeds. Ogilvie (1964) presents results that show a very good agreement with model tests, particularly in irregular sea.

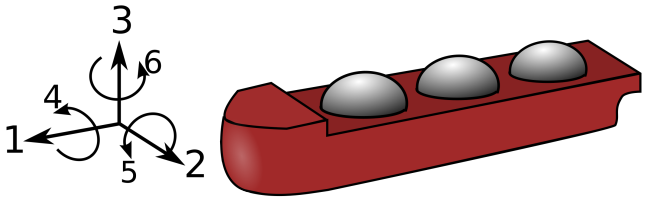


Figure 4.1 – The six rigid body motions of a ship

The motion, $x(t)$, of a ship can be described by the following equation when assuming the ship motions to be time invariant, meaning that there are no parts of the equation that depend on the absolute time except the excitation force and the response.

$$\big[M + A(\omega)\big]\ddot{x} + B(\omega)\dot{x} + Cx = Fx \tag{4.1}$$

where the terms are as follows

$$
\begin{aligned}
M &- \text{Structural mass and inertia} \\
A(\omega) &- \text{Added mass (frequency dependent)} \\
B(\omega) &- \text{Linear damping (frequency dependent)} \\
C &- \text{Restoring force} \\
F &- \text{Harmonic excitation force proportional to } x
\end{aligned}
$$

In the following all equations are given for one degree of freedom. The above terms can also describe a system with six degrees of freedom if A, B, C and M are 6×6 real matrices. F is then a 6×1 vector of complex numbers to allow the force function to be out of phase with the motion. No further mention of more than one degree of freedom is given, but all equations are general if the terms are thought of as having more than one dimension.

The response, $x(t)$, of a linear system to an arbitrary force, $f(t)$, can be written as follows by making use of the response, $r(t)$, to a unit impulse:

$$x(t) = \int_{-\infty}^{t} r(t-\tau) f(\tau) \mathrm{d}\tau \tag{4.2}$$

Unfortunately, finding a generic expressing for $r(t)$ is difficult as A and B are frequency dependent. Cummins (1962) shows that the hydrodynamic equivalent of (4.2) can be written

$$\phi = \Lambda \dot{x} + \int_{-\infty}^{t} \Upsilon(t-\tau) \dot{x}(\tau) \mathrm{d}\tau \tag{4.3}$$

where ϕ is the velocity potential resulting from an arbitrary motion $x(t)$. Λ is the velocity potential during an impulse displacement and Υ is the potential after the impulse when waves radiate away.

Cummins further shows that this can be used to find an expression for the equation of motion (4.1) where the excitation force is no longer required to be harmonic. The transient motions can be described by use of a retardation function, $h(t)$.

$$\left[M + A^{\infty}\right] \ddot{x} + B^{\infty} \dot{x} + Cx + \int_{-\infty}^{t} h(t-\tau) \dot{x}(\tau) \mathrm{d}\tau = f(t) \tag{4.4}$$

The added mass and damping are here decomposed into frequency dependent and infinite frequency parts

$$
\begin{aligned}
A(\omega) &= a(\omega) + A^{\infty} \\
B(\omega) &= b(\omega) + B^{\infty}
\end{aligned}
$$

4.2 The Retardation Function

Ogilvie (1964) gives the following expressions for the retardation function

$$
\begin{aligned}
h(t) &= \frac{2}{\pi}\int_0^\infty b(\omega)\ \cos\omega t\ \mathrm{d}\omega \\
h(t) &= -\frac{2}{\pi}\int_0^\infty \omega\ a(\omega)\ \sin\omega t\ \mathrm{d}\omega
\end{aligned}
\tag{4.5}
$$

Ogilvie also shows that a relationship between the added mass and damping coefficients can be found using a Hilbert transform, also known as a Kramers–Kronig relationship

$$
\begin{aligned}
A(\omega) - A^\infty = a(\omega) &= \frac{2}{\pi}\int_0^\infty \left[B(\omega_1) - B^\infty\right]\frac{\mathrm{d}\omega_1}{\omega_1^2 - \omega^2} \\
B(\omega) - B^\infty = b(\omega) &= -\frac{2}{\pi}\int_0^\infty \left[A(\omega_1) - A^\infty\right]\frac{\omega_1^2\mathrm{d}\omega_1}{\omega_1^2 - \omega^2}
\end{aligned}
\tag{4.6}
$$

These relations hold for any system where $h(t) = 0$ for $t < 0$, i.e. any causal system (Price & Bishop 1974, p. 203). The relations can be used to check the accuracy of the modelling of $A(\omega)$ or $B(\omega)$ if they are not known analytically, but only for discrete values of ω. An analytical model must then be created and it can be tested by transforming one set of coefficients to the other and checking how well they fit to the given discrete values. An other way of validating the analytical model is to find the retardation function and use the following equations from Ogilvie (1964, p. 38). This is easier to perform numerically than the integral in (4.6) which has a problematic denominator.

$$
\begin{aligned}
A(\omega) &= A^\infty - \frac{1}{\omega}\int_0^\infty h(t)\sin\omega t\mathrm{d}t \\
B(\omega) &= B^\infty + \int_0^\infty h(t)\cos\omega t\mathrm{d}t
\end{aligned}
\tag{4.7}
$$

4.3 Modelling the Added Mass

The retardation function can be calculated from the added mass as shown above. A continuous representation of $a(\omega)$ is needed in order to perform this calculation. Greenhow (1986) writes that the asymptotic behaviour of

the two quantities connected by the Kramers–Kronig relationship in (4.6) is such that

$$a(\omega) = \frac{k_1}{\omega^2} + \frac{k_2}{\omega^4} + \cdots \quad \text{when} \quad \omega \to \infty \tag{4.8}$$

Kvålsvold (1994) uses the following expression for $A(\omega)$ when the added mass is known for a set of frequencies along with the added mass at infinite frequency:

$$A(\omega) = \begin{cases} \hat{A}(\omega) & \text{for } \omega \leq \omega_u \\ A^\infty + \frac{k_1}{\omega^2} + \frac{k_2}{\omega^2} & \text{for } \omega > \omega_u \end{cases} \tag{4.9}$$

The function $\hat{A}(\omega)$ interpolates the discrete values of the added mass. ω_u is the upper frequency of the discrete values above which no values are known except A^∞. A good interpolation of the discrete values is obtained by using cubic B-splines. The constant k_1 is found by demanding a continuous representation of A. Kvålsvold finds that $k_2 = 0$ is a reasonable approximation.

Unlike Kvålsvold, Rognebakke (2002) uses both k_1 and k_2 to approximate the asymptotic behaviour. He calculates k_2 from the continuity criterion, just like Kvålsvold did with k_1. He then solves a boundary value problem to find k_1. The following equation is the result, where ϕ^∞ is the infinite frequency velocity potential and S_F is the free surface:

$$k_1 = -\rho g \int_{S_F} \left(\frac{\partial \phi^\infty}{\partial z}\right)^2 \mathrm{d}s \tag{4.10}$$

Equation (4.10) does not guarantee a smooth transition between $\hat{A}(\omega)$ and the asymptotic behaviour, but it does give results that correspond very well to other results from literature (Rognebakke 2002). The advantage of Kvålsvold's (1994) approach is that it needs no further information about the body than what is given by the discrete added mass coefficients.

A combined approach is constructed where k_1 and k_2 are found by demanding continuity in both $A(\omega)$ and its derivative. A comparison between Kvålsvold's C^0 continuous approach and the implemented C^1 continuous approach can be seen in figure 4.2. It should be noted that a smoother transition does not necessarily mean that the curve is closer to the real behaviour of the added mass coefficient. In the case of figure 4.2, ω_u is chosen such that the C^1 continuous approach gives an asymptote that corresponds very well with the reference, which is a dataset where the added mass is calculated for many more frequencies than what is normally done ($\omega \in [0, 50]$).

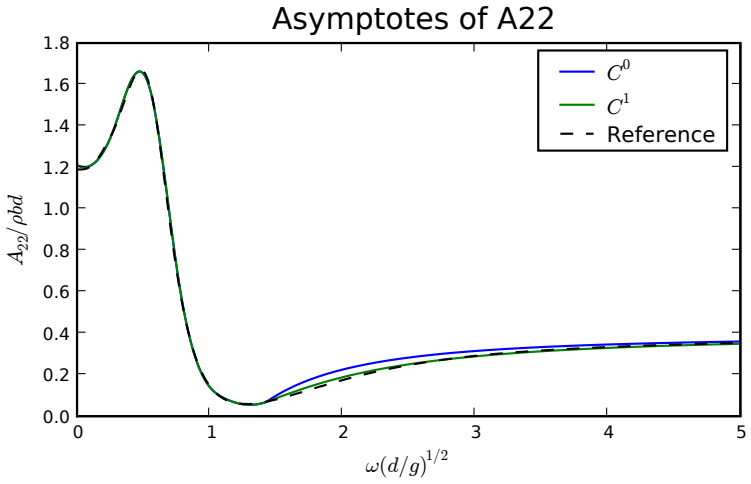


Figure 4.2 – Two approaches for the asymptotic added mass behaviour plotted against frequency compared to a case where the added mass is calculated for frequencies 0 to 50. $\omega_u = 10.0$ rad/s

4.4 The Implemented Coupled Solver

The implemented solver only handles sway motion, but an extension to more degrees of freedom should be straight forward. The solver uses the C^1 continuous approach from section 4.3 for modelling the infinite frequency limit of the added mass.

The implementation uses a simple first order Euler integration routine. Due to the demands on the time step in the VOF solver, δt is deemed sufficiently small for this simple method to be accurate. As a perfect coupling of the ship motions and the sloshing simulation is not possible, the sloshing simulation runs with an input acceleration calculated from the ship velocity by a first order backward difference formula. The VOF simulation runs one time step with this acceleration and the integrated pressure forces are returned. This sloshing force is then fed back in to the simulation as a contribution to the total exciting force in the next time step. The pseudo code below shows the main structure of the coupled solver.

The time complexity of calculating each convolution integral is $\mathcal{O}(t)$. Looking at the convolution operations needed to simulate a series of length t, the total runtime of the convolution has an upper bound of $\mathcal{O}(t^2)$. This can be improved by looking back in time only a period $\theta < t$. This will give the correct result if $h(t) = 0$ for all $t > \theta$. The upper bound on the time complexity of each evaluation of the convolution integral is then $\mathcal{O}(\theta) = \mathcal{O}(1)$ if θ is a constant. The upper bound of the whole simulation will be $\mathcal{O}(t)$ which is a significant improvement. For the present case, the VOF solver is

much slower than the convolution calculation so this optimisation will not have a significant contribution to the overall run time except for very long simulations.

```
COUPLED-SOLVER
    READ-INPUT()
    h(t) ← CALCULATE-RETARDATION-FUNCTION(a(ω))
    for t ← 0 to t_end
          do
             f_w ← WAVE-EXCITATION-FORCE(ω, ζ_a)
             f_h ← HYDRODYNAMICAL-FORCE(B^∞, C, [x, ẋ]_old)
             f_c ← RETARDATION-FORCE(ẋ_old, h)
             f_s ← RUN-SLOSHING-TIME-STEP(ẍ_old)
             ẍ ← COMPUTE-ACCELERATION(f_w, f_h, f_c, f_s, M, A^∞)
             x, ẋ ← EULER-STEP(ẍ)
             [x, ẋ, ẍ]_old ← [x, ẋ, ẍ]
```

4.5 Tests of the Coupled Solver

Rognebakke (2002) gives experimental and numerical results for the box shaped hull section in figure 4.3. The section has two internal tanks, one in each end. The chosen configuration is one with a structural mass of 37.01 kg, an internal filling height of 18.6 cm in one tank and the other tank empty.

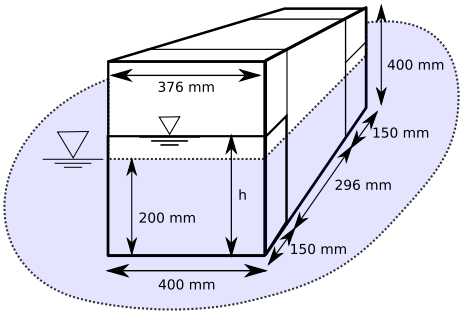


Figure 4.3 – The hull and tank geometry in the coupling simulations.

4.5.1 Testing the Retardation Function

If the asymptotic behaviour of the added mass is modelled correctly, the equalities in (4.7) should hold and $A(\omega)$ and $B(\omega)$ should be recreated very well from the retardation function. The results from calculations using the C^1 continuous added mass coefficient from figure 4.2 can be seen in figure 4.4.

A very good fit with the added mass can be seen as expected. The damping does not fit as well, but it is very close.

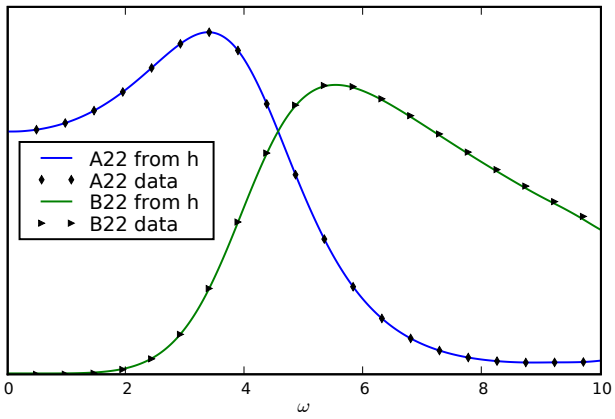


Figure 4.4 – A comparison between the coefficients calculated from the retardation function and the original data points.

Looking at the equation for stable (4.1) and transient motion (4.4) it is clear that the contribution from the convolution term is such that

$$\int_{-\infty}^{t} h(t-\tau)\dot{x}(\tau)\mathrm{d}\tau \quad \rightarrow \quad a(\omega)\ddot{x} + b(\omega)\dot{x} \tag{4.11}$$

when the excitation, $f(t)$, is purely sinusoidal at an angular frequency ω.

The results from simulations show that for low wave frequencies equation (4.11) fits very well in the steady–state part of the time series. The contribution from the convolution term is slightly lower that the result would have been with only constant hydrodynamic coefficients for frequencies higher than approximately the first natural period of sloshing. A test has been run for the system in figure 4.3 with a pure sinusoidal incoming wave at different frequencies. The natural frequency of the tank is 8.66 rad/s. The results can be seen in figure 4.5.

4.5.2 Testing the Coupled Motion

The chosen configuration of the test section from figure 4.3 with one tank filled up to 18.6 cm was subjected to an incoming wave with an amplitude of 1.5 cm and a frequency of $\omega \in [0.5, 12]$ rad/s. The grid is coarse at 19x19 elements, but a change to 38x38 elements showed only small changes so the discretisation was deemed good enough. The time step is 0.005 s. The program managed to simulate 50 seconds at most wave frequencies, but

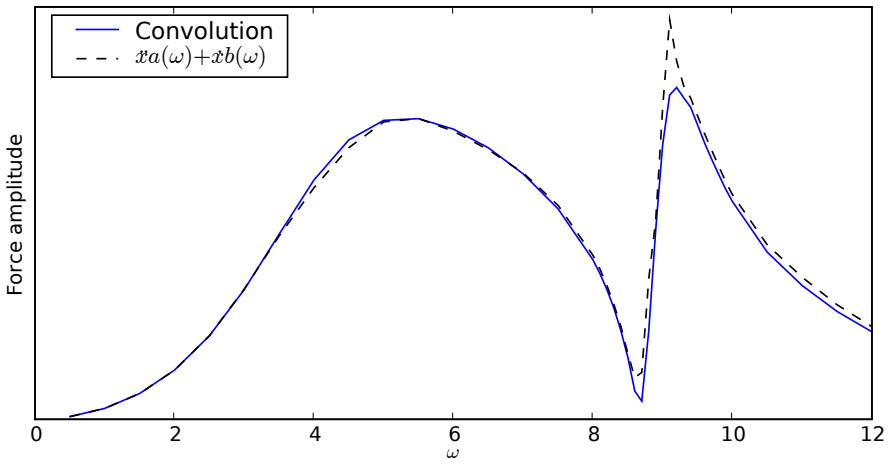


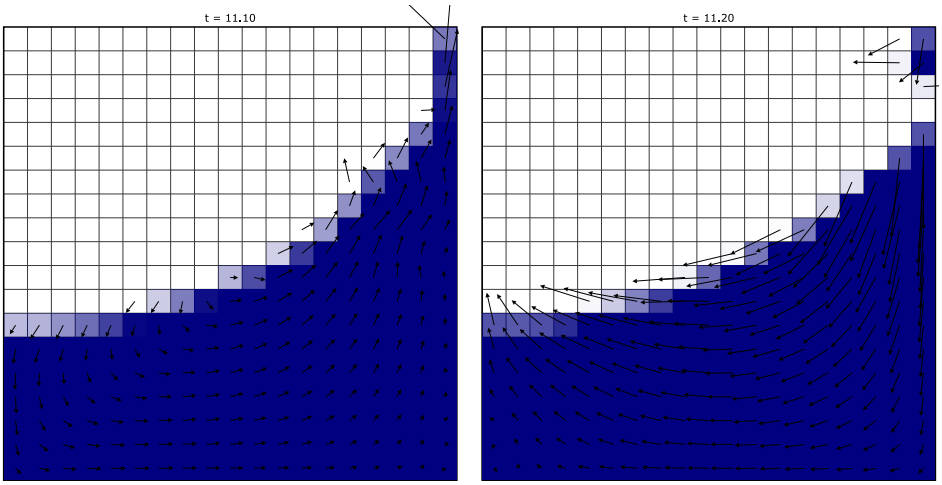
Figure 4.5 – A comparison of the contribution from the convolution term and the equivalent force from (4.11).

between 8.4 and 9.7 rad/s most simulations broke down due to problems with roof impact. An example of this can be seen in figure 4.6

To get results in the frequency range around the first natural sloshing frequency, which is the most interesting, the simulations were run again with an incoming wave amplitude that was five times smaller. In figure 4.7 the excitation forces for this run is plotted. As can be seen, the sloshing force has a lowering effect on the total force up to the first natural frequency of the tank. The sloshing force then changes phase and has the opposite effect for higher frequencies.

Rognebakke (2002) found that cancellation occurred at a frequency that was slightly lower than the first natural frequency of the tank sloshing motion. This is not the case for the simulations with the implemented solver. The solver simulates a ship that is unrestrained, in contrast to the experimental data that by necessity has some restrainment from the test rig. Cancellation of the wave excitation force in figure 4.7 occurs at the first natural frequency. Figure 4.8 shows the simulated motions of the ship hull compared to results from experiments done by Rognebakke. The results from the simulations with the low amplitude waves have been multiplied by five, and seem to coincide fairly well with the results from the simulations where the amplitude was 1.5 cm. This indicates that the coupled system is fairly linear in this range of wave amplitudes, at least outside the area of cancellation.

From the figure it can be seen that the results fit fairly well with the experiments in the range of frequencies below the cancelation frequency. The results are a bit conservative, but this can probably be attributed to the lack of viscous effects in the solution of the external motion and that the



(a) The surface impacts the roof.

(b) The fluid in the upper corner retracts too slowly and a gap forms.

Figure 4.6 – The sloshing solver breaks down when the fluid impacts the roof. The vector field is the velocities while the scalar field in the background is the colour function. From a simulation at $\omega = 8.7\,\mathrm{rad/s}$.

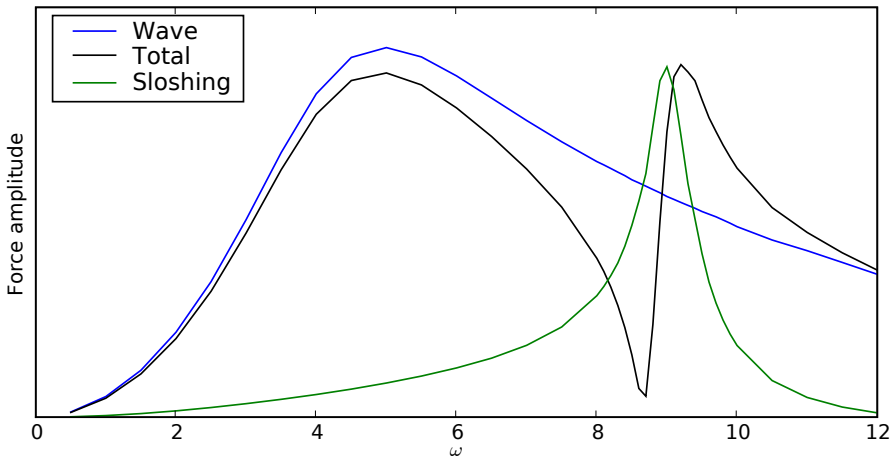


Figure 4.7 – The excitation forces on the coupled system.

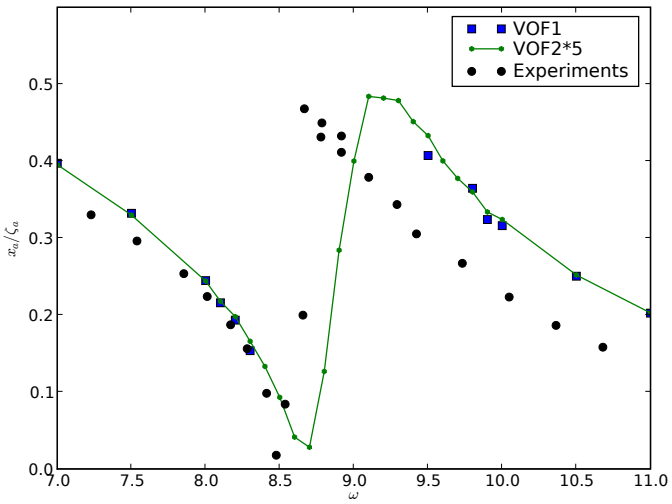


Figure 4.8 – The motion amplitudes of the test section normalised by the incoming wave amplitude. Results for $\zeta_a = 1.5$ cm are denoted VOF1, while the results with a five times lower wave are denoted VOF2. The experimental results are from Rognebakke (2002).

experimental results include a contribution from friction in the bearings of the test rig. The match is not so good for the upper frequencies, but here the effect of friction in the bearings of the test section from the experiments has a large impact. A set of simulations where the bearing friction force is included is written about later in this chapter.

A set of simulations were run with the low amplitude waves where a viscous contribution was added to the left side of the equation of motion (4.4). The viscous contribution is $B_{visc}\dot{x}|\dot{x}|$ where $B_{visc} = \frac{1}{2}\ C_D \rho b\ 2.0$ (following Rognebakke 2002) where b is the breadth of the section, 0.6 m, and C_D is approximated to 3.0 (Faltinsen 1990, Eq. 7.24, low KC flow). The results can be seen in figure 4.9 along with the linear sloshing model from Rognebakke. The figure shows that the viscous damping only contributes in the small range above the first natural sloshing frequency were the amplitudes of motion and velocity of the test section are large. Since the viscous damping is a non linear effect it can be assumed to have a larger effect with the large wave amplitude. Lastly, the transition between small and large motions in the figure can be seen to follow the behaviour of the linear model from Rognebakke (2002) very closely.

To get a complete comparison with the experimental results, all the significant influences in the experiment must be included. For the experiments by Rognebakke this means that the friction force in the bearings of the test rig

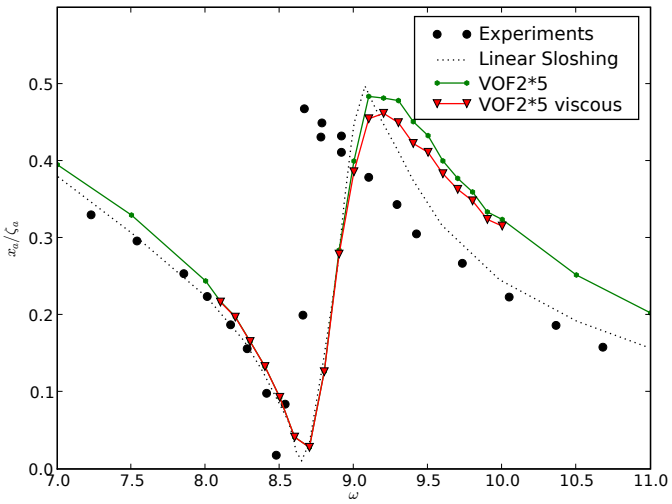


Figure 4.9 – The motion amplitudes of the test section normalised by the incoming wave amplitude. Results for simulations with and without viscous damping of the external flow. The experimental and linear results are from Rognebakke (2002).

must be included. The friction is an approximately constant force of 2.0 N acting against the motion, $F_{bearing} = -2.0 \text{ sign}(\dot{x})$. The results from these simulations can be seen in figure 4.10. As the simulations still break down in the range of frequencies close to the first natural frequency of sloshing for the incoming wave with an 1.5 cm amplitude, no results are given in this range. The effect of the bearing friction force is highly non linear, so no results are given for the low amplitude wave as these simulations showed little or no similarity to the results with the larger wave. The friction force could be scaled down, but the effect of this force on the cancellation frequency is also non linear, so finding the behaviour in the troublesome frequency range and comparing the cancelation frequency to that found by Rognebakke remains impossible with the VOF solver in the present state.

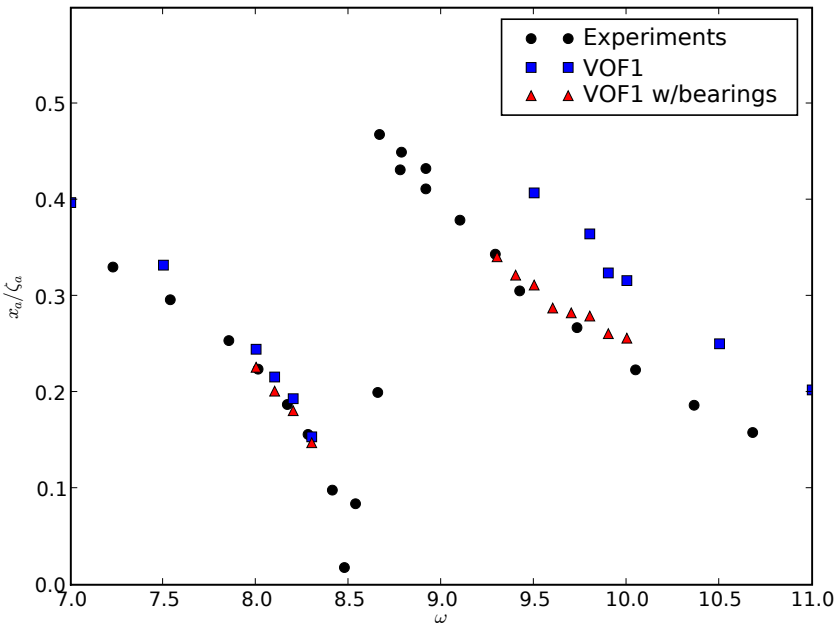


Figure 4.10 – The motion amplitudes of the test section normalised by the incoming wave amplitude. Results for simulations with and without a constant bearing friction force. The experimental results are from Rognebakke (2002).

Chapter 5

Conclusion

How well can coupled ship motions and sloshing be simulated by simple numerical methods? The answer seems to be *rather well*, as long as the free surface in the tank does not hit the roof, at least in this implementation. With a linear equation of motion for the coupled system and no influential non–linear effects in the motion of the fluid in the tank, a low enough incoming wave amplitude can be found that lets the system be simulated at all frequencies. In addition, the results correspond well with experiments for all the simulations that do not crash the solver. The downside of this approach is that the solution is no better or worse than the linear model from Rognebakke (2002), a model that is computationally much less demanding.

The major use case for VOF simulations of sloshing is for simulating non–linear effects and violent sloshing. A beginning of such a use can be seen in the red triangles in figure 4.10. The results above the first natural frequency of sloshing can *not* be simulated well with a linear model, but the VOF solver gives results that corresponds well with the experimental data. The VOF method has great potential, but the implemented solver needs to be refined some more to really reap the benefits of using a fully non–linear method.

The choice of a simple solution algorithm to Navier–Stokes was a good one, given the time allotted for the master thesis. The problems encountered have been much more related to the boundary conditions than to the basic SOLA solution algorithm which has worked well and not too slowly for the non–viscous test cases where resolution of the boundary layer is relatively unimportant. The behaviour of the coupled system can be found in fifteen to twenty minutes per wave frequency.

As a conclusion it can be said that the presented theory and the implemented code shows great promise for simulations in the domain of ships with rectangular tanks that are not very large in the direction normal to the computational xy–plane.

Chapter 6

Recommendations for Further Work

6.1 Extending the Implementation

The pressure and velocity boundary conditions on the free surface in the VOF solver should be improved so that the solver is more robust and can handle more physical cases. Simulation of overturning surfaces should be possible with some work. Spray and mixing may also be possible extensions if the boundary conditions can be made more general.

Roof impact can be studied. This has not been a focus in this thesis, and it has been avoided in the tests of the solver detailed in this report, except those mentioned in section 4.5.2, *Testing the Coupled Motion*. As implemented, roof impact works in only a few cases with relatively mild sloshing, but in many cases the surface will detach a bit to slowly from the roof so that the fluid domain separates into a small, slowly descending part, and a the main part underneath which descends along the wall at the expected speed. This causes the solver to crash and should be fixed.

Other possible extensions are simulating coupled sloshing and roll, and coupled sway, roll and heave with sloshing. Internal obstacles can also be introduced into the tank. If this means that the grid must resolve small obstacles, a non–regular grid can be evaluated. This makes the Navier–Stokes solver a bit harder to implement, but it should not be too difficult as Hirt & Nichols (1981) presents the needed changes. A more involved extension is the extension to 3D. Changes need to be made to the basic SOLA algorithm, but the main difficulties will probably be the VOF flux and boundary condition calculations which get a whole new level of complexity with the introduction of a new dimension. The resulting code will probably also be very slow due to the need for adding more elements, which seriously hampers debugging.

6.2 Using the Implementation

This thesis has mainly been focused on the theory behind and the implementation of the coupled VOF solver. The solver can now, when keeping in mind its limitation, be used for analysis of a range of phenomena from free surface flows to coupled simulations of a ship with an anti rolling tank. Various obstacles can be placed in the tank to dampen the motion and to tune the natural frequency to fit that of a real ship.

The effect on a ship of having a free surface in an internal tank can be compared to having only a rigid mass. The effect of the size and degree of filling of the tank can also be studied. Sloshing can also have an effect on the drift forces on a ship, and this effect can possibly be studied by using the code as it is.

References

Chen, S., Johnson, D. B. & Raad, P. E. (1995), 'Velocity Boundary Conditions for the Simulation of Free Surface Fluid Flow', *Journal of Computational Physics* **116**, 262–276.

Cummins, W. E. (1962), The Impulse Response Function And Ship Motions, Technical Report 1661, David Taylor Model Baisin.

Faltinsen, O. M. (1990), *Sea Loads on Ships and Offshore Structures*, Ocean Technology, Cambridge University Press.

Gerrits, J. (2001), Dynamics of Liquid–Filled Spacecraft, PhD thesis, Rijksuniversiteit Groningen.

Ghia, U., Ghia, K. & Shin, C. T. (1982), 'High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method', *Journal of Computational Physics* **48**, 387–411.

Greenhow, M. (1986), 'High- and low-frequency asymptotic consequences of the Kramers-Kronig relations', *Journal of Engineering Mathematics* **20**, 293–306.

Harlow, F. H. & Welch, J. E. (1965), 'Numerical Calculation of Time–Dependent Viscous Incompressible Flow of Fluid with Free Surface', *Physics of Fluids* **8**(12), 2182–2189.

Hirt, C. W. & Nichols, B. D. (1981), 'Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries', *Journal of Computational Physics* **39**, 201–225.

Hirt, C. W., Nichols, B. D. & Romero, N. C. (1975), SOLA: A Numerical Solution Algorithm for Transient Fluid Flows, Technical Report LA-5852, Los Almos Scientific Laboratory, New Mexico, USA.

Kleefsman, T. (2005), Water Impact Loading on Offshore Structures, PhD thesis, Rijksuniversiteit Groningen.

Kothe, D. B., Mjolsness, R. C. & Torrey, M. D. (1991), Ripple: A computer program for incompressible flows with free surfaces, Technical report, Los Almos Scientific Laboratory, New Mexico, USA.

Kvålsvold, J. (1994), Hydroelastic Modelling of Wetdeck Slamming on Multihull Vessels, PhD thesis, The Norwegian University of Science and Technology.

Langtangen, H. P. (1999), *Computational Partial Differential Equations*, Springer.

Ogilvie, T. F. (1964), Recent Progress Towards the Understanding and Prediction of Ship Motions, *in* 'Proc. 5th Symp. on Naval Hydrodynamics', Bergen, Norway, pp. 3–80.

Price, W. G. & Bishop, R. E. D. (1974), *Probabilistic Theory of Ship Dynamics*, Chapman and Hall.

Rider, W. J. & Kothe, D. B. (1998), 'Reconstructing Volume Tracking', *Journal of Computational Physics* **141**, 112–152.

Rognebakke, O. (2002), Sloshing in rectangular tanks and interaction with ship motions, PhD thesis, The Norwegian University of Science and Technology.

Rudman, M. (1997), 'Volume-Tracking Methods for Interfacial Flow Calculations', *International Journal for Numerical Methods in Fluids* **24**, 671–691.

Solaas, F. (1995), Analytical and Numerical Studies of Sloshing in Tanks, PhD thesis, The Norwegian University of Science and Technology.

White, F. M. (2006), *Viscous Fluid Flow*, 3 edn, McGraw-Hill, New York. International Edition.

Youngs, D. L. (1982), Time–Dependent Multi–Material Flow With Large Fluid Distortions, *in* K. W. Morton & M. J. Baines, eds, 'Numerical methods for Fluid Dynamics', The Institute om Mathematics and its Applications, Academic Press, London, pp. 273–285.

Appendix A

The Implementation Code

The VOF solver code is implemented as a library usable from the Python high level programming language. It is written in a combination of C++ and Python. A simplified class diagram of the VOF solver can be seen in figure A.1. The diagram is drawn according to the Unified Modelling Language (UML) standard. The container boxes are packages while the smaller boxes are classes. Triangle connectors denote specialisation while the diamond shape connectors denote aggregation, classes containing references to other classes. The structure of the coupled motion solver can be seen in figure A.2.

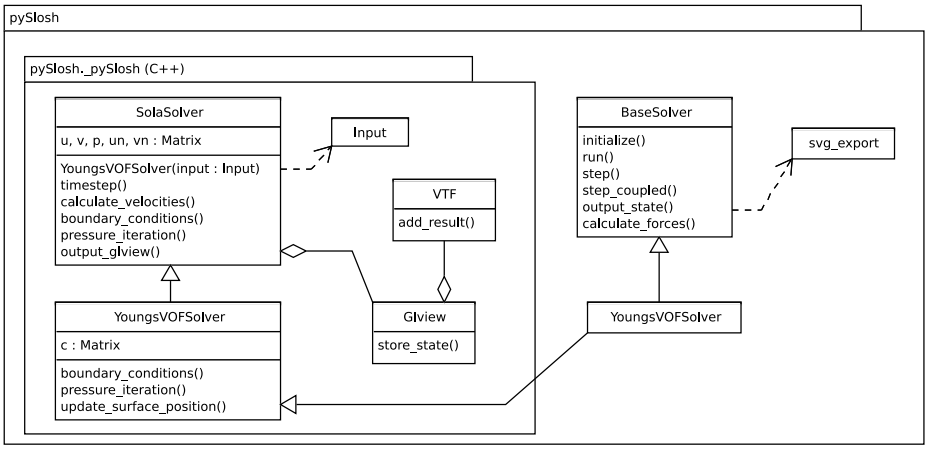


Figure A.1 – A simplified class diagram of pySlosh. The solver inherits the SOLA and VOF algorithm code from the C++ library and the ability to run coupled simulations from BaseSolver written in Python.

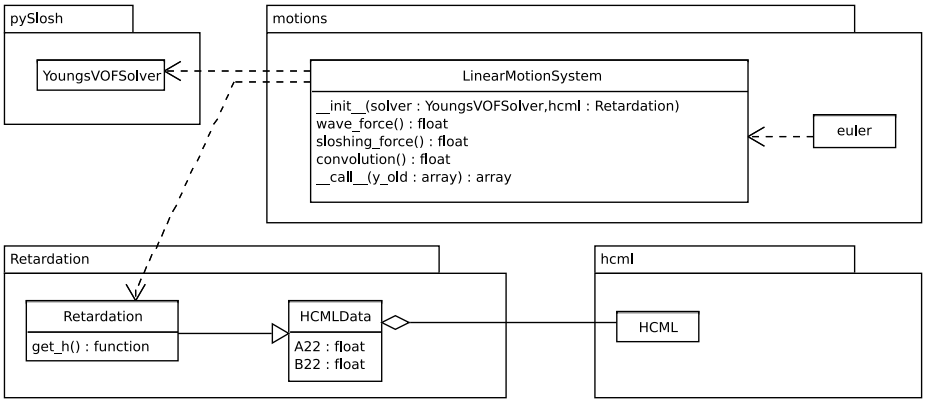


Figure A.2 – A simplified class diagram of the coupled motion solver. The coupled solver uses the the VOF solver and the hydrodynamical coefficients to solve the coupled equations in the time domain with Euler's algorithm.

A.1 Overview of the Source Code

The code is included on a CD (see appendix B) and is also available from `http://launchpad.net/pyslosh` which also has the development history of the code with all changes.

The following files holds the C++ code that implements the speed critical parts of the coupled solver. This is mainly the Navier–Stokes and VOF code with some utility classes.

`cpp/`

Solver classes

`vofslosh.hpp`

`sola_solver.cpp`

`sola_surf_solver.cpp`

`youngs_vof_solver.cpp`

Container class for holding the simulation input parameters

`input.hpp`

`input.cpp`

Utility code

`utils.hpp`

`utils_rotation.hpp`

Code for writing VTF files (on `ASCII` format)

`glview.cpp`

`vtf.hpp`

`vtf.cpp`

The following files holds the C++ code that forms a wrapper layer for Python around the solver code (uses `Boost.Python`)

`cpp/wrapper/`

Main wrapper code

`pySlosh.cpp`

Wrap the solver classes

`solvers.hpp`

`solvers.cpp`

Wrap the `boost::numeric::ublas::matrix` and `vector` classes

`boost_numeric.hpp`

`boost_numeric.cpp`

The following files implements the Python part of the VOF solver. The contents are mainly wrapper code, input and output code and Scalable Vector Graphics export facilities.

`python/pySlosh`

`__init__.py`

`input.py`

`utils.py`

`svg_export.py`

`boost_numeric.py`

`base_solver.py`

The following files make up the example solvers that use the pySlosh library to get the results described in the thesis.

`python/pySlosh/example_solvers`

`__init__.py`

`analytical_velocities_solver.py`

`velocities_from_file_solver.py`

`harmonic_motion_solver.py`

`tilting_solver.py`

`natfreq_solver.py`

`lid_driven_cavity_flow_solver.py`

The following file implements the coupled solver

`python/motions.py`

The following files make up a small library to read HCML files which are used for storing the hydrodynamical coefficients for the coupled analysis. This code was written to experiment with XML based formats for storing structured information and have little relevance to the coupled solver except for reading the input data.

`python/hcml`

- `__init__.py`
- `elements.py`
- `metadata.py`

The files in the following directory are used for driving the simulations. The calculation of the retardation function, plotting and creation of animation are also done by scripts in this directory. An input file to run the coupled simulations in this report is also included. Heavy use is made of the excellent SciPy and NumPy libraries for numerical calculations and Matplotlib for plotting. These libraries help bring Python almost on par with Matlab in features while surpassing it easily for ease of structured coding and general system integration by virtue of being written for the freely available Python programming language.

`python/examples`

Run sloshing and coupled simulations

- `runsimulation.py`
- `runcoupled.py`
- `debugsimulation.gdb`

Calculate the retardation function and plot the results

- `retardation.py`
- `calculate_h22.py`

Generate plots and animations

- `plot_forces.py`
- `animate.py`
- `pylab_utils.py`

Input data for the coupled simulations

- `rognebakke.hcml`
- `reference.hcml`

In addition to these files there are also some unit and integration tests for both the C++ and the Pyhon parts of the code. A build system created for the CMake program is used for Makefile generation, compilation, installation and for running the included tests. Some documentation is also included. This documentation is readable as pure text files, but may also be transformed into for example HTML format due to being written in reStructuredText format. These files on HTML format can be found on `http://tormod.landet.net/code/sloshing/` along with the API documentation of the libraries and links to the source code.

All the code has been tested with C++ compiler GCC version 4.1.3 and 4.2.3 and Python version 2.5 on Linux. With some work on the build system everything should compile, run and install on most modern operating systems including Microsoft Windows.

Appendix B

An Overview of the Contents of the Attached CD

The included CD has the following contents:

- This report as a PDF file
- The source code described in appendix A
- An animation of a coupled simulation where the sloshing is so violent that the VOF solver eventually breaks down.
- An animation of the same time series as in figure 3.13(b) which is the simulation of the second natural sloshing mode of motion.