Numerical simulation of the side launching of a ship

Peter de Jong

Department of
Mathematics

RuG

Master's Thesis

Numerical simulation of the side launching of a ship

Peter de Jong

Supervisor:
Prof.dr. A.E.P. Veldman
Department of Mathematics
University of Groningen
P.O. Box 800
9700 AV Groningen

March 2004

Contents

Chapter 1

Introduction

Already for long ships have been side launched off slopes made of wood or steel into canals in the Netherlands. This has proved to be a successful method of launching ships in confined waters. The risks involved (like capsizing or hitting the canal bed with the waterside chine) were minimized by the centuries of experience and the fact that shipyards used to build rather the same type of ships.

However, nowadays there is a trend of moving the production of uniform ships to cheap labor countries (for example Eastern Europe of Asia) for evident reasons. This movement threatens the continuation of Dutch shipyards. A number of shipyards have chosen to change their production to custom made ships. But now an uncertainty arose among ship builders about using the same old methods of launching the new custom made ships. There was a need for something that can predict the outcome of a side launch, so that ship builders could alter the parameters of the launch if needed in order to get a successful launch.

Figure 1.1: Launch of the Suryawati in 2002 at the shipyard Bodewes-Volharding

The objective of this project is to create and validate a program which can predict the path of a rectangular ship being side launched in two dimensions on the basis of real life parameters (the geometry and mechanical characteristics of the slopes, canal and ship like

the mass, moment of inertia and friction coefficients). A workable version of a program which can predict the path in three dimension will be presented without validation (a future project will perform this). The mathematical model incorporated into the program consists of a dry part (describing the situation, forces, momenta on the quay) and a wet part (describing the motion and reaction of the water). The dry part is based on the research of Chr. M. van Hooren [8]. Van Hooren compared his theory with a large number of model experiments, which were designed by J. Versluis [10]. The water reaction forces and momenta in his mathematical model were calculated using a simplified hydrodynamic model from Ir. J. H. Vugts [11]. Nowadays, with the increase in processor speeds and memory sizes and improvement of the numerical algorithms, the hydrodynamic part of the mathematical model can be computed with more accuracy using software like ComFlo, which is able to simulate two and three dimensional fluid flows with free surfaces. It is based on a finite volume method. In order to let ComFlo predict the path of a ship being side launched, a modular shell (or rind) is created around the original ComFlo code. By evaluating forces and momenta and using a time integration method the computer program is able to calculate a ship's path in full interaction with the water.

Figure 1.2: End of the launch of the Suryawati in 2002 at the shipyard Bodewes-Volharding

During the project a number of launches were recorded for validation purposes of future versions of this software, which will be capable of simulating the side launch of a lifelike ship (not only the rectangular kind). These recordings will be processed in future projects. A scenario for the recordings of these launches is included in this report.

This report consists of five chapters and two appendices. First the mathematical and numerical model used in the numerical simulations are presented in chapters two and three. After that the results of the validation of this extended version of ComFlo is presented in chapter four. Finally conclusions are drawn in chapter five. A program description will be presented in the first appendix, followed by the scenario for the recording of a side launch.

Chapter 2

Mathematical model

The mathematical model, used in this study, describes the movement of a vessel being launched from a quay into a canal. This model consists of a dry part (ie. normal and friction forces from the quay) and a wet part (water reaction forces). The dry part of the model is based on the mathematical model [8] by Chr. M. van Hooren, which is an improvement of Ju. S. Jakovlev's model [5]. Originally Van Hooren's mathematical model already contains equations for the water reaction forces based on the work of Ir. J. H. Vugts [11], but we will discard this. Instead the aquatic part of the model will be formed by the Navier-Stokes equations.

2.1 Dry model

The launch of the ship is divided into three phases. In phase 1 the ship is still located on the slope/quay. Phase 2 starts when the ship starts to tilt but has contact with the quay. In phase 3 the ship no longer has any contact with the quay. Note that in any phase the vessel can plunge in the water.

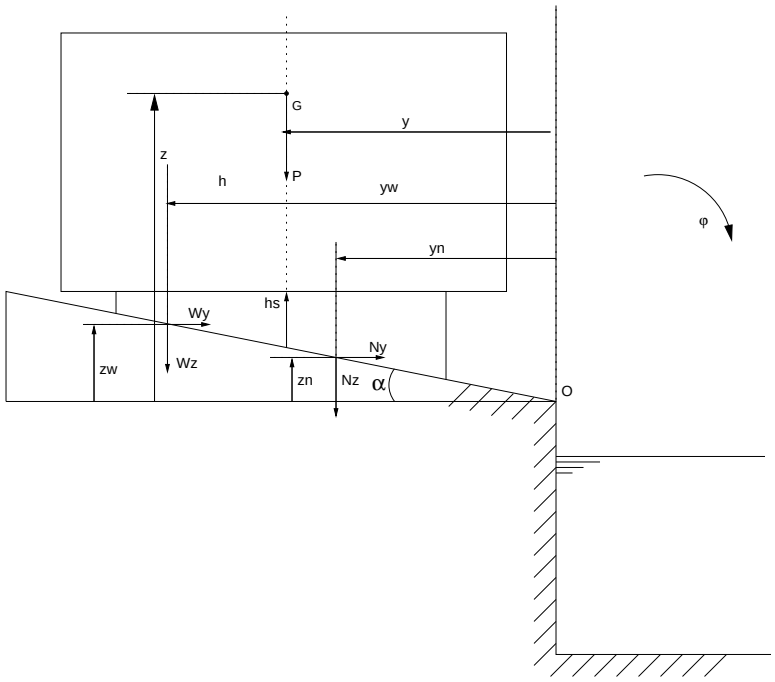

Figure 2.1: First phase

2.1.1 Table of used variables

parameter	description
m	Ship's mass
(x,y,z)	Position of the center of gravity (G) in the three coordinate directions
P	Gravitational force working on G
W	Friction force working on the bottom of the sleighs/top of the slope
u	Friction coefficient used for determining W
N	Normal force working on the bottom of the sleighs/top of the slope
R	Water force working on the ship
B	Wind resistant force
M	Torque around the coordinate axis through G
I	Moment of inertia
h_s	Height sleighs w.r.t. midship
ϕ	Rotation around x coordinate direction through G (roll)
θ	Rotation around y coordinate direction through G (pitch)
ψ	Rotation around z coordinate direction through G (yaw)
α	Angle of inclination of the slope
h	height G w.r.t. the keel

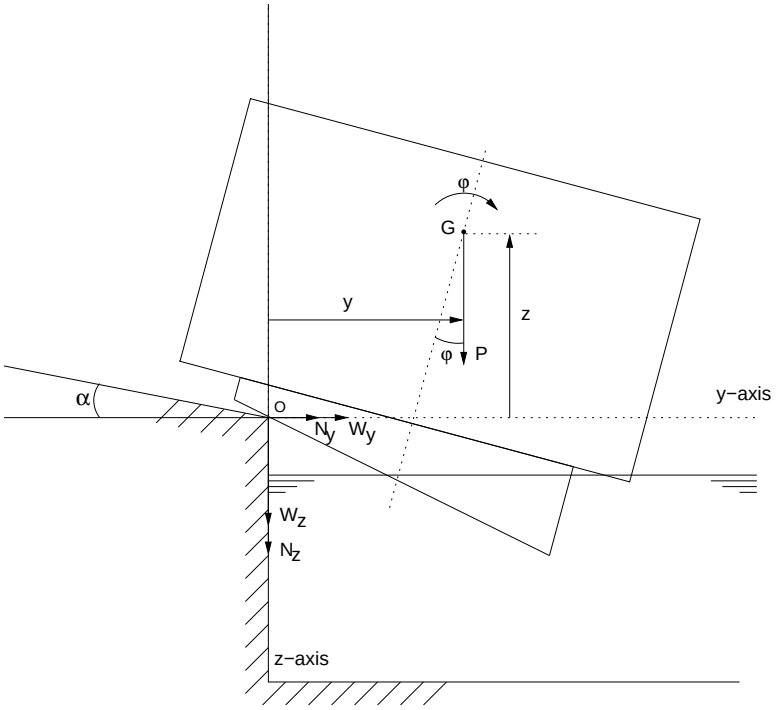

Figure 2.2: Second phase

2.1.2 Equations of motion

We will now define the equations of motion for the situations depicted by figure 2.1 and figure 2.2. We define the coordinate origin at the edge of the quay and define the x coordinate axis along it's edge. The y and z coordinate directions are defined in such a manner that the center of gravity G will be located in the plane defined by these coordinate directions. The equations of motion for this are:

$$mx'' = P_x + W_x + N_x + R_x + B_x \tag{2.1}$$
$$my'' = P_y + W_y + N_y + R_y + B_y \tag{2.2}$$
$$mz'' = P_z + W_z + N_z + R_z + B_z \tag{2.3}$$
$$I_x\phi'' = M_{W_x} + M_{N_x} + M_{R_x} + M_{B_x} \tag{2.4}$$
$$I_y\theta'' = M_{W_y} + M_{N_y} + M_{R_y} + M_{B_y} \tag{2.5}$$
$$I_z\psi'' = M_{W_z} + M_{N_z} + M_{R_z} + M_{B_z} \tag{2.6}$$

Because the direction of the force of gravity is parallel to the z coordinate direction and the center of gravity G lies in the y-z plane, we simplify the equations with:

$$P_x = P_y = 0 \tag{2.7}$$
$$P = P_z \tag{2.8}$$
$$N_x = W_x = 0 \tag{2.9}$$

We neglect the influence of the wind:

$$B_x = B_y = B_z = M_{B_x} = M_{B_y} = M_{B_z} = 0 \tag{2.10}$$

After applying these simplifications the equations of motion are:

$$\begin{aligned} mx'' &= R_x && (2.11)\\ my'' &= W_y + N_y + R_y && (2.12)\\ mz'' &= P + W_z + N_z + R_z && (2.13)\\ I_x\phi'' &= M_{W_x} + M_{N_x} + M_{R_x} && (2.14)\\ I_y\theta'' &= M_{W_y} + M_{N_y} + M_{R_y} && (2.15)\\ I_z\psi'' &= M_{W_z} + M_{N_z} + M_{R_z} && (2.16) \end{aligned}$$

We assume that during the launch the following relation exists between normal force N and friction force W, where u is the friction coefficient

$$W = uN \tag{2.17}$$

With this we can define the normal force and friction forces in the y and z coordinate directions:

$$\begin{aligned} W_y &= uN\cos(\phi+\alpha) && (2.18)\\ W_z &= uN\sin(\phi+\alpha) && (2.19)\\ N_y &= N\sin(\phi+\alpha) && (2.20)\\ N_z &= N\cos(\phi+\alpha) && (2.21) \end{aligned}$$

And we rewrite the equations of motion:

$$\begin{aligned} mx'' &= R_x && (2.22)\\ my'' &= -uN\cos(\phi+\alpha) + N\sin(\phi+\alpha) + R_y && (2.23)\\ mz'' &= P - uN\sin(\phi+\alpha) - N\cos(\phi+\alpha) + R_z && (2.24)\\ I_x\phi'' &= -uN\cos(\phi+\alpha)(z-z_w) + uN\sin(\phi+\alpha)(y-y_w) + \\ &\quad N\sin(\phi+\alpha)(z-z_n) + N\cos(\phi+\alpha)(y-y_n) + M_{R_x} && (2.25)\\ I_y\theta'' &= uN\sin(\phi+\alpha)(x-x_w) + N\cos(\phi+\alpha)(x-x_n) + M_{R_x} && (2.26)\\ I_z\psi'' &= uN\cos(\phi+\alpha)(x-x_w) + N\sin(\phi+\alpha)(x-x_n) + M_{R_z} && (2.27) \end{aligned}$$

We assume that:

- the reaction force from the water in the x coordinate direction is small compared to the reaction force in the other two coordinate directions. Therefore $R_x = 0$.
- the normal force N and the friction force W work in the same plane as G. Therefore $x_w = x_n = 0$.
- the ship is on an even keel (it has a symmetrical weight distribution). Therefore $M_{r_y} = M_{r_z} = 0$.

- during the first phase the normal force and friction force are applied in the point straight under G: $z_w = z_n = y\tan(\alpha)$ and $y_w = y_n = y$. After the ship starts to tilt (phase 2 and 3): $z_w = z_n = y_w = y_n = 0$.

To furthermore simplify matters we define:

$$a = \sin(\phi+\alpha) - u\cos(\phi+\alpha) \tag{2.28}$$
$$b = \cos(\phi+\alpha) + u\sin(\phi+\alpha) \tag{2.29}$$

As a result of the above assumptions and simplifications we can rewrite the equations of motion for the first phase:

$$my'' = Na + R_y \tag{2.30}$$
$$mz'' = P - Nb + R_z \tag{2.31}$$
$$I_x\phi'' = Na(z - y\tan(\alpha)) + M_{R_x} \tag{2.32}$$

with $N = P\cos(\alpha)$ and unknown variables y,z and ϕ. The equations of motion for the second phase are:

$$my'' = Na + R_y \tag{2.33}$$
$$mz'' = P - Nb + R_z \tag{2.34}$$
$$I_x\phi'' = N(az + by) + M_{R_x} \tag{2.35}$$

with unknown variables y,z,ϕ and N. And the equations of motion for the third phase ($N = 0$) are:

$$my'' = R_y \tag{2.36}$$
$$mz'' = P + R_z \tag{2.37}$$
$$I_x\phi'' = M_{R_x} \tag{2.38}$$

with unknown variables y,z and ϕ. Systems of equations 2.30-2.32 and 2.36-2.38 can now be solved. But to solve the system of equations 2.33 to 2.35, mhich consists of 3 equations and 4 unknown variables, we need to introduce an extra equation. For this we define a relation between the horizontal and vertical position of the ship while it still has contact with the quay:

$$z = y\tan(\phi+\alpha) - \frac{h+h_s}{\cos(\phi)} \tag{2.39}$$

Before we can add this relation to our system of equation, we will first differentiate it twice.

$$z'' = y''\tan(\phi+\alpha) + \phi''e + d \tag{2.40}$$

with e and d defined as:

$$d = \frac{y}{\cos^2(\phi+\alpha)} - (h+h_s)\frac{\sin(\phi)}{\cos^2(\phi)} \tag{2.41}$$
$$e = -(h+h_s)\phi'^2\frac{1+\sin^2(\phi)}{\cos^3(\phi)} + 2\phi'\left(\frac{y'}{\tan(\phi+\alpha)} + y\phi'\right)\frac{\sin(\phi+\alpha)}{\cos^3(\phi+\alpha)} \tag{2.42}$$

With this extra equation we can solve N from the equations 2.33-2.35 and :

$$N = -\frac{R_y \tan(\phi+\alpha) + \frac{MR_x}{I_x} me + md - P - R_z}{b + a\tan(\phi+\alpha) + \frac{az+by}{I_x} me} \tag{2.43}$$

We are now able to solve the equations of motion for all three phases (2.30-2.32,2.33-2.35 and 2.36-2.38).

2.2 Wet model

In the previous section we looked only at the dry physics in the mathematical model and assumed that the reaction forces and the torques from the water were known. We will now describe the wet mathematical model, from which these forces and torques can be determined.

2.2.1 Navier-Stokes equations

We will describe the motion of the fluid through the domain by the Navier-Stokes equations for an incompressible and viscous fluid. Let $\mathbf{u}$ denote the velocity vector, ρ denote the density (which will eventually be normalized to unity ($\rho = 1$)), p denote the pressure and F denote an external body force (i.e. the force of gravity). The Navier-Stokes equations are:

$$\nabla \mathbf{u} = 0 \tag{2.44}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu(\nabla\cdot\nabla)\mathbf{u} + F \tag{2.45}$$

with equation 2.44 as the equation for the conservation of mass and equation 2.45 as the equation for the conservation of momentum.

2.2.2 Conditions at the solid boundary

The walls of our domain, quays and canal do not let any liquids through. We will define these walls as free slip. The conditions at the solid boundary are:

$$u_n = 0 \tag{2.46}$$

$$\frac{\partial u_t}{\partial \mathbf{n}} = 0 \tag{2.47}$$

where $\mathbf{n}$ denotes the normal direction, $u_t = \mathbf{u}\cdot\mathbf{t}$ denotes the tangential velocity and $u_n = \mathbf{n}\cdot\mathbf{u}$ denotes the velocity normal to the wall. Equation 2.46 describes the free slip condition and equation 2.47 describes that the fluid cannot move through the wall.

2.2.3 Conditions at the free surface

We need to define an extra set of boundary conditions for the free surface. These conditions are:

$$-p + 2\mu\frac{\partial u_n}{\partial n} = -p_0 + 2\sigma H \tag{2.48}$$

$$\mu(\frac{\partial u_n}{\partial t} + \frac{\partial u_t}{\partial n}) = 0 \tag{2.49}$$

where μ denotes the dynamic viscosity, p_0 denotes the atmospheric pressure, σ the surface tension and $2H$ represents the total curvature of the surface.

2.2.4 Forces and moments

As was mentioned in the previous section, the fluid induces a certain force and torque on the ship. This force normally consist of two parts, namely the pressure force and the shear force. In this project we will neglect the shear force, because it's much smaller. The pressure force can be determined as the following integral:

$$R = \int_S p\mathbf{n}dS \tag{2.50}$$

The torque, caused by the forces, w.r.t. the center of gravity of the ship can be calculated as follows:

$$M_R = \int_S p(\mathbf{r} \times \mathbf{n})dS \tag{2.51}$$

where $\mathbf{r}$ is the distance from the point on which the force is exerted to the center of gravity.

Chapter 3

Numerical model

In this chapter the mathematical model presented in the previous chapter will be discretized to obtain a numerical model, with which a computer program can be made. Section 1-6 from this chapter originate from the master's thesis of Geert Fekken [3].

3.1 Description of geometry and free surface

The first thing to do is to lay a Cartesian (rectilinear) grid over the three dimensional domain Ω. The discretization will be done on a totally staggered grid, which means that the pressure will be set in the cell centers, and the velocity components in the middle of the cell faces between two cells (figure 3.1). Like all figures of geometries in this report, this is a 2-dimensional example. Extension to 3D is straightforward.

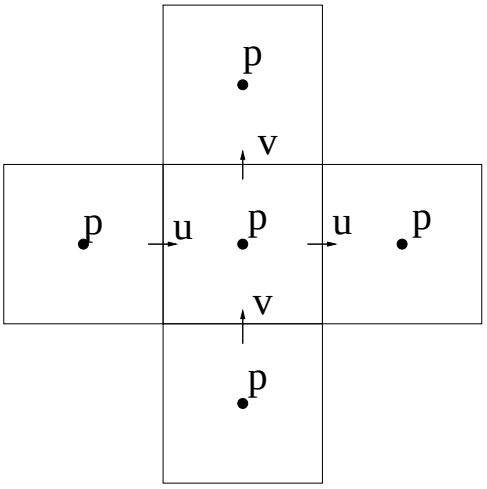

Figure 3.1: Location of the pressure and velocity components

In general the shape of the domain may be complex, so the grid cells will not fit exactly in Ω, but they will run through the boundaries in several ways. Also the free surface can admit different shapes, with an extra complexity since the free surface is time-dependent.

To handle these problems we need a method to describe the geometry and the free surface. The method used in ComFlo will be discussed now.

3.1.1 Apertures

An indicator function is used in the form of so-called *apertures*, they will be divided into two classes:

1. volume apertures

   In every cell, the geometry aperture F_b defines the fraction of the cell which is contained in Ω, in other words the fraction where fluid is able to flow. The (time-dependent) fluid aperture F_s defines the fraction of the cell which is occupied by fluid. Of course $0 \leq F_s \leq F_b \leq 1$.

2. edge apertures

   The edge apertures A_x, A_y, A_z define the fraction of a cell surface which is contained in Ω, so A_x indicates the fraction of the cell surface through which fluid is able to flow in x-direction, A_y in y-direction and A_z in z-direction.

It will be clear that, using geometry and edge apertures, arbitrary forms of Ω can be handled. Figure 3.2 shows a 2-dimensional example of a grid cell using apertures.

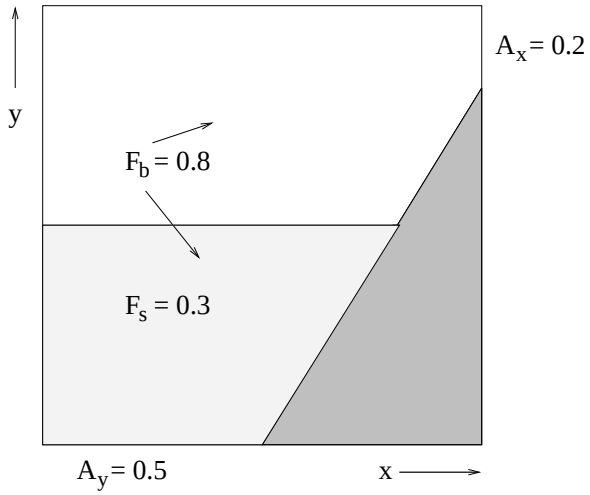

Figure 3.2: Example of a grid cell with geometry and fluid apertures

3.1.2 Labels

After calculating the apertures, every cell will be given a *cell label*, to make distinction between the boundary, the fluid and the air, and because the pressure is treated different near the wall and the free surface. As noted before, the free surface is time-dependent, therefore two classes of labeling are introduced:

1. geometry cell labels

   This labeling class is time-independent, consisting of three different labels:
   - **F**-cells: All cells with $F_b \geq \frac{1}{2}$
   - **B**-cells: All cells adjacent to an **F**-cell
   - **X**-cells: All remaining cells

2. free-surface cell labels

   Free-surface labels are time-dependent and they are a subdivision of the **F**-cells:
   - *E*-cells: All cells with $F_s = 0$
   - *S*-cells: All cells adjacent to an *E*-cell
   - *F*-cells: All remaining **F**-cells

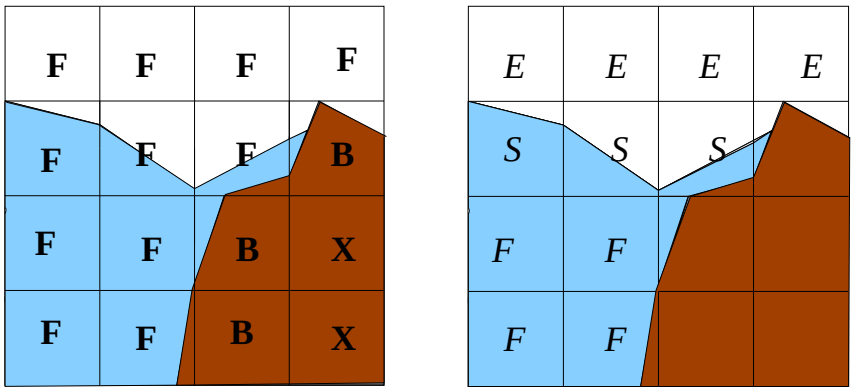

Figure 3.3: Example of geometry cell labeling (left) and free-surface cell labeling (right)

For the treatment of the velocity, the velocities between cells have to be labeled, too. So we introduce *velocity labels*, which, like the cell labels, have to be subdivided in a time-dependent and a time-independent class:

1. geometry velocity labels

   These (time-independent) labels are a combination of the labels of the geometry where the velocities lie in between. Five combinations are possible: **FF, FB, BB, BX** and **XX.**

2. free-surface velocity labels

   These labels are time-dependent and they are a combination of the labels of the free surface. The following combinations are possible: *FF, FS, SS, SE, EE, FB, SB* and *EB.*

Further, there is one more class of labeling, namely inflow and outflow labels, resp. **I**- and **O**-cells. They are just a specific subset of the **B**-cells.

3.2 Discretization of the Navier-Stokes equations

When all cells and velocities are labeled, the Navier-Stokes equations can be discretized in time and in space. First the Navier-Stokes equations are written more simplified as:

$$\nabla \cdot \mathbf{u} = 0 \tag{3.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla p = \mathbf{R} \tag{3.2}$$

Here $\frac{p}{\rho}$ is replaced by p (ρ is normalized to 1) and $\mathbf{R} = \nu\Delta\mathbf{u} - (\mathbf{u}\cdot\nabla)\mathbf{u} + \mathbf{F}$, containing all convective, diffusive and body forces.

Discretization in time

The explicit first order Forward Euler method is used:

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{3.3}$$

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\delta t} + \nabla p^{n+1} = \mathbf{R}^n \tag{3.4}$$

Here $n+1$ and n denote the new and old time level respectively, and δt is the time step. Equation (3.3) and the pressure in (3.4) are treated on the new time level, to make sure the new $\mathbf{u}$ is divergence free.

Discretization in space

The spatial discretization can be explained using the scheme in figure 3.4:

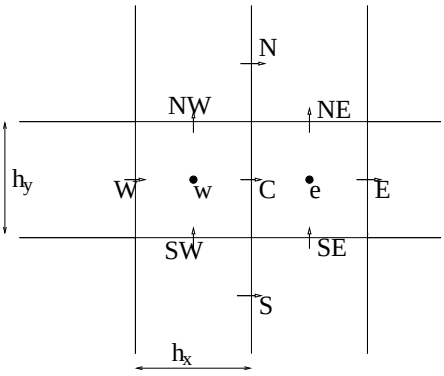

Figure 3.4: Discretization scheme

Equation (3.3) is applied in the centers of the cells and a central discretization is used. In the cell with center w the discretized equation becomes:

$$\frac{u_C^{n+1} - u_W^{n+1}}{h_x} + \frac{v_{NW}^{n+1} - v_{ZW}^{n+1}}{h_y} = 0 \tag{3.5}$$

The momentum equation (3.4) is applied in the centers of the cell faces, for instance the discretization in point C becomes:

$$\frac{u_C^{n+1} - u_C^n}{\delta t} + \frac{p_o^{n+1} - p_w^{n+1}}{h_x} = R_C^n \tag{3.6}$$

The diffusive terms in R_C^n are discretized centrally, and for the convective terms upwind or central discretization are possible. For wildly moving fluids mostly an upwind discretization is used, since central discretization may cause stability problems (see [9] section 2.4).

Discretization near the free surface

Near the free surface besides *F*-cells, *S*-cells and *E*-cells appear. In *E*-cells the pressure is set to the atmospheric value p_0. In *S*-cells the pressure is determined by linear interpolation between the pressure in *F*-cells and the free surface. The pressure p_F in *F*-cells is obtained from the pressure Poisson equation which is handled in the next section. The pressure at the free surface p_f is obtained from equation (2.48), where the term $2\mu\frac{\partial u_n}{\partial n}$ is neglected. So $p_f = p_0 - 2\gamma H$. The pressure p_S now becomes

$$p_S = \eta p_f + (1-\eta)p_F. \tag{3.7}$$

Here $\eta = \frac{h}{d}$ (figure 3.5).

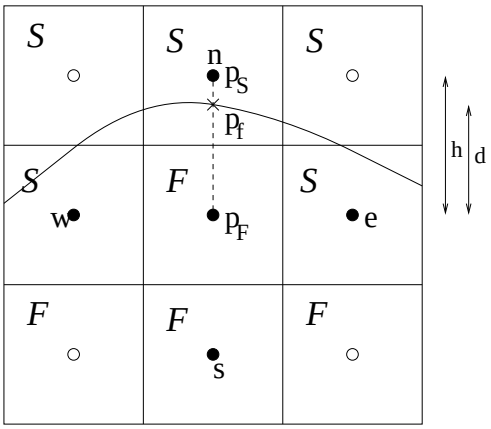

Figure 3.5: Pressure interpolation in *S*-cells

For the velocities in equation (2.47) there are a number of possibilities. The velocities *FF, FS* and *SS* are obtained solving the momentum equations. But when discretizing derivatives *SE*- and *EE*- velocities are needed. How these velocities are obtained is treated in [4] page 13-16.

In- and outflow discretization

As noted in section 3.1.2 in- and outflow cells are a specific subset of the **B**-cells.

- Inflow
  The velocity between an **I**-cell and an **F**-cell gets a prescribed value and is labeled as **FI**.

- Outflow
  In an **O**-cell two velocities have to be labeled: **FO** and **OX**. The **FO**-velocity is computed from the momentum equations and then the **OX**-velocity is set equal to the **FO**-velocity, to satisfy the condition $\frac{\partial \mathbf{u}}{\partial n} = 0$.

Figure 3.6: In- and outflow cells

3.3 The pressure Poisson equation

The pressure p^{n+1} in (3.4) has to be determined in such a way, that equation (3.3) holds. This can be attained by substituting (3.4) into (3.3), resulting in the following equation:

$$\Delta p^{n+1} = \nabla \cdot (\frac{\mathbf{u}^n}{\delta t} + \mathbf{R}^n) \tag{3.8}$$

This equation is known as the Poisson equation for the pressure. No boundary conditions are available for this equation, since they only involve the velocity $\mathbf{u}$. Therefore we first discretize the equations and substitute the boundary conditions, and after that we substitute these discretized equations to create the discretized Poisson equation. It will follow that no more boundary conditions are required now (see [9] section 4.4).

We will now introduce the following notation for the discretized equations: D_h denotes the discrete divergence operator, G_h the discrete gradient operator and R_h is the discrete version of R. Further the divergence operator is divided in an operator on **F**-cells, D_h^F, and an operator on **B**-cells, D_h^B, so that $D_h = D_h^F + D_h^B$.

Because we do not know the velocities at the boundary at the new time step we use $\mathbf{u}^{n+1} = \mathbf{u}^n$ at the boundary. Now the discretized equations are

$$D_h^F \mathbf{u}^{n+1} = -D_h^B \mathbf{u}^n \tag{3.9}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \delta t \mathbf{R}_h^n - \delta t G_h p^{n+1} \tag{3.10}$$

and the discretized Poisson equation for the pressure becomes

$$D_h^F G_h p^{n+1} = D_h^F (\frac{\mathbf{u}^n}{\delta t} + \mathbf{R}_h^n) + D_h^B (\frac{\mathbf{u}^n}{\delta t}) \tag{3.11}$$

Solving the Poisson equation

The discrete operator $D_h^F G_h$ in (3.11) consists of a central coefficient C_p and six coefficients C_n, C_s, C_w, C_e, C_u and C_d, related to the six neighboring cells. When we denote the right hand side of (3.11) by f_p, the Poisson equation can be written as:

$$C_p p_p + C_n p_n + C_s p_s + C_e p_e + C_w p_w + C_u p_u + C_d p_d = f_p \tag{3.12}$$

This equation is easily combined with the pressure condition (3.7) for p_S. Considering figure 3.5, $p_S = p_n$. The Poisson equation in the F-cell becomes: $(C_p + (1-\eta)C_n)p_p + C_e p_e + C_w p_w + C_s p_s = f_p - \eta C_n p_f$.

To keep the program readable, it should be handy when the Poisson equation also holds for *E-, X-, B-* and *S*-cells. In *S*-cells the Poisson equation is again combined with the pressure p_S. Looking at figure 3.5, in the *S*-cell $p_p = p_S$ and $p_s = p_F$. To acquire equation (3.7) we have to set $C_p = 1$, $C_s = \eta - 1$, $f_p = \eta p_f$, while all other coefficients are taken zero. In *E-, X-* and *B*-cells the coefficient C_p is set to one, and all other are taken zero. Here f_p is set to p_0, so in these cells the Poisson equation yields $p_p = p_0$, the atmospheric value.

The Poisson equation is solved by SOR-iteration (Successive Over Relaxation), which has some advantages:

- simple implementation, immediately using every new value.
- easy vectorization and parallelization, using a **Red-Black** ordering of the cells
- rapid convergence, using an automatically adjusted relaxation parameter ω [2].

The SOR-iteration can be written as

$$p_p^{n+1^{k+1}} = (1-\omega)p_p^{n+1^k} + \tag{3.13}$$

$$\frac{\omega}{C_p}(-C_n p_n^{n+1^k} - C_e p_e^{n+1^k} - C_u p_u^{n+1^k} - \tag{3.14}$$

$$C_s p_s^{n+1^{k+1}} - C_w p_w^{n+1^{k+1}} - C_d p_d^{n+1^{k+1}} + f_p) \tag{3.15}$$

When the SOR-iteration has finished the pressure in every cell is known at the new time step. The new velocities can now be computed from $\mathbf{u}^{n+1} = \mathbf{u}^n + \delta t(-\nabla p^{n+1} + \mathbf{R}_h^n)$.

3.4 Free surface displacement

When the velocities at the new time step are known, the free surface can be displaced. The sequence of actions that have to be done to achieve this are:

1. *compute fluxes between cells*
   The fluxes between cells are computed by velocity times the area of the cell, taking into account the edge apertures.
2. *compute new fluid apertures F_s*
   Using the fluxes between the cells, the new F_s can be computed
3. *adjust free-surface labeling*
   When the new fluid apertures are known the free-surface labeling can be adjusted according to the conditions in section 3.1.2.

This algorithm is called the Donor-Acceptor algorithm, which means that fluid is transported from a donor cell to an acceptor cell. A few things have to be taken into account: A donor cell cannot loose more fluid than it contains and an acceptor cell cannot receive

more fluid than the amount of flow space that is available in the cell. Further, in S-cells the fluid has to be tamped towards F-cells to prevent the creation of artificial holes; this is accomplished by making use of a local height function (see [6]).

3.5 The CFL-number

One can imagine that when the fluid is moving very wildly, the time step has to be smaller than when the fluid is moving very calm. It would be useful to adjust the time step to these changes, to achieve an improvement in the computation time. Therefore the Courant-Friedrichs-Levy number (CFL-number) is introduced:

$$CFL = \frac{|u|\delta t}{h_x} + \frac{|v|\delta t}{h_y} + \frac{|w|\delta t}{h_z} \tag{3.16}$$

Here h_x, h_y and h_z denote the distances between the cell centers in x-,y- and z-direction. The condition to keep the computation stable, which can be proved by Fourier analysis (see [9]), turns out to be $CFL \leq 1$. This means that the fluid is transported over no more than one cell in one time step, which corresponds with our intuitive approach of stability. In ComFlo CFL-number is determined, and with respect to this number the time step is adjusted: The time step is immediately halved when the CFL-number becomes larger than a certain constant $C_1 < 1$, and the time step is doubled when the CFL-number is smaller than another constant C_2 which is small enough to be sure the time step can be doubled.

3.6 Computation of forces

For many applications the possibility to compute forces is useful. In ComFlo it is possible to compute the forces by integration of pressure, using the apertures. Considering the grid cell on the next page, the force $\mathbf{F} = (F_x, F_y, F_z)$ is computed by pressure times area. For instance $F_x = \mathbf{F} \cos\alpha \, ds = p \cos\alpha \, ds = p(1 - A_x)dy$.

3.7 Time integration

In order to solve the equations of motion in the dry part of the mathematical model, we need to perform a double time integration. For this the fourth order Runge-Kutta method is used.

Fourth-order Runge-Kutta Method

The 4th order Runge-Kutta method [1] [7] extends the idea of the mid-point-method, in which a simple Euler step is made to the mid-point of the interval $[t_n, t_n + \Delta t]$, after which $f(x_n + 1, t_{n+1/2})$ is evaluated and used in order to jump from t_n to t_{n+1}. In the 4th order Runge-Kutta method we first jump 1/4th of the way first, then we will jump half-way (just

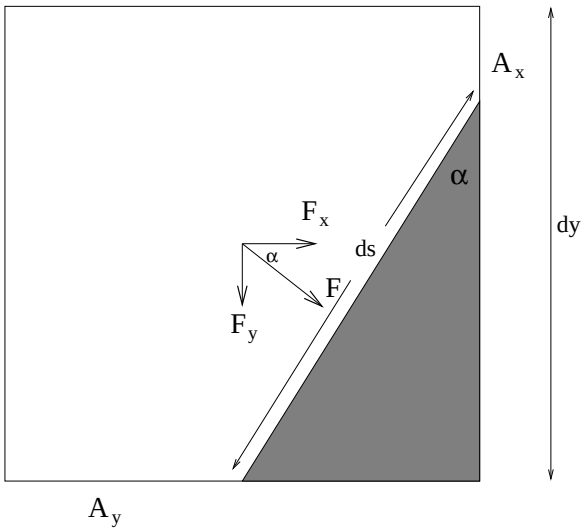

Figure 3.7: Force computation using apertures

like the midpoint method), after that we jump 3/4th of the way and at last we jump all the way:

$$V_1 = f(x_n, t_n) \tag{3.17}$$

$$V_2 = f(x_n + \frac{1}{2}\Delta t V_1, t_n + \Delta t/2) \tag{3.18}$$

$$V_3 = f(x_n + \frac{1}{2}\Delta t V_2, t_n + \Delta t/2) \tag{3.19}$$

$$V_4 = f(x_n + \Delta t V_3, t_n + \Delta t) \tag{3.20}$$

$$x_{n+1} = x_n + \frac{\Delta t}{6}(V_1 + 2V_2 + 2V_3 + V_4) \tag{3.21}$$

Adaptive step size control

We will further enhance the 4th order Runge-Kutta method with adaptive step size control. We will will do this by "step doubling", which enables us to determine the accuracy of the numerical integration and adjust the step size to maintain the requested accuracy. We will take the step twice, once as a full step, leading to $x_1(t + \Delta t)$ and then as two half steps, leading to $x_2(t + \Delta t)$. We can now determine the truncation error:

$$\Delta x(\Delta t) = x_2(t + \Delta t) - x_1(t + \Delta t) \approx \mathcal{O}((\Delta t)^5) \tag{3.22}$$

$x_2(t + \Delta t)$ will be returned as an answer, because it will be the most accurate one. We will now determine what the best step size will be for the next step. We will assume that $\Delta x \approx \mathcal{O}((\Delta t)^5)$ (equation 3.22). We then can try two different values of Δt and we have:

$$\frac{(\Delta x)_0}{(\Delta x)_1} = \left(\frac{(\Delta t)_0}{(\Delta t)_1}\right)^5 \tag{3.23}$$

We rewrite 3.23 and get a formula for a step size:

$$(\Delta t)_0 = (\Delta t)_1 \left| \frac{(\Delta x)_0}{(\Delta x)_1} \right|^{\frac{1}{5}} \tag{3.24}$$

Let $(\Delta x)_0$ be the requested accuracy, then if $(\Delta x)_1 > (\Delta x)_0$ equation 3.24 tells us how much to reduce the step size so we can repeat this "failed" step. If $(\Delta x)_1 < (\Delta x)_0$ equation3.24 tells us how much we can stretch the step size for the next step.

Chapter 4

Results

In this chapter the validation of this version of ComFlo is presented.

4.1 Results 2D interactive simulations

The validation was done by comparing the result from the calculations in Comflo with data from measurements of model experiments. We will first give a short descriptions of the numerical experiments and the real life model experiments.

4.1.1 Numerical simulations

The numerical simulation is performed by a special version of Comflo. This version of Comflo consists of two parts: the original Comflo and a shell (or rind) around it. The shell contains the dry mathematical model as defined in 2.1. During a simulation, Comflo calculates at each time step the water reaction forces and torques working on the ship and returns these to the shell. The shell then combines these forces and moments with the normal, friction and gravitational forces and torques and the shell calculates after a double integration the position, velocity, rotation and angular velocity of the ship at the next time step.

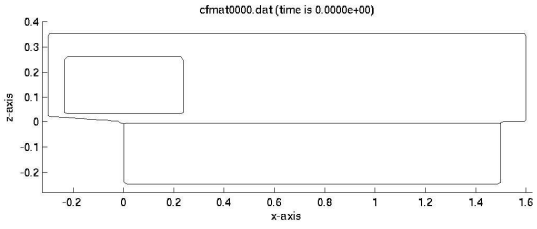

Figure 4.1: Geometry and free surface from experiment 2a in ComFlo

But first we need to define our parameters and general setup. The parameters and geometry

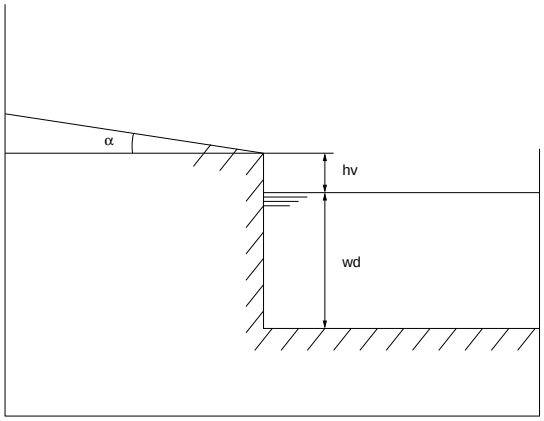

Figure 4.2: Geometry of the canal

of a model experiment are placed in the input files for the shell and COMFLO (See A.4). A typical geometry is plotted in figure 4.1.
Using a grid refinement study we will decide which grid size we use for the validation of the numerical simulations With some grid sizes it can occur that the simulation fails as a result of a steadily growing pressure or speed in one cell located at the edge of the ship. This problem is currently being fixed and is not the focus of this project. In the meantime small alterations in the grid size may solve this problem.

Early in this project, problems arose in the dynamical interaction between object and water. Changes were made to the original COMFLO by G. Fekken. He added an implicite (brute force) method which seperated solving of the Poisson equation and calculation of the velocities at the new timestep by creating a subiteration where those velocities are computed using a relaxation method.

4.1.2 Model experiments

The model experiments, which we use for the validation, were designed in 1968 by J. Versluis [10] and were performed for validating van Hooren's mathematical model for the side launch of a ship [8]. There are recorded experiments with two types of ships: a rectangular model of a ship and a 1:25 model of the "Flynderborg". As was said in the introduction we will focus on the validation of the numerical experiments of the side launch of a box shaped ship.
The setup of these model experiments consisted of a modeled canal, slope and ship.
The canal was modeled using a rectangular tank 8.5 m long and 1.5 m wide. The bottom was adjustable in the vertical direction, so it was possible to perform experiments with different values for wd and hv (see figure 4.2).
In the model experiment the slope was modeled using a transporter belt. This gave a constant speed to the model and simplified matters because no grease or other lubricant had to be used. The angle of the slope and the speed on the slope could be adjusted. By using the transporter belt there was no friction involved during the launch of the model.
The ship was modeled using a rectangular box with a length of 1.76 m, a width of 0.47 m

and a height of 0.228.

Using this setup a total of twelve model experiments were performed. Each of these experiments had a different set of parameters. Experiment 2a was given the default parameters:

description	symbol	standard setting
speed on slope	$fb1$	0.6 m/s
acceleration on *slope*	$ffb1$	0 m/s
height of the quay	hv	0.006 m
water depth	wd	0.24 m
height of sleigh	hs	0.032 m
width of sleigh	bs	0.235 m
angle of slope	α	0.07 rad
height C.G.	KG	0.162 m
mass ship	P	47.8 Kg
radius of gyration	i	0.170 m

The parameters of the other eleven model experiments consisted of a small alteration from these default values. The differences in parameters for the other experiments were:

experiment	parameter	value
1a	$fb1$	0.3 m/s
3a	$fb1$	0.889 m/s
4a	hv	0.048 m
5a	wd	0.24 m
	hv	0.017 m
6a	wd	0.28 m
	hv	0.017 m
7a	α	0.14 rad
	hs	0.042 m
8a	hs	0.042 m
9a	KG	0.194 m
	$fb1$	0.3 m/s
10a	KG	0.194 m
11a	P	37.8 Kg
	$fb1$	0.3 m/s
12a	P	37.8 Kg

During each of these experiments the position of the center of gravity and the rotation around it were recorded. We must observe that the event that marks the beginning of the recording is unclear. The report [8] doesn't show us whether $t = 0$ is defined as the moment when the center of gravity passes $x = 0$ or $t = 0$ is defined as the moment when the ship starts to rotate. Because the time difference between these events is small, this will only result in a minor phase shifts.

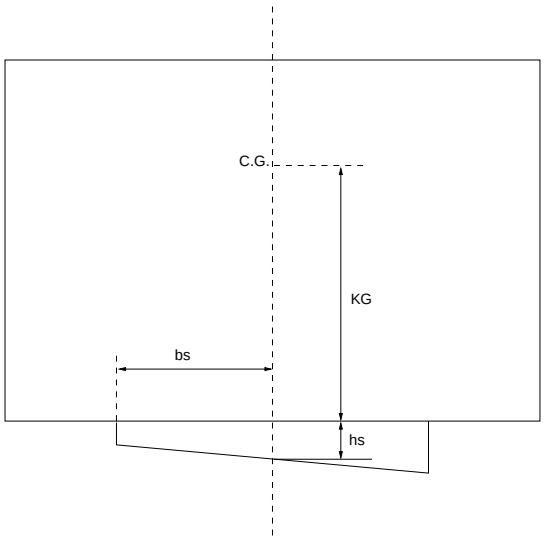

Figure 4.3: Geometry of ship and sleighs

4.1.3 Grid refinement

Before the validation of the program is presented, we will first perform a grid refinement study to determine the influence of different grid sizes on the results of the numerical simulations. This will give us also a first glance at the performance of the program. For the grid refinement study we pick out a small number of model experiments. Model experiments $7a$,$9a$ and $11a$ are chosen, because they represent the most influential changes one can make to the parameters of a launch (higher c.g., larger weight and a steeper slope). We simulate them with grid size 80x1x30,160x1x60 and 319x1x120. The simulations with grid size 319x1x120 are only performed for the most important first 1.5 seconds because they can take as long as a week to complete. The results of those simulations are plotted in figures 4.4, 4.5 and 4.6 together with the results from the model experiments to get a global view of the performance of the program.

Looking at figures 4.4 through 4.6 we see that:

- in figure 4.5 the results of the numerical simulations converge away from the results of the model experiments as the grid size becomes finer;
- the horizontal and vertical positions in the results of the numerical simulations show a consistent behavior for all the used grid sizes. This shows that the forces for the different grid sizes concur.
- the rotational results of the numerical simulations with grid size 319x1x120 are worse than the results of the simulations with a coarser grid. Although the forces are rather consistent for the different grid sizes, the moments are apparently vulnerable

for changes in the grid size. A reason for this may be the numerical loss of precision due to the deductions in the calculation of the moments.

- the results of the numerical simulations at different grid sizes do not converge towards the results of the model experiments.

We have seen that grid refinement does not lead to better results in the numerical simulations. It is advised to study this phenomena with the new version of COMFLO. The issue with spikes in the results, as we will mention in the next section, may also contribute to this. We also see some global differences between the results of the numerical simulations and the results of the model experiments. These differences will be discussed as we validate the program in the next section. Because we see no improvement of the performance of the program with finer grids, we choose the grid size 160x1x60 for the validation of the numerical simulations. This is mainly based on the runtime of the program with that grid size.

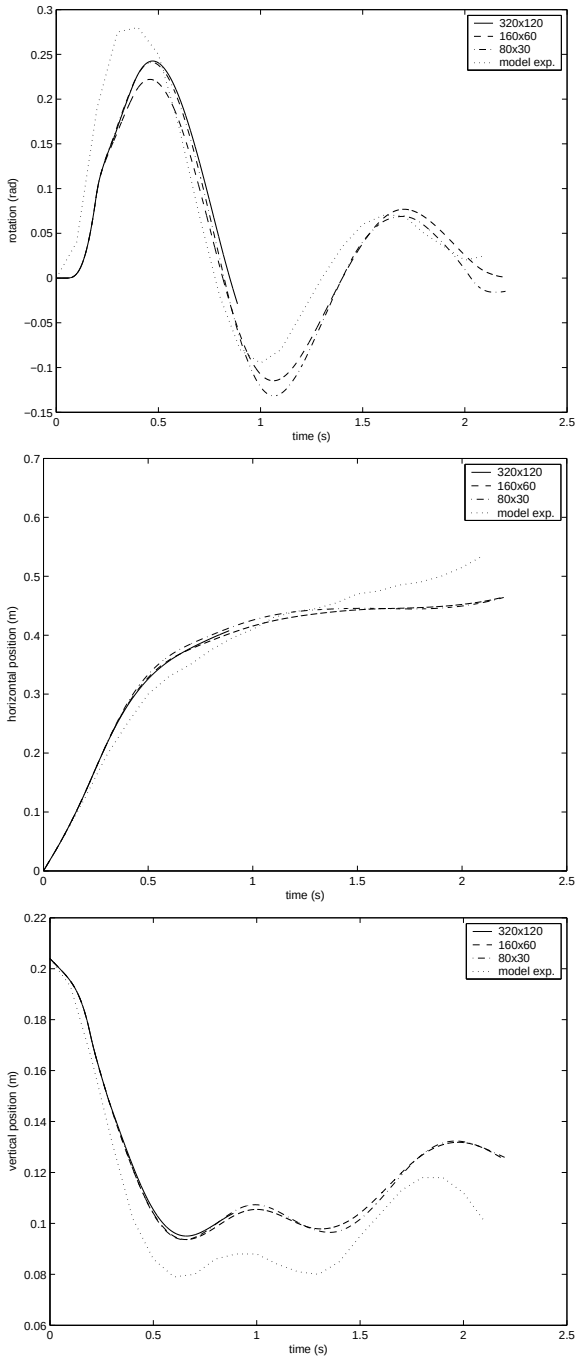

Figure 4.4: Experiment 7a ($\alpha = 0.14rad$,$hs = 0.042m$)

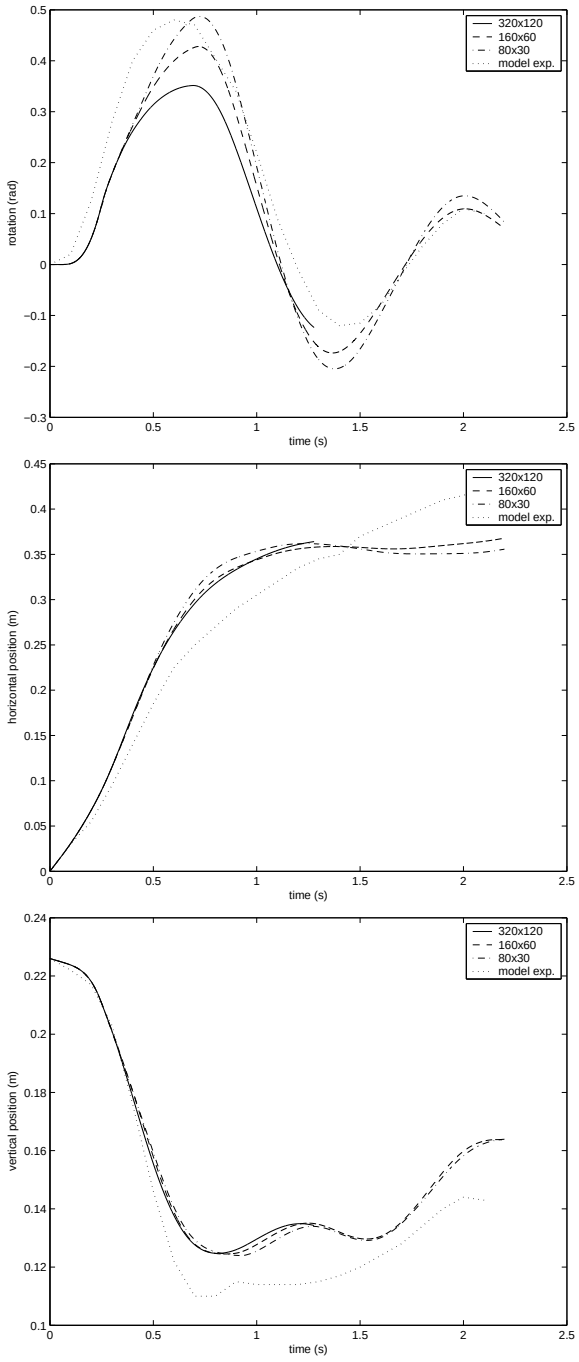

Figure 4.5: Experiment 9a ($KG = 0.194m, fb1 = 0.3m/s$)

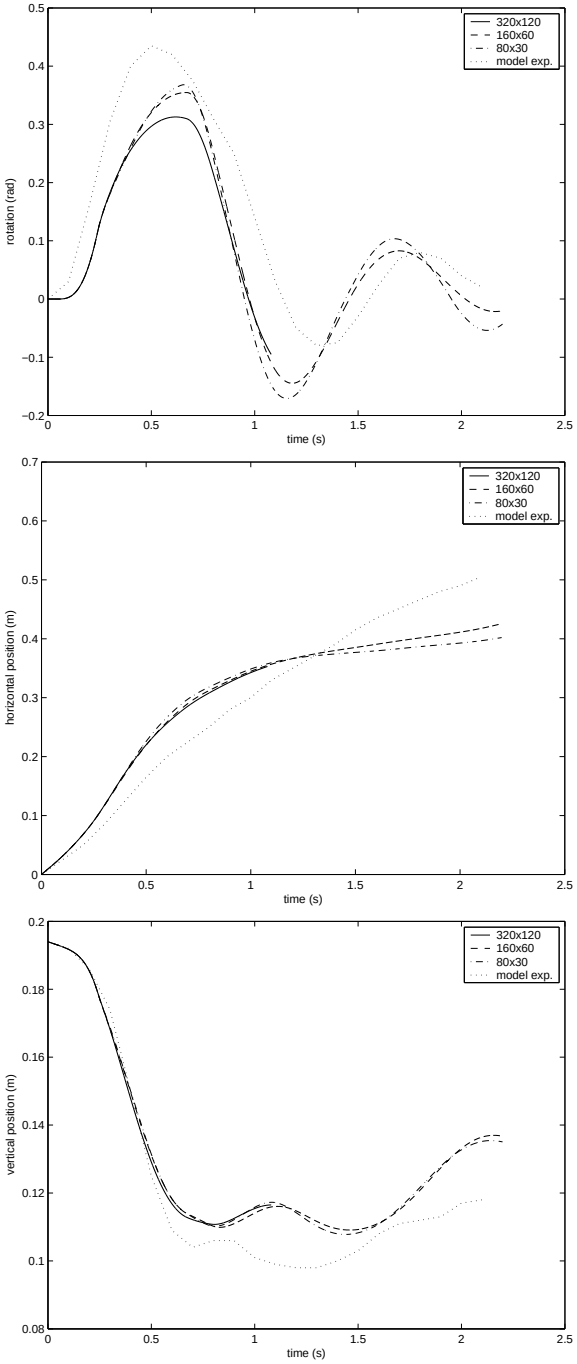

Figure 4.6: Experiment 11a ($P = 37.8kg, fb1 = 0.3m/s$)

4.1.4 Validation

In this section the validation of the program is presented. The validation is done by comparing the results of the model experiments $1a$ through $12a$ and the results of the numerical simulations of these model experiments. The data from the model experiments and the numerical simulations is plotted in figures 4.10 through 4.21. The motion of both chines is very important for ship builders, so plots of the motion of both chines are included in figures 4.10-4.21. Simulations $1a$ and $6a - 12a$ are performed on a 160x1x60 uniform grid. Simulation $2a - 5a$ are performed on a slightly different grid due to stability problems when running them on a $160x1x60$ grid.
Looking at the plots, we see that:

- in all simulations the ship goes less deep than in the recorded experiments. The difference in depth differs from 1 to 2.5 cm during the first seconds of the launch.
- the maximum angle of rotation during the first second in simulations $1a - 2a, 4a - 12a$ is smaller than the recorded angles in the experiments. As a result of this the water side chine in the simulation goes less deep than in the model experiment.
- the maximum and minimum angle of rotation during after the first second of the simulation are larger than the recorded angles of rotation in the model experiment. The damping effect of the simulated water appears to be smaller than it should be.
- in all simulations the rotational movement seems more or less out of phase.
- the movement of the chines in the numerical simulation differs much from the movement of the chines in the model experiments. Differences in the angle of rotation and the vertical and horizontal position cause greater changes in the movement of the chines.
- during the first phase, when the ship has not started to tilt, the results of the numerical simulation differs from the results of the model experiments. The model of the quay in the program does not exactly match the quay of the model experiments.
- during the 3rd phase, when the ship has no contact with the quay, the global movement of the vessel matches, although the exact position differs. This is especially shown by the position of the chines.

As we have seen, there are some large differences between the model experiments and the numerical simulations. The main difference is caused by the difference in draught (depth).

Draught

Because we are running the simulations on a coarse grid, we will first look at the height of the cells in the simulation:

experiment	grid	z_{min}	z_{max}	cell height
1a	160x1x60	$-0.246m$	$0.3536m$	$0.999cm$
2a	160x1x59	$-0.246m$	$0.3536m$	$1.016cm$
3a	159x1x60	$-0.246m$	$0.3536m$	$0.999cm$
4a	160x1x59	$-0.246m$	$0.3536m$	$1.016cm$
5a	159x1x60	$-0.257m$	$0.3536m$	$1.018cm$
6a	160x1x60	$-0.297m$	$0.3536m$	$1.084cm$
7a	160x1x60	$-0.246m$	$0.3636m$	$1.016cm$
8a	160x1x60	$-0.246m$	$0.3636m$	$1.016cm$
9a	160x1x60	$-0.246m$	$0.3856m$	$1.053cm$
10a	160x1x60	$-0.246m$	$0.3856m$	$1.053cm$
11a	160x1x60	$-0.246m$	$0.3536m$	$0.999cm$
12a	160x1x60	$-0.246m$	$0.3536m$	$0.999cm$

As we can see in the previous table the cell heights are about $1cm$. Because the difference in draught between the simulation and the model experiment is typically about 1-2.5 cm, the cell height is not the cause of the problem. We will therefor take a look at the general draught behavior of a floating object in ComFlo.
Our model ship has a water displacement of $47.8Kg$ and has therefore in rest a theoretical draught of $5.781cm$. We will simulate the draught of the ship after we place it in the water with it's center of gravity at $x = 1.0m$ and $z = KG - hv - 0.05781 = 0.09819m$, where hv is the water level and KG the height of the ship's center of gravity relative to the keel. For the simulation we use a 100x1x60 grid (cell height: $0.878cm$). The vertical position during the simulation is plotted in figure 4.7. We see that the vertical position of the c.g. converges to about $1.005cm$ (a draught of $5.550cm$). The difference in theoretical draught and the simulated one is smaller than the height of a cell. Under these normal circumstances ComFlo simulates a correct draught. The model experiments offer us

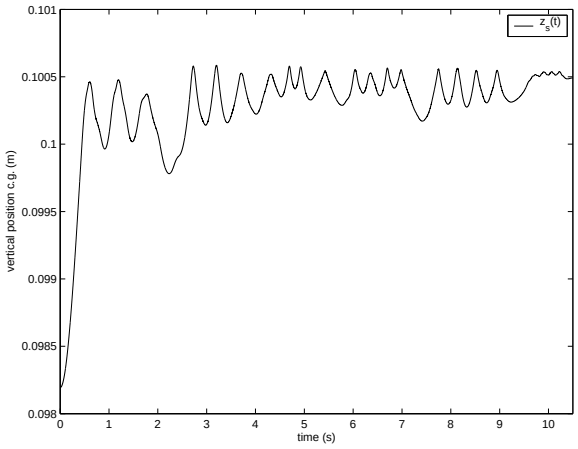

Figure 4.7: Draught test

a minimum of data (horizontal position, vertical position and rotation). To compare the vertical velocities of the real life model experiments and the numerical simulations, we use

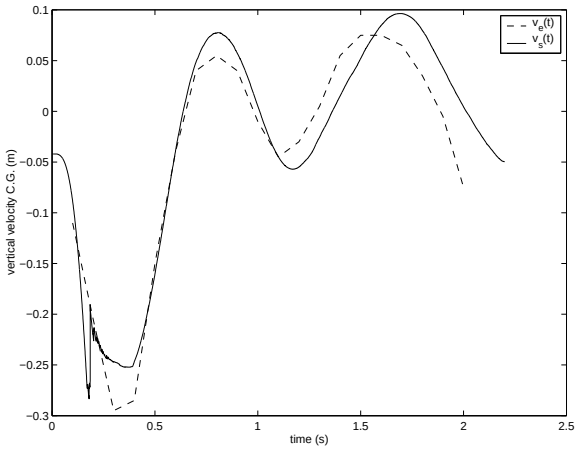

Figure 4.8: Experiment/Simulation 2a - vertical velocity

method A [9] to determine the velocity at time t_i:

$$u(t_i) = \frac{x(t_{i+1}) - x(t_{i-1})}{t_{i+1} - t_{i-1}} \tag{4.1}$$

The vertical velocity of model experiment 2a and simulation 2a is plotted in figure 4.8.

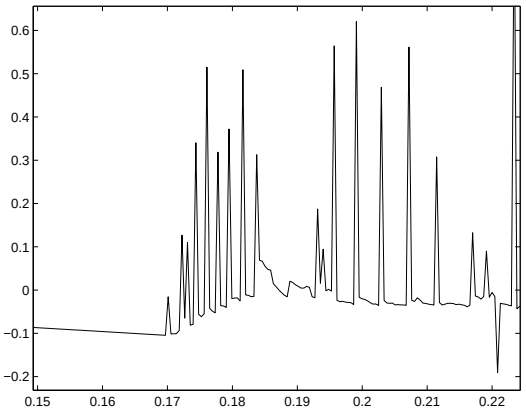

Figure 4.9: Vertical force from numerical simulation 2a

Looking at figure 4.8 we observe a non-smooth sudden rise in the velocity. We trace this back in the total vertical force in figure 4.9, where we notice large spikes in the force over multiple time steps (a time step is typically smaller than 10^3). These spikes (also known as "grass") could be the reason for the lack of draught.

All calculations in this report are done with a version of COMFLO, which was available at the start of this study. When the above problem was noticed, extra checks were done in the source code of COMFLO and only at the end of this project significant improvements

were made available which reduces the amount of spikes. Further evaluation of the above mentioned problem must be done with the new version of COMFLO. In the meantime we can only find the results of the program not accurate enough to predict the launch of a ship.

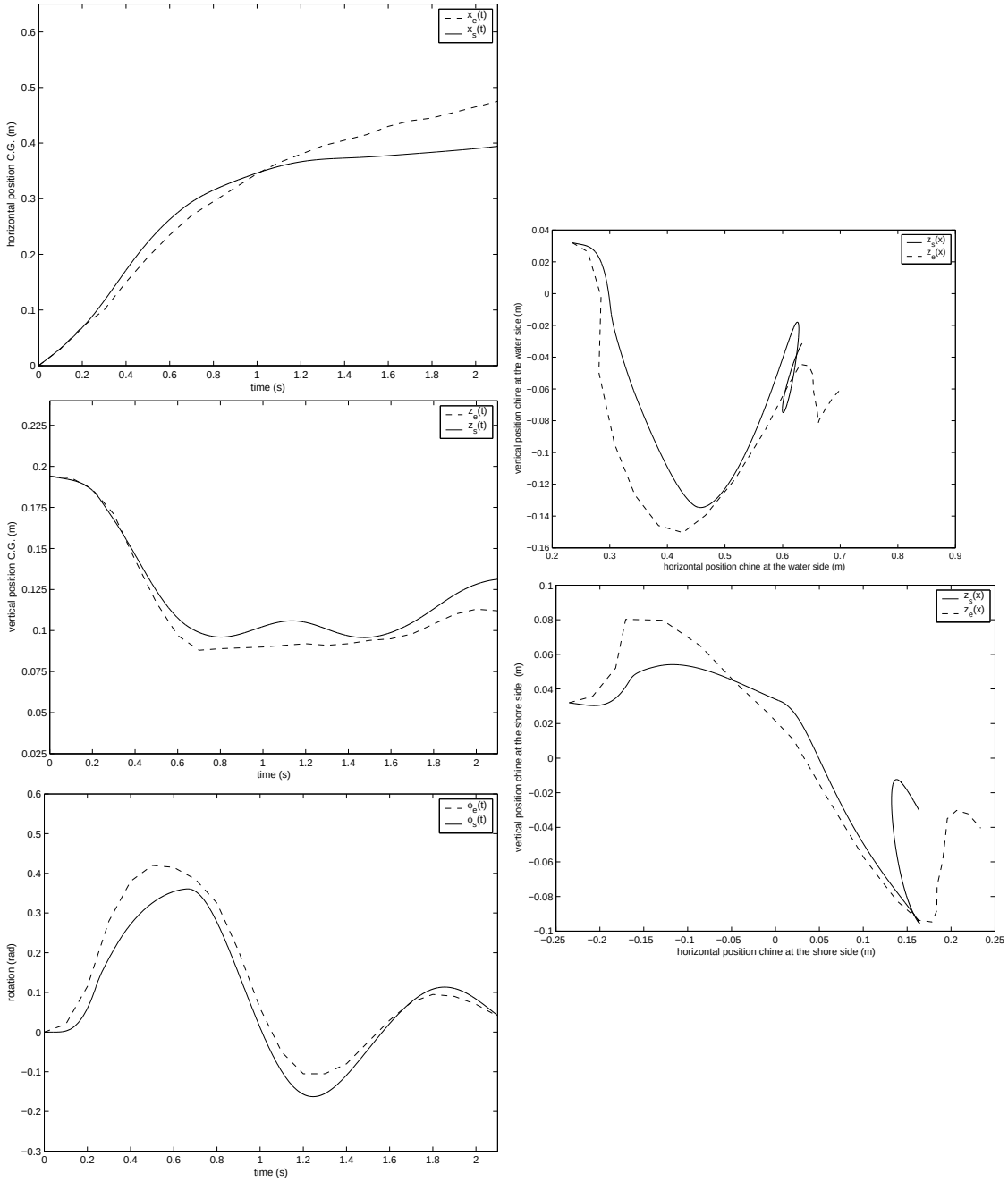

Figure 4.10: Experiment 1a ($fb1 = 0.3m/s$), grid 160x1x60

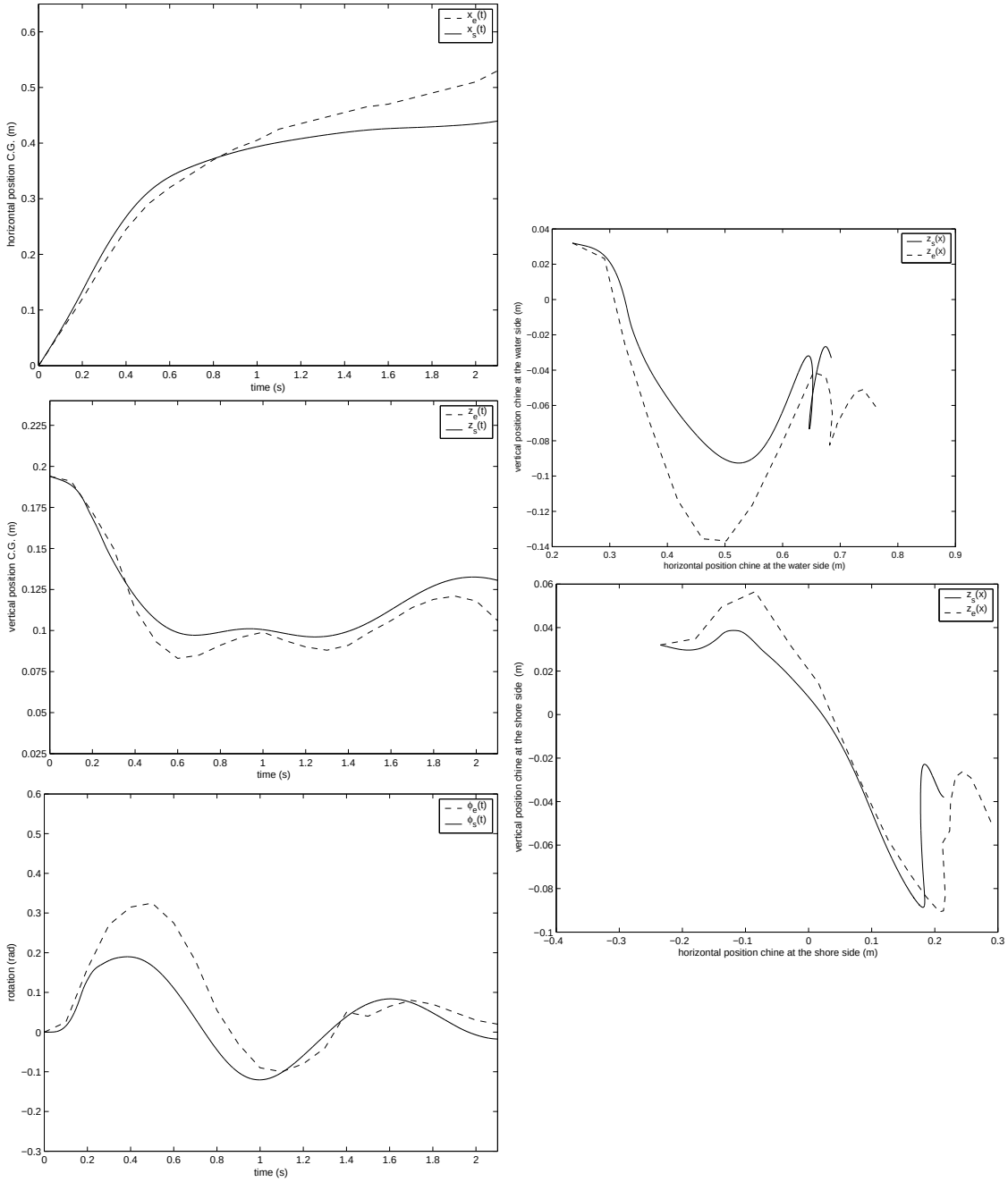

Figure 4.11: Experiment 2a (standard settings), grid 160x1x59

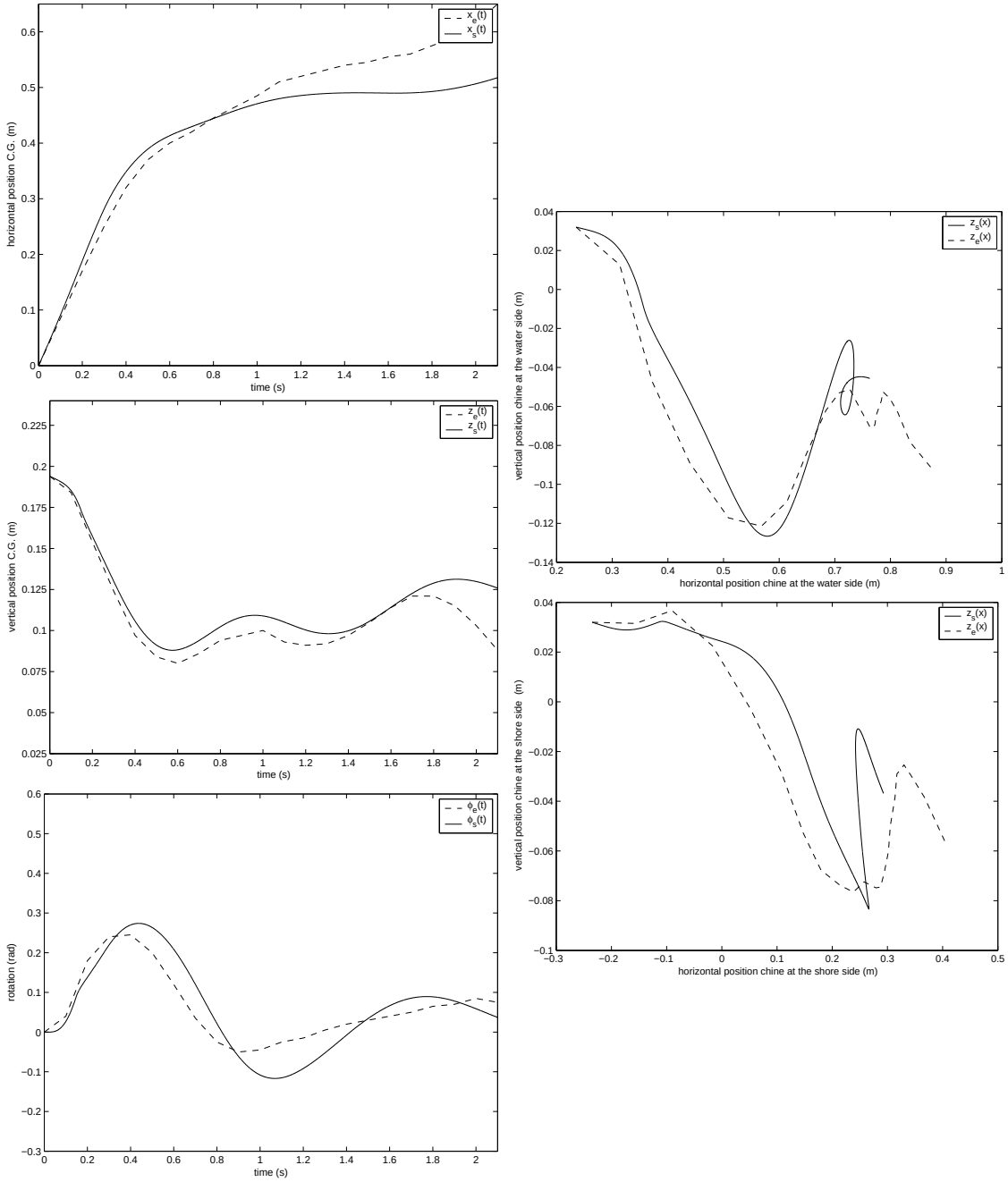

Figure 4.12: Experiment 3a ($fb1 = 0.889m/s$), grid 159x1x60

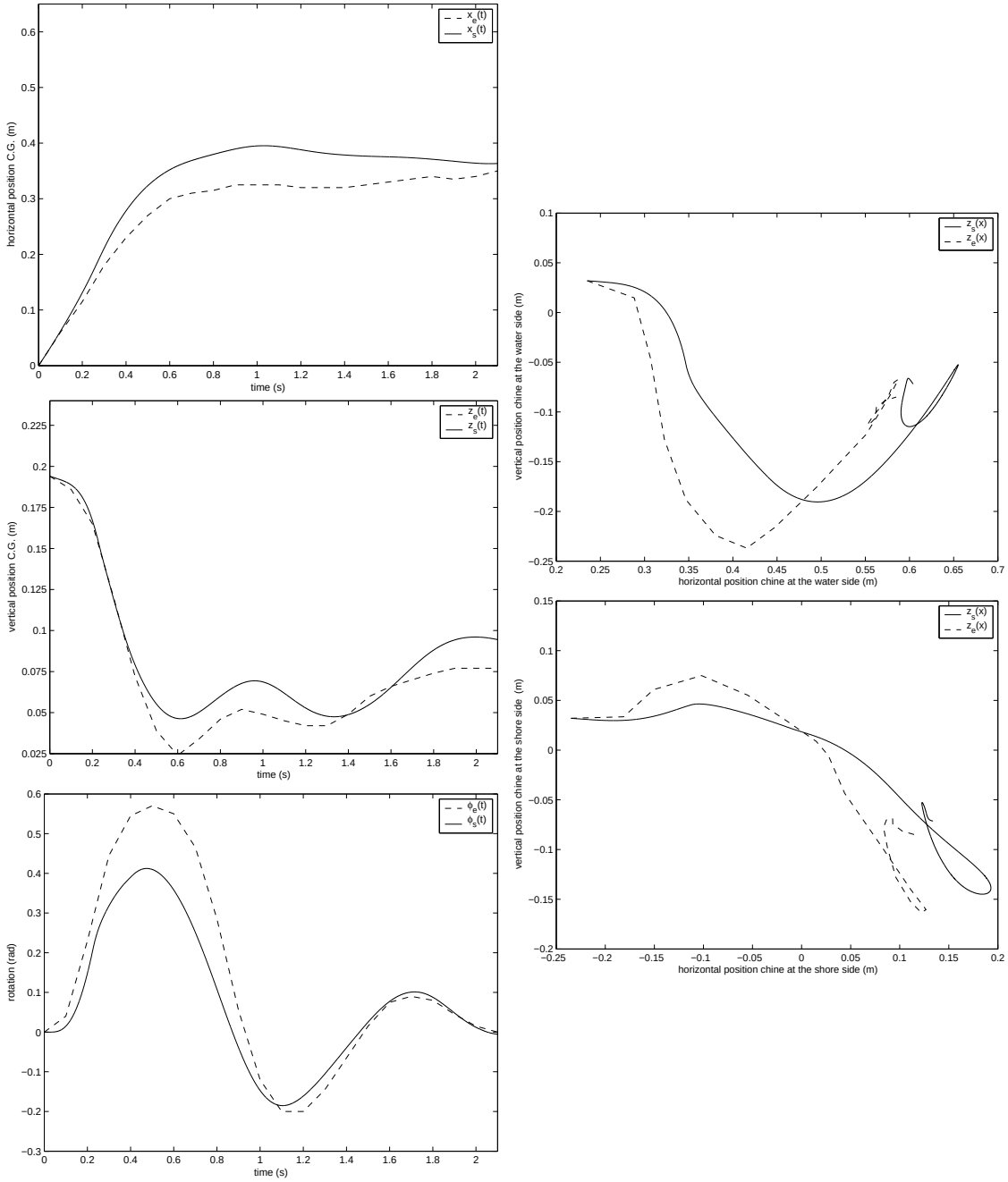

Figure 4.13: Experiment 4a ($hv = 0.048m$), grid 160x1x59

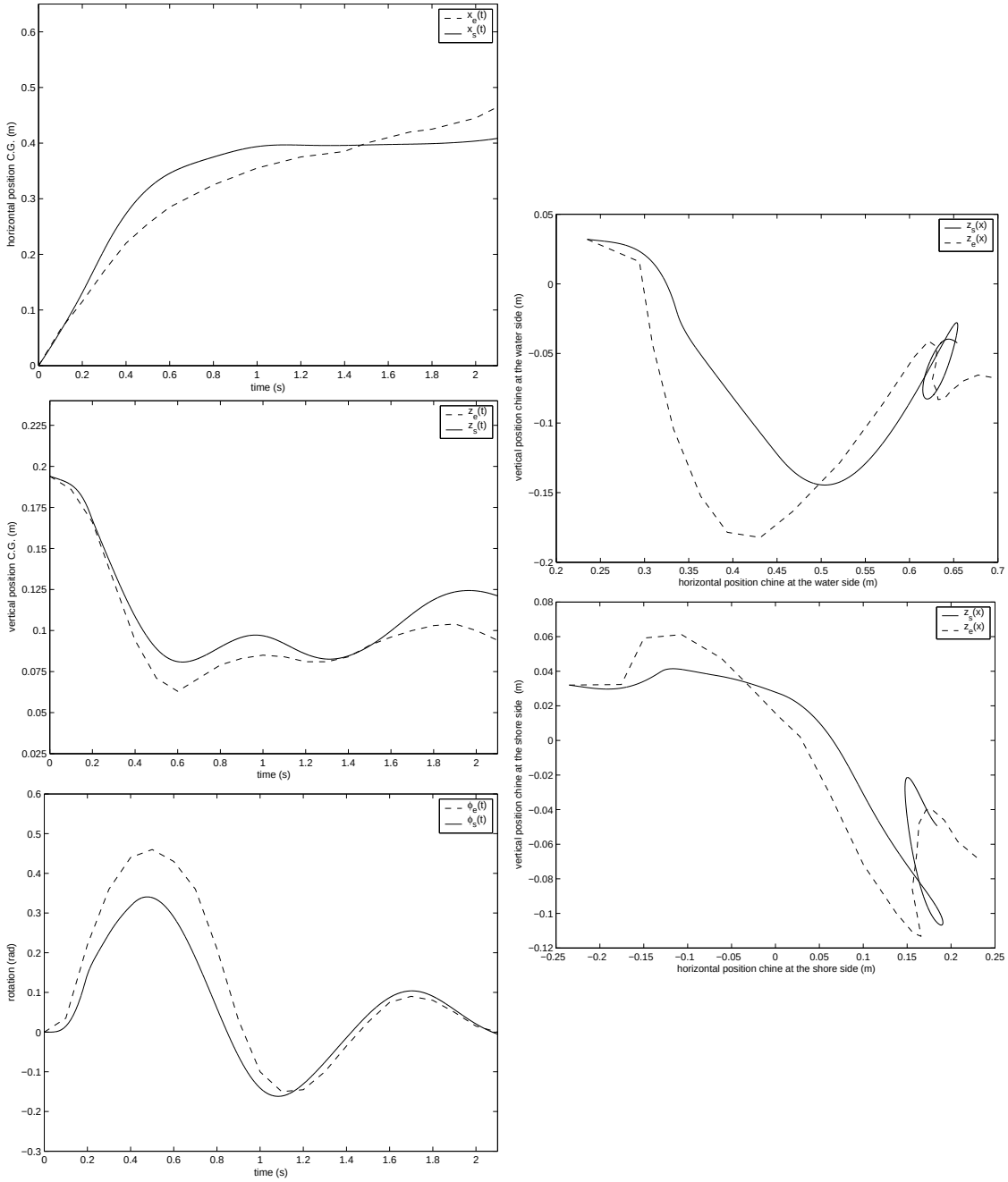

Figure 4.14: Experiment 5a ($wd = 0.24m$,$hv = 0.017m$), grid 159x1x60

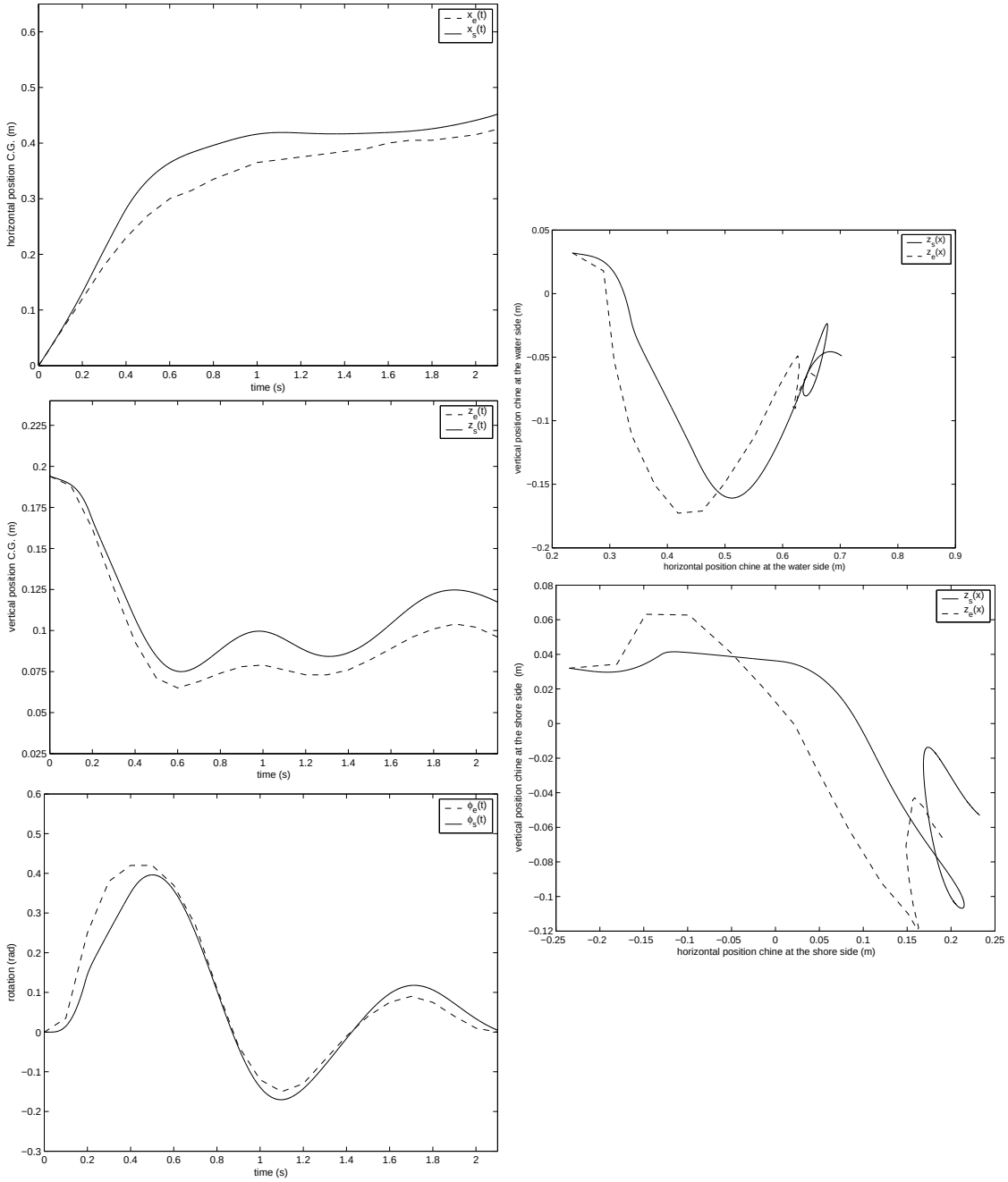

Figure 4.15: Experiment 6a ($wd = 0.28m$,$hv = 0.017m$), grid 160x1x60

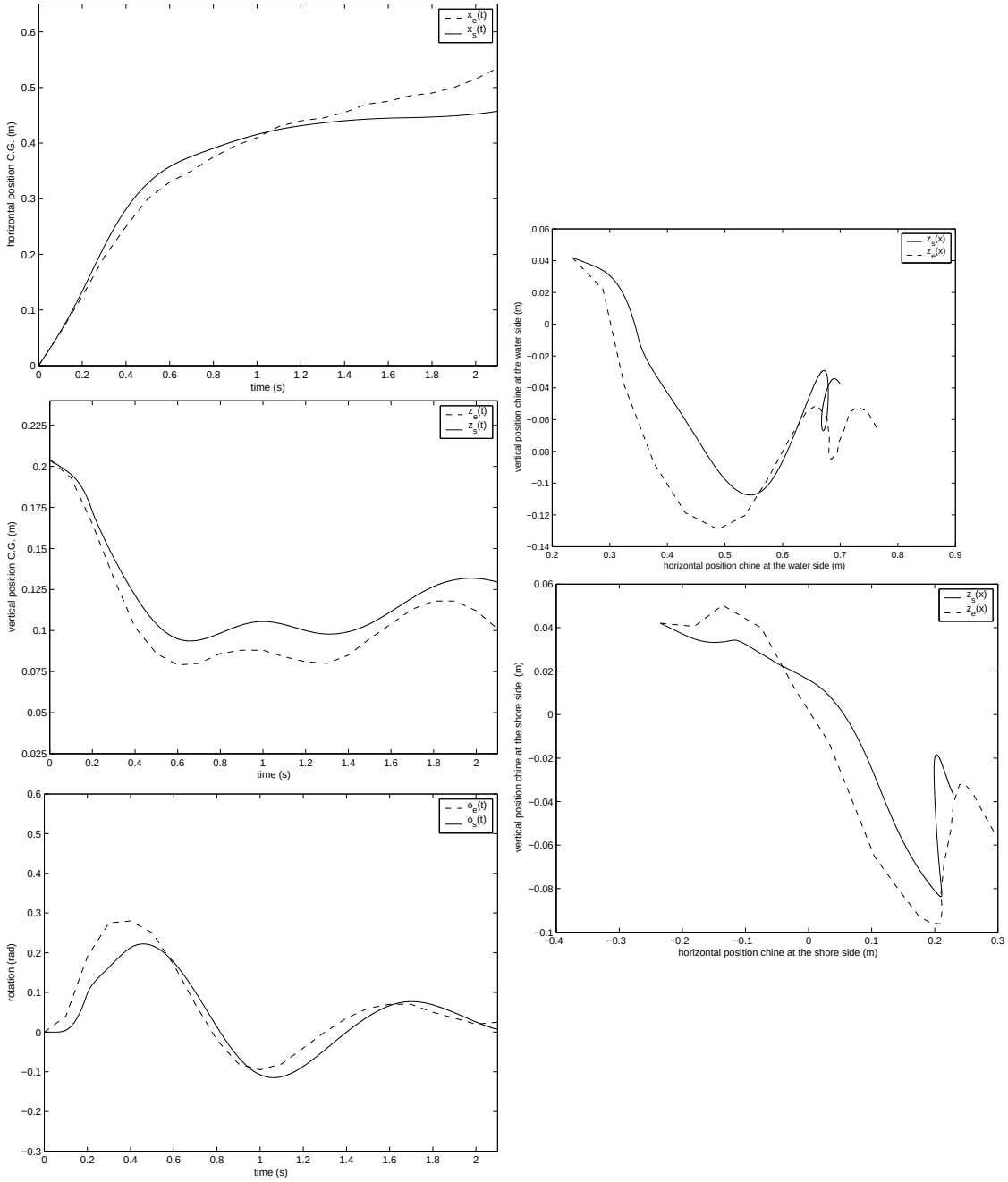

Figure 4.16: Experiment 7a ($\alpha = 0.14rad$,$hs = 0.042m$), grid 160x1x60

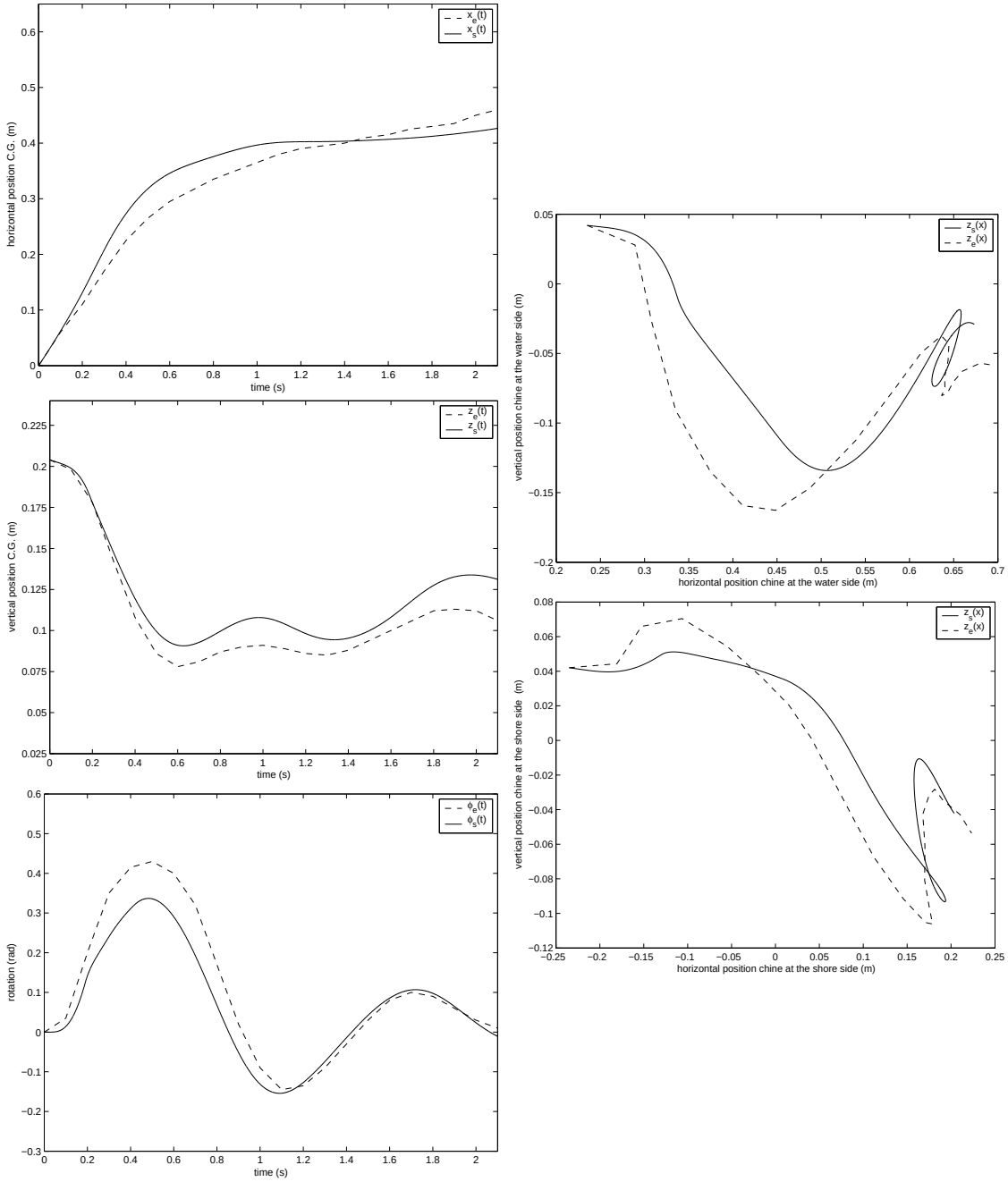

Figure 4.17: Experiment 8a ($hs = 0.042m$), grid 160x1x60

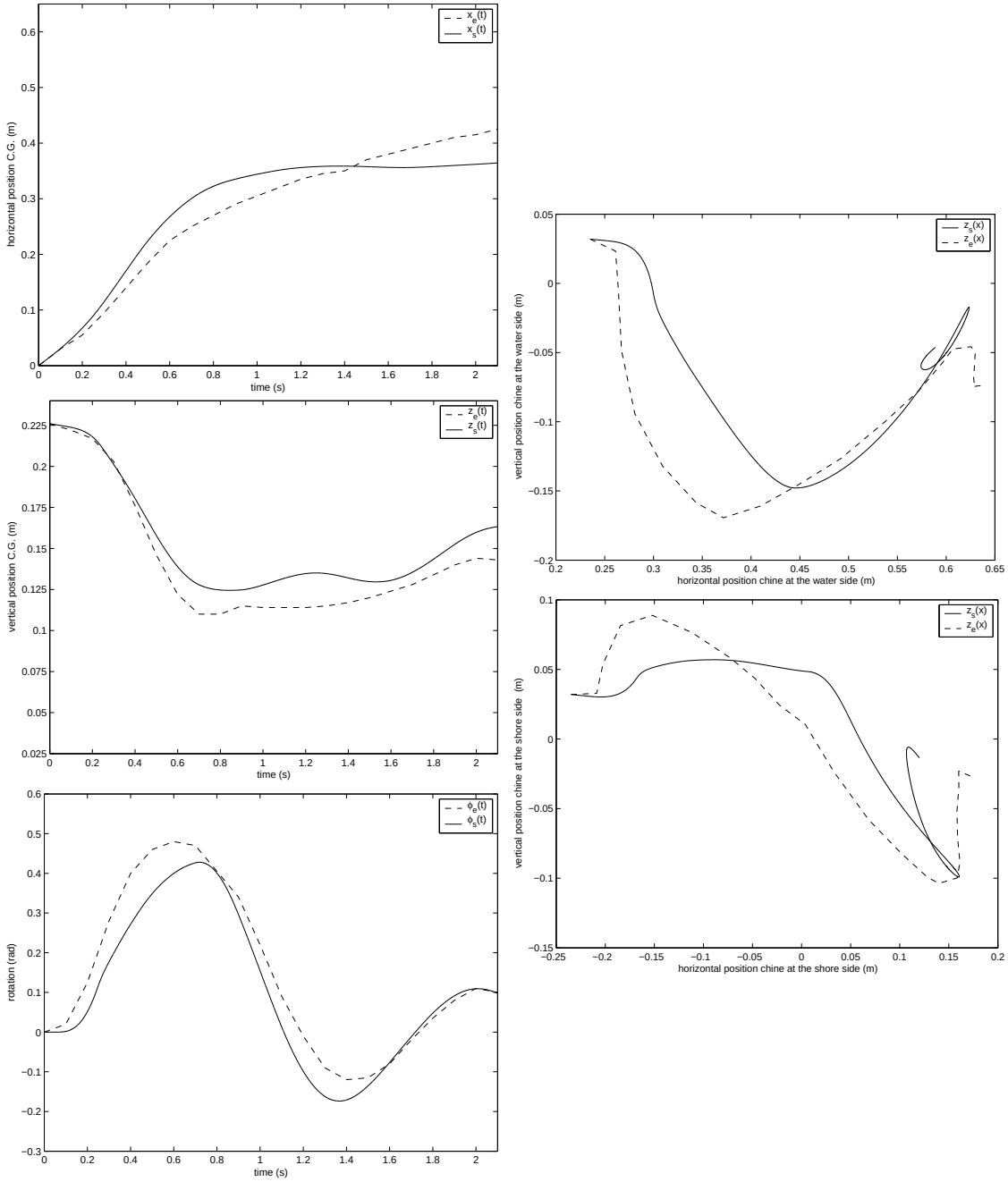

Figure 4.18: Experiment 9a ($KG = 0.194m, fb1 = 0.3m/s$), grid 160x1x60

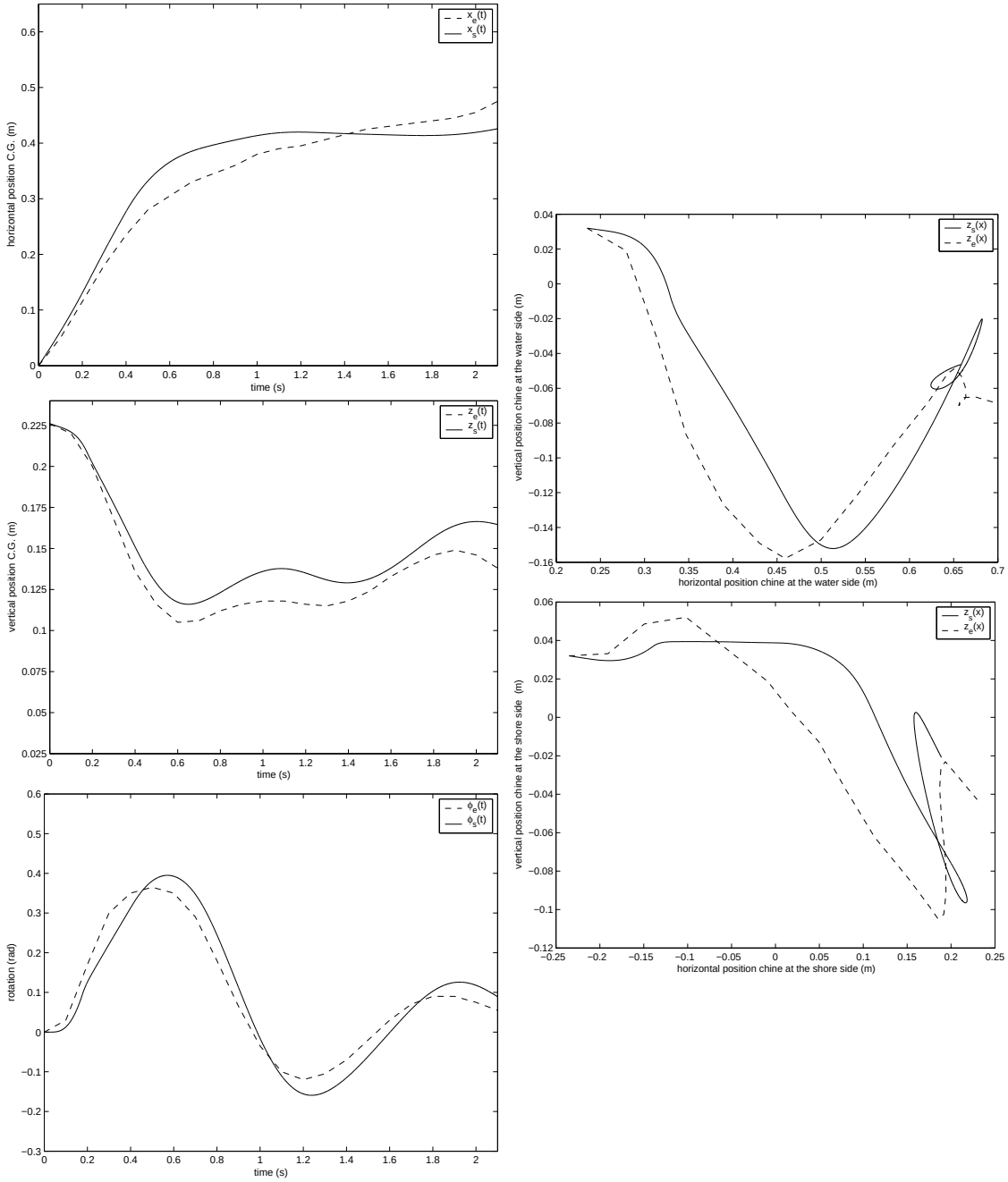

Figure 4.19: Experiment 10a ($KG = 0.194m$), grid 160x1x60

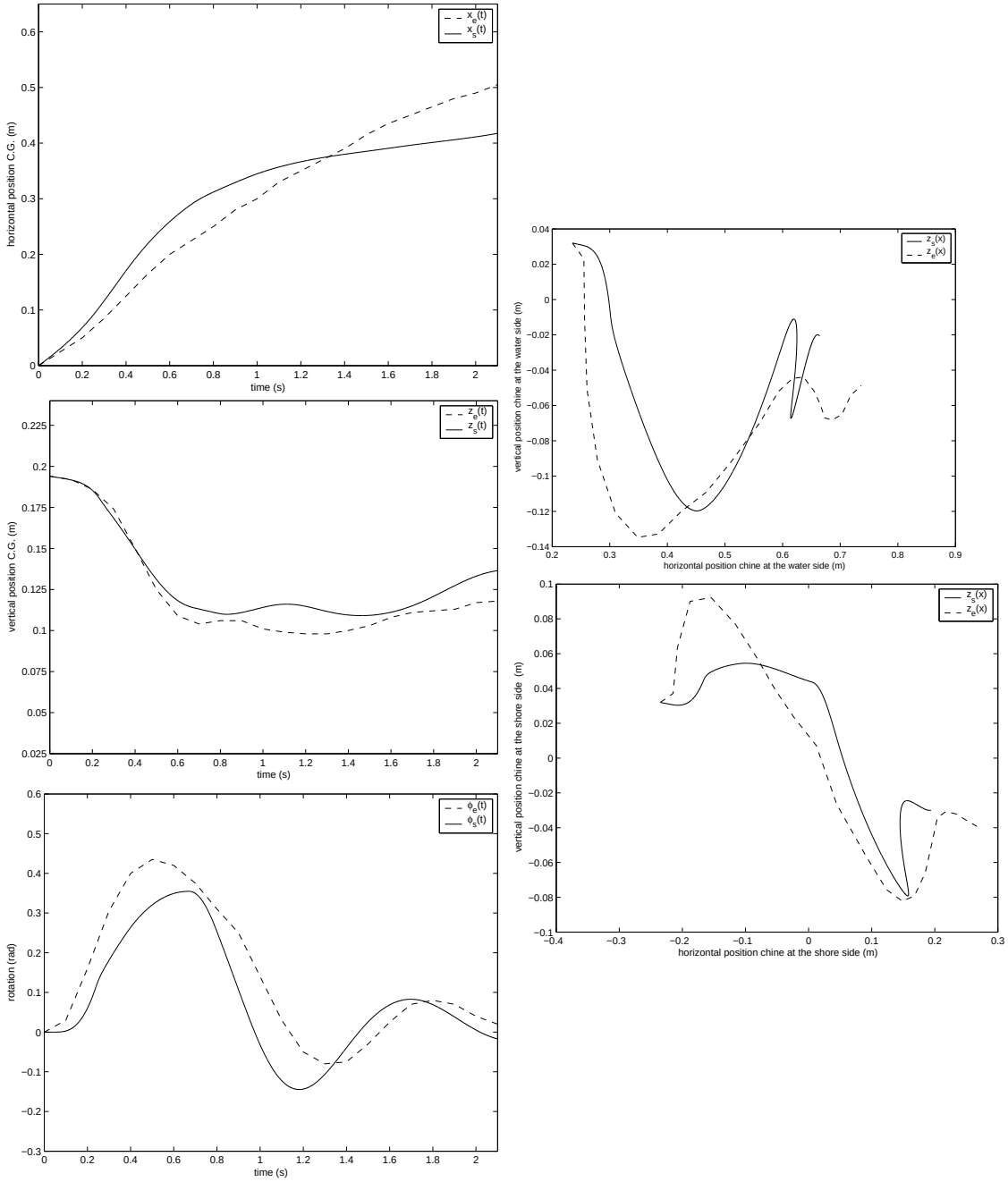

Figure 4.20: Experiment 11a ($P = 37.8kg, fb1 = 0.3m/s$), grid 160x1x60

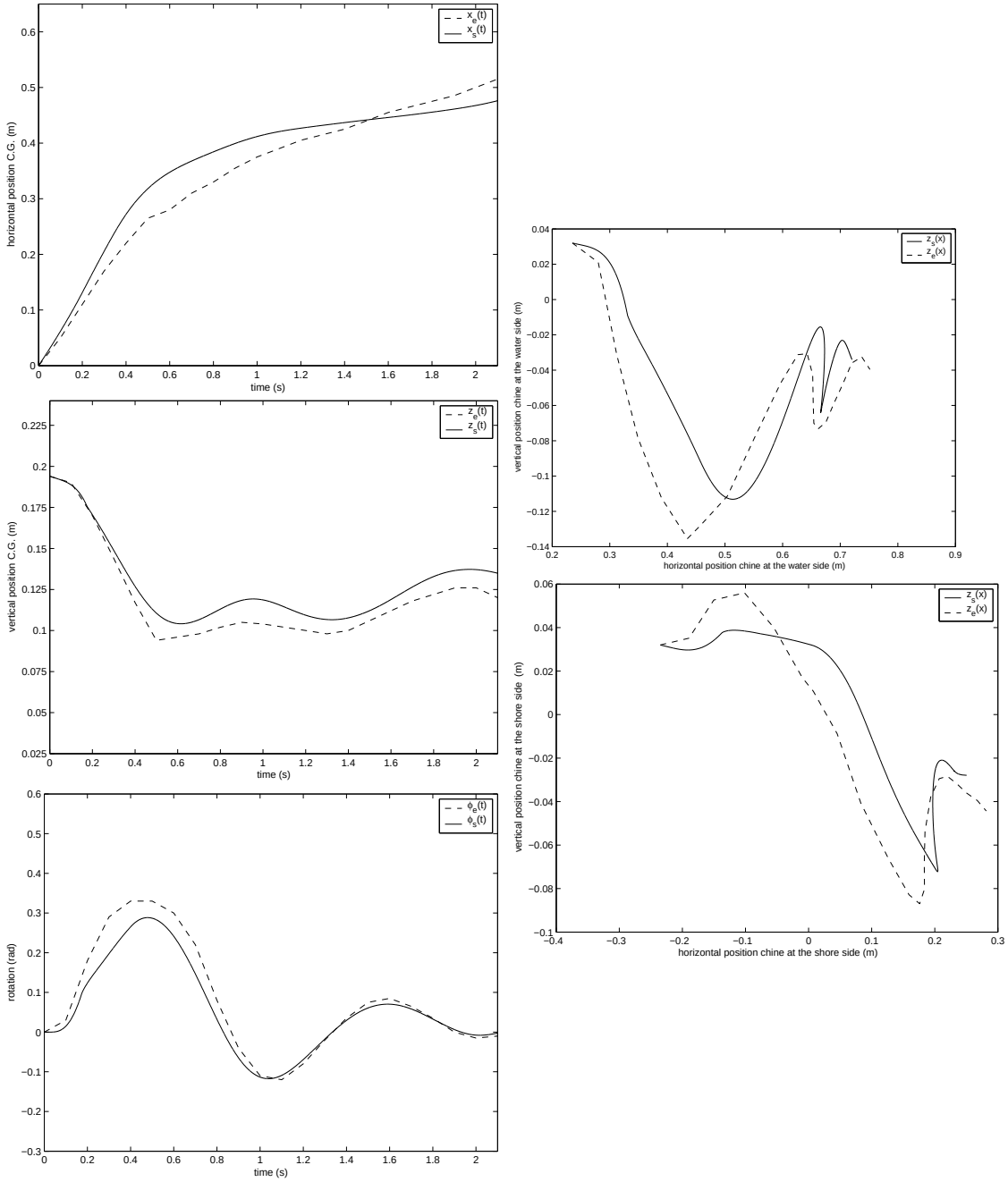

Figure 4.21: Experiment 12a ($P = 37.8kg$), grid 160x1x60

4.2 Results 3D interactive simulations

Part of this project is dedicated to extend the program so it could simulate the side launch of a rectangular ship in three dimensions. The validation of these simulations in three dimensions will be done in future projects, when it will be possible to simulate the launch of life like ships.

The numerical instabilities described in the previous section, is much more of a problem during 3D simulations and as a results of that it was almost impossible to run a 3d simulation with grid sizes finer than 70x40x40. The program uses the same input files for

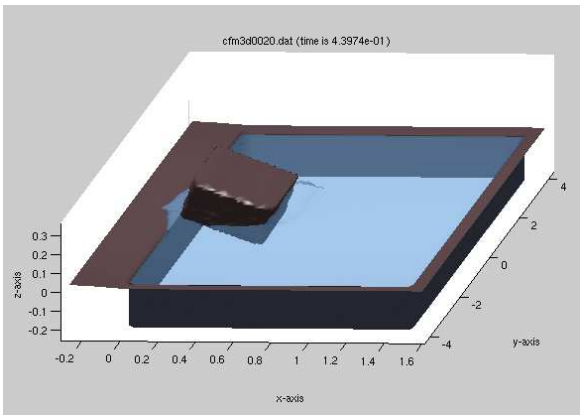

Figure 4.22: Snapshot of the simulation of a rectangular ship in three dimensions

two dimensional as three dimensional simulations. Depending on the grid size the program determines whether to create a two dimensional or three dimensional geometry (see figure 4.22. In figure 4.23 the results are plotted of a numerical simulation with grid size 61x41x41 of the second model experiment together with the recorded data from the second model experiment. Because of the coarse grid and numerical instabilities the computer program in its current state is not fit for validation and this will be done in a future project.

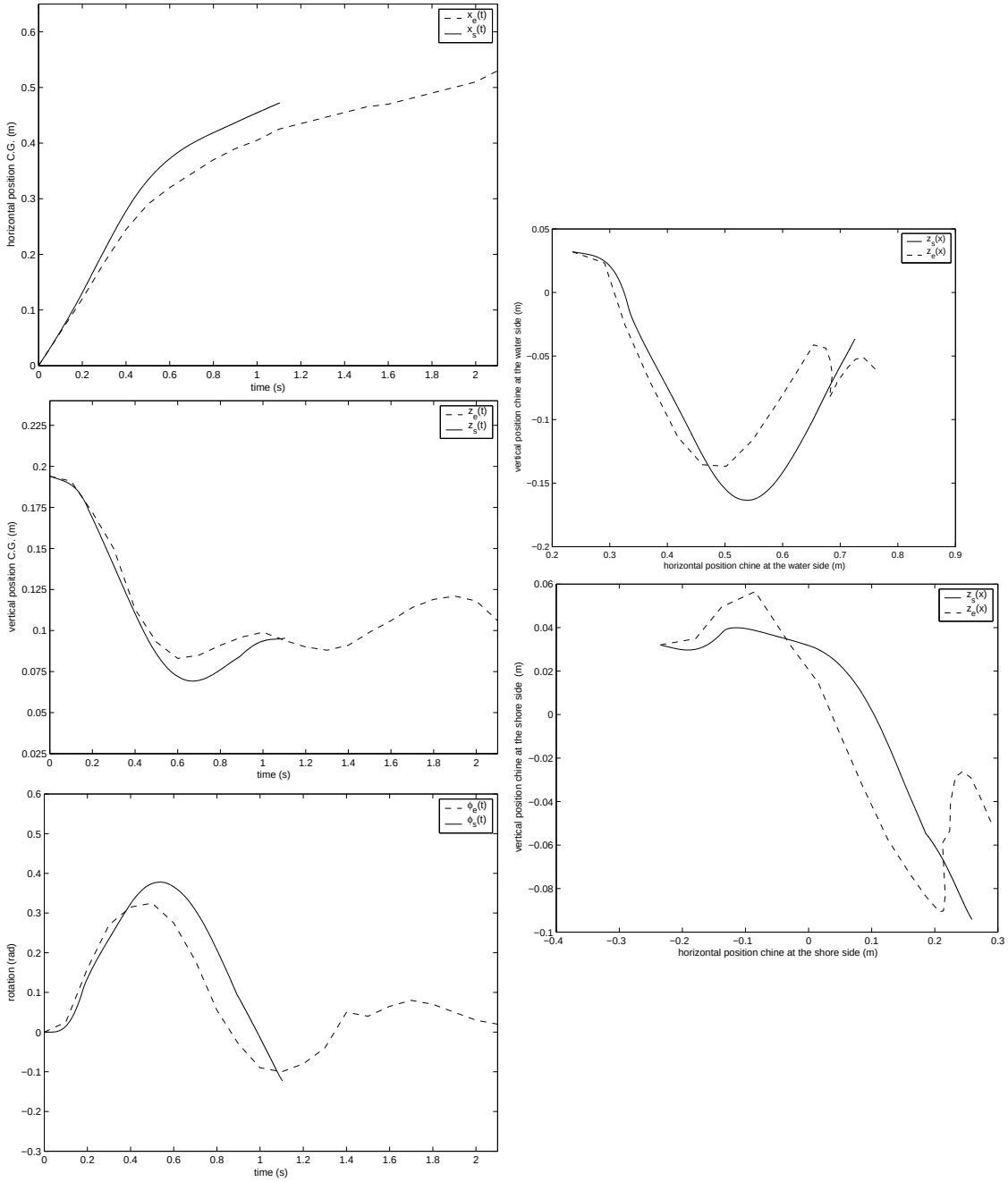

Figure 4.23: Experiment 2a (standard settings), grid 61x41x41 (3D)

Chapter 5

Conclusion

This report contains a mathematical model for the side launch of a ship and the results of computer simulations using this mathematical model to simulate the side launch of rectangular shaped ships. The mathematical model used in the numerical simulations is a combination of a simple "dry" mathematical model [8], created by Van Hooren, and the Navier-Stokes equations [9].

The validation of the created computer program (an extended version of ComFlo) is included in the results. This validation was done by comparing the results of model experiments, which were designed by J. Versluis [10] in 1968, and the results of the numerical simulations of these model experiments. During the validations, some numerical problems were found, which cause a reduced draught during simulations.

At the beginning of this project the interaction between an object and water was still in its infancy. The interaction was not yet validated and is still not free of numerical instabilities. Especially exponential increasing speeds and pressures typically in one cell on the edge of the moving object can cause simulations to fail. Also grass (or spikes) in the calculated water forces and moments are a problem and cause a different movement of the ship during a simulated launch. All calculations in this report are done with a version of Comflo, which was available at the start of this study. Only at the end of this project significant improvements were made available which reduces the amount of spikes. Although Comflo still causes some problems when simulating objects interacting with water, the general conclusion of this report is that the extended version of Comflo is suitable for this purpose, but still a lot must be done before an accurate prediction of a real launch can be made. Things to do are:

- For a realistic prediction of real launches the mathematical model has to be extended with:
  - rounded (parabolic shaped) ends of the slope.
  - a shifted center of gravity of the ship. During a real launch the center of gravity isn't located midship, but more to the side where the quay is located. This is done by ballast tanks.
  - the possibility that a ship already has a certain angle of rotation on the slope. At this moment $\phi = 0$ on the quay, but in real life this often isn't be the case.

- For a realistic prediction of real launches the program must be able to simulate ships with a more realistic geometry. At this moment the program always launches a rectangular ship, which even differs with the shape of a coaster at both sterns. In the near future it should be possible to input the exact shape of the ship from CAD/CAM applications.

A future program which has been extended with these features must be validated with real life data. During this study some recordings of real life launches were gathered for this purpose. To avoid the problems with the water forces and moments, which were encountered in this study, future studies must be done with the new version of COMFLO.

The results of the validations done in the previous chapter show that when in the near future these numerical problems are solved the program will be able to accurately predict the path of a rectangular ship. When besides that the discussed features are added to the program, the ship builders will have their tool to increase the chance of a successful launch of a real ship.

Appendix A

Program description

A.1 Calling Sequence

As was said before, a rind was created around COMFLO. Both COMFLO and the rind consist of several subroutines. Table A.1 shows in which order these subroutines are called, when a simulation is run.

A.2 subroutines

In this section a list of the most important subroutines in COMFLO will be presented. Because COMFLO was divided in a COMFLO kernel and a shell (or rind) around it, we'll first present a list of the procedures in the COMFLO kernel:

AUTOSV	:	Creates and reads auto save files that are used for restarts
AVS	:	Post-processing: outputs data for use with AVS
BC	:	Calculates the boundary velocities
BDYFRC	:	Computes translation and rotation in virtual body force. Determines the roll frequency for control reasons
BNDDEF	:	Defines the boundary of the geometry
BNDLAB	:	Calculates apertures and labels the cells and velocities
CELLNR	:	Computes indices of the vertex of a point with given coordinates
COEFL	:	Calculates the left hand side of the pressure Poisson equation
COEFLQ	:	Calculates the left hand side of the pressure Poisson equation for liquid movement
COEFR	:	Calculates the right hand side of the pressure Poisson equation
COEFRQ	:	Calculates the right hand side of the pressure Poisson equation for liquid movement
COM	:	Computes the centre of mass of the fluid
COMFLO_FINALISE	:	Finalises some output and variables

COMFLO_GET_ALL_DATA	:	Retrieves the current position, speed, angle and angle velocity of the C.G.
COMFLO_GET_FORCES	:	Determines the current force and moment on the C.G.
COMFLO_GET_SPEED_DATA	:	Determines the current speed and angle velocity of the C.G.
COMFLO_GET_TIME	:	Determines the current time, time step and maximum time
COMFLO_INITIALISE	:	Initialises variables and prepares for time step
COMFLO_PUT_DATA	:	Applies the calculated position, speed, angle and angle velocity of the C.G.
COMFLO_PUT_FORCES	:	Applies the calculated total force and moment working on the C.G.
COMFLO_READ_CONFIG_BEGIN	:	Interface for setpar_begin
COMFLO_READ_CONFIG_END	:	Interface for setpar_end
COMFLO_TIMESTEP	:	Performs a time step
CROSS	:	Returns the cross product of two vectors
DTADJ	:	Adapts the time step using the CFL-condition
DYNINT	:	Implicite computation of the interaction between object and liquid (2D)
DYNINT3D	:	Implicite computation of the interaction between object and liquid (2D)
FILLBX	:	Post-processing: Returns data about the volume of fluid in certain boxes
FLUXBX	:	Post-processing: Returns data about the fluxes through a certain box
FRCBX	:	Post-processing: Returns data about the forces and moments in a certain box
GEOMV2D	:	Defines 2D object geometry at the next time step
GEOMV3D	:	Defines 3D object geometry at the next time step
GRID	:	Generates a (stretched) grid
INIT	:	Initialises variables for a new time step
INTERP	:	Interpolates between values of variables
IOBC	:	Creates boundary conditions at input and output boundaries
IOLAB	:	Creates input and output labels
LIQPCT	:	Computes liquid percentage and returns it to screen
MATL3D	:	Post-processing: produces 3D data for use of Matlab
MATLAB	:	Post-processing: produces 2D data for use of Matlab
MNTR	:	Post-processing: monitoring of pressure or velocity in a fixed point
NEIGHB	:	Determines the neighbours of a certain cell
NORMAL	:	Computes the normal of a solid boundary
OBJDEF	:	
ORDLST	:	Reorder a list in order of magnitude for computation of edge apertures
PLIC	:	

PRESBC	:	Resets the pressure in the BS, BE and E-cells to p_0. Computes the pressure in the S-cells using interpolation. Changes the pressure at the free surface to block the water if control of U-tank is on.
PRESIT	:	Performs iterations to solve the pressure equation until the error is small enough of the number of iterations is greater than the maximum number of iterations.
PRNT	:	Console output
PROJ	:	Projects a given vector on a given plane.
SETFLD	:	Initialises variables at start of computation
SETPAR_BEGIN	:	Begins to read input files
SETPAR_END	:	Second and last part of reading input files
SLAG	:	Computes one SOR-sweep to solve the pressure Poisson equation
SOLVEP	:	Solves pressure from the pressure Poisson equation
SOLVEQ	:	Solves pressure from the pressure Poisson equation for liquid movement
STREAM	:	Post-processing: Produces streamlines
SURDEF	:	Initialises the volume of fluid function FS
SURLAB	:	Labels the cells and velocities based on the free surface configuration
TILDE	:	Computes the temporary field $\tilde{u}$
TRAFOS	:	Determine translation and rotation with respect to inertial system.
VELBC	:	Computes the free surface velocities.
VFCONVF	:	Computes the volume of fluid function FS at the new time level.
WRTFRC	:	Writes raw data to multiple files

The procedures COMFLO_GET_ALL_DATA, COMFLO_GET_FORCES, COMFLO_GET_SPEED_DATA, COMFLO_GET_TIME, COMFLO_PUT_DATA and COMFLO_PUT_FORCES are added to the original COMFLO code to provide a means of transportating data between rind and kernel. The main COMFLO kernel procedure is divided into COMFLO_INITIALISE, COMFLO_TIMESTEP, COMFLO_FINALIZE. The kernel procedure SETPAR is split into SETPAR_BEGIN and SETPAR_END and their interfaces COMFLO_READ_CONFIG_BEGIN and COMFLO_READ_CONFIG_END were added. Furthermore some changes were made to the kernel procedures DYNINT, DYNINT3D, BNDDEF, GEOMV2D and GEOMV3D.
And the list of procedures in the rind around the COMFLO kernel:

DERIVS	:	Calculates acceleration and angular acceleration
CALC	:	Calculates the total force and moment on the centre of gravity
CALC_POSITION	:	Calculates the new position of the centre of gravity, relative to position in the previous time step.
READ_SETTINGS	:	Initialises variables from 'rind.in'

RKCK	:	Evaluates next y for the suggested value of Δx and gives truncation error y_{err}
RKQS	:	Implementation of a fourth-order Runga-Kutta method with adaptive step size
TIME_DOMAIN_ODEINT	:	Time integration using a fourth-order Runga-Kutta method.
TIME_DOMAIN_ODEINT_FE	:	Time integration using a forward Euler method.

A.3 Common blocks variables

The variables that are used global are grouped in common blocks. Most of them are presented here in alphabetical order.

- **/ANGVEL3D/**

ANGVELX,ANGVELY,ANGVELZ	Angular velocity of the ship

- **/APERT/**

AX(I,J,K), AXN(I,J,K)	Current and old edge-aperture between cells (i,j,k) and $(i+1,j,k)$
AY(I,J,K), AYN(I,J,K)	Current and old edge-aperture between cells (i,j,k) and $(i,j+1,k)$
AZ(I,J,K), AZN(I,J,K)	Current and old edge-aperture between cells (i,j,k) and $(i,j,k+1)$
FB(I,J,K), FBN(I,J,K)	Current and old volume-aperture with respect to the geometry of cell (i,j,k)
FS(I,J,K), FSN(I,J,K)	Current and old volume of fluid function in cell (i,j,k)

- **/BAILOUT/**

halt	Boolean used for signalling premature program stop

- **/CALCPOSMATRICES/**

transmatrix	Predefined skeleton for a translation matrix in homogeneous coordinates
posvector	Current position in homogeneous coordinates

- **/COEFP/**

  The coefficient in the pressure Poisson equation

DIV(I,J,K)	The right hand side of the equation in cell (i,j,k)
CC(I,J,K)	Coefficient of $p_{i,j,k}$ in left hand side
CXL(I,J,K),CXR(I,J,K)	Coefficient of $p_{i\pm1,j,k}$ in left hand side
CYL(I,J,K),CYR(I,J,K)	Coefficient of $p_{i,j\pm1,k}$ in left hand side
CZL(I,J,K),CZR(I,J,K)	Coefficient of $p_{i,j,k\pm1}$ in left hand side

- **/DENSITY/**

density_of_water	Density of water in kg/m^3

Figure A.1: Calling sequence

- **/FEAB/**

`UNN(I,J,K)`	Old velocities in cell (i, J, K) in x-direction
`VNN(I,J,K)`	Old velocities in cell (i, J, K) in y-direction
`WNN(I,J,K)`	Old velocities in cell (i, J, K) in z-direction
`FEAB1, FEAB2`	

- **/FILLAR/**,**/FLUXAR/** and**/FRCAR/**
  The variables in these common blocks are used to create output for the post-processing.

- **/FLUID/**
  Characteristics of the fluid

`NU`	Kinematic viscosity number (ν)
`SIGMA`	Surface tension parameter
`THETA`	Contact angle RHO density

- **/FORCE/**
  Characteristics of the external and body forces

`AMPLX,AMPLY,AMPLZ`	Amplitude of the oscillation used in the subroutine **BDYFRC**.
`FREQX,FREQY,FREQZ`	Frequency of the oscillation used in the subroutine **BDYFRC**.
`GRAVX,GRAVY,GRAVZ`	Gravitational acceleration
`OMET(i)`	Rotational velocity of the tank
`XO,YO,ZO`	Point of rotation

- **/GRDSTR/**

`XCONC, YCONC, ZCONC`	Centre point of stretching
`XSTR, YSTR, ZSTR`	Stretch factors

- **/GRIDAR/**

`IMAX, JMAX, KMAX`	Number of grid points in x-,y- and z-direction.
`X(I), Y(J), Z(K)`	Coordinates of grid points
`DXP(I), DYP(J), DZP(K)`	Mesh size
`DXU(I), DYV(J), DZW(K)`	Mesh size

- **/KADE/**

`CanalLength`	Length of the canal
`CanalWidth`	Width of the canal
`CanalDepthRTLeftQuay`	Height of the left quay with respect to the canal bed
`CanalWidth`	Width of the left quay
`OSQuayWidth`	Width of the right quay
`OSQuayHeightRTLeftQuay`	Height of the right quay with respect to the canal bed
`WaterLevelRTLeftQuay`	Difference in height between the water level and the left quay.
`SimulateQuay`	Boolean that determines if the quay is added to the simulation

- **/LABELS/**

ULABEL(I,J,K), ULABFS(I,J,K)	Geometry label and free-surface label of velocity u between cells (i,j,k) and $(i+1,j,k)$
VLABEL(I,J,K), VLABFS(I,J,K)	Geometry label and free-surface label of velocity v between cells (i,j,k) and $(i,j+1,k)$
WLABEL(I,J,K), WLABFS(I,J,K)	Geometry label and free-surface label of velocity w between cells (i,j,k) and $(i,j,k+1)$
PLABEL(I,J,K)	Geometry label of pressure p in cell (i,j,k)
PLABFS(I,J,K), PLABFSN(I,J,K)	Free-surface label of pressure p in cell (i,j,k) at old and new time level.

- **/LDSV/**
  Variables used to make or read a restart file.

- **/LIQPAR/**

LIQCNF	Parameter that indicates whether you read the free surface from an input file or not.
LQXMIN, LQXMAX, LQYMIN, LQYMAX, LQZMIN, LQZMAX	Input parameters which indicates the region where the water is located.

- **/MARK/**
  Variables that define the object's geometry in a 2D simulation

XCOORD(N),ZCOORD(N)	x- and z-coordinates of each vertex in the object's geometry.
XCOORD_0(N),ZCOORD_0(N)	initial x- and z-coordinates of each vertex in the object's geometry.
NMARK	Number of vertices in the objects geometry

- **/MARK3D/**

MP,MPTOT	Current marker cell, total number of marker cells
IC(M),JC(M),KC(M)	(i,j,k) coordinate of the marker cell
XCM(M),YCM(M),ZCM(M)	Position of the marker cell in three coordinate directions
XHM(M),YHM(M),ZHM(M)	Geometry of the marker cell
VOL(M)	Volume of the marker cell

- **/MASS/**

mass_matrix_inverse	Predefined $6x6$ matrix, used for extracting acceleration and angular acceleration out of the total force and moments working on the object.
mass	Mass in kilograms

- **/MODES/**

`QuayMode`	When true the ships still has contact with the quay
`TippingMode`	When true the ship is tipping over into the water, but still has contact with the quay.
`UseComfloTimeIntegration`	When true use the forward Euler time integration method
`UseRK4TimeIntegration`	When true use the 4th-order Runga Kutta time integration method
`ConstantSpeedOnSlope`	When true the ship will have a constant speed on the slope
`PrettyPrintMode`	When true output information about forces/moments to the console

- **/MOMINERTIA/**

`Ixx`	Moment of inertia around the x axis
`Iyy`	Moment of inertia around the y axis
`Izz`	Moment of inertia around the z axis

- **/MVLAB/**

`MOVE(I,J,K)`	Set to -1 if the cell at (i, j, k) is part of a solid boundary and set to 1 if the cell at (i, j, k) is near the object

- **/MVMNTR/**

`XMNTRP_0(N),YMNTRP_0(N), ZMNTRP_0(N)`	Coordinates of the monitoring points.
`MVP`	Set to 1 if it's a moving monitoring point

- **/MUG/**

`mu`	Friction coefficient
`g`	Gravitation

- **/NUMER/**

`ALPHA`	Upwind parameter (0 central, 1 upwind)
`EPS`	Required accuracy of pressure equation
`OMEGA`	Actual relaxation factor of SOR
`OMSTRT`	Starting value of the relaxation factor of SOR
`ITMAX`	Maximum number of iterations for an iteration process
`ITER`	The actual value of the number of iterations
`NORMP`	Norm of the pressure
`ITTOT`	Total amount of iterations during the simulation
`DVL`	If set to 1 small holes in the fluid will be filled

- **/OBJMV/**

XCOM,YCOM,ZCOM	Position CG̈in x,y and z-direction
UCOM,VCOM,WCOM	Velocity CG̈in x,y and z-direction
AXCOM,AYCOM,AZCOM	Acceleration CG̈in x,y and z-direction
UCOMN,VCOMN,WCOMN	Velocity CG̈in x,y and z-direction at the previous time step
ANGVEL	Angular velocity C.G.
MVOBJ	Is set to 1 if it's a moving object
UOBJ(I,J,K),VOBJ(I,J,K), WOBJ(I,J,K)	Velocities of individual cells in the object
UOBJN(I,J,K),VOBJN(I,J,K), WOBJN(I,J,K)	Velocities of individual cells in the object at the previous time step
MVBXMIN,MVBYMIN,MVBZMIN, MVBXMAX,MVBYMAX,MVBZMAX	Boundaries in which the object can move
POS,VEL,ACC	

- **/OLDANGVEL/**

ANGVELN,ANGVELNN	Angular velocity at the old and older time level

- **/OLDANGVEL3D/**

ANGVELXN,ANGVELXNN	Angular velocity at the old and older time level around the x-direction
ANGVELYN,ANGVELYNN	Angular velocity at the old and older time level around the y-direction
ANGVELZN,ANGVELZNN	Angular velocity at the old and older time level around the z-direction

- **/OLDFB/**

FB_OLD(I,J,K)	Old volume-aperture F^b in cell (i,j,k)
AX_OLD(I,J,K)	Old edge-aperture A^x between cells (i,j,k) and $(i+1,j,k)$
AY_OLD(I,J,K)	Old edge-aperture A^y between cells (i,j,k) and $(i,j+1,k)$
AZ_OLD(I,J,K)	Old edge-aperture A^z between cells (i,j,k) and $(i,j,k+1)$

- **/PHYS/**

U, UN	The velocity in x-direction at old and new time level
V, VN	The velocity in y-direction at old and new time level
W, WN	The velocity in z-direction at old and new time level
P	The pressure

- **/SHIPFORCE/**

FORCEX,FORCEY,FORCEZ	Forces working on the c.g. in the x-,y- or z-direction
MOMX,MOMY,MOMZ	Moments working on the c.g. in the x-,y- or z-direction

- **/SLEIGH/**

H	Height c.g. relative to midship
HS	Height of the sleigh under midship
SLEIGHLENGTH	Full length of the sleigh

- **/SPACE/**

`DOMAIN`	Type of low domain (cube, cylinder, sphere, etc.)
`XMIN, XMAX`	Minimum and maximum x-coordinate of the mesh
`YMIN, YMAX`	Minimum and maximum y-coordinate of the mesh
`ZMIN, ZMAX`	Minimum and maximum Z-coordinate of the mesh
`OBJECT`	Type of object in the flow
`OBXMIN, OBXMAX`	Minimum and maximum x-coordinate of the object
`OBYMIN, OBYMAX`	Minimum and maximum y-coordinate of the object
`OBZMIN, OBZMAX`	Minimum and maximum z-coordinate of the object
`SLOSH`	Is set to one in the input file if there is a free surface
`MVBD`	Is set to 1 in the input file if the object can be displaced

- **/TIME/**

`TIME`	Current time
`NEXTTIME`	New time after 1 time step
`TIMESTEP`	Time step
`MAXTIME`	Maximum simulation time

- **/TIMES/**

`TMAX`	Maximum simulation time
`T`	Current time
`DT`	Time step
`DTMAX`	Maximum time step according to diffusive time step limit
`CYCLE`	Current time cycle
`CFL`	The *CFL*-number
`CFLCNT`	Number of times that `CFL` is less than `CFLMAX`.
`CFLQ`	If value is 1 then `DT` will be adjusted according to the *CFL*-condition
`CFLMIN`	Minimal *CFL*. If *CFL* ten successive time steps smaller, then time step is doubled
`CFLMAX`	Maximal *CFL*. If *CFL* is larger, then time step is halved.

A.4 Input files

To allow easy configuration of the many variables that COMFLO uses, the computer program uses input files. The main configuration files is 'rind.in'. It defines the overall geometry and characteristics of the ship, quay and canal. Below the `rind.in` used for simulation $2a$ is presented:

```
------ Initial situation ------------------------------------
Quai    Tippin. ComTI   RK4TI   constv  FrcPrnt
T       F       T       T       T       F
------ Gravitation ------------------------------------------
g
9.81
------ Canal -----------------------------------------------
length  width   depth.in.respect.to.left.quai(=0,0)
```

```
4.0     1.5     0.246
verst.  height
F       .1
------ Quai ---------------------------------------------------
Add to simulation?
T
------ Leftquai ----------------------------------------------
width
0.3
------ Rightquai ---------------------------------------------
width   height.in.respect.to.left.quai(=0,0)
0.1     0
dike    base    height
F       0.05    0.1
------ Slide (angles in degrees) -----------------------------
alpha   mu
4.0107  0
------ Sleighs -----------------------------------------------
height  width   simulate.sleighs?
0.032   0.47    F
------ Ship ------------------------------------------------
width   height  length  hgt.cg  hor.pos init.speed.allong.slide
0.47    0.228   1.76    0.162   0       0.3
mass
47.8
Ixx     Iyy     Izz (moments of inertia)
0.170   0.170   0.170
------ Water -------------------------------------------------
density
999.63
waterlevel.in.respect to.left.quai(=0,0)
-0.006
```

The original COMFLO input file `comflo.in` is still used, but the domain settings x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}, the object settings x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max} and the liquid settings lqx_{min}, lqx_{max}, lqy_{min}, lqy_{max}, lqz_{min}, lqz_{max} are overridden. The `comflo.in` input file has the following structure:

```
-------------------------------------------------------------------------
dim     cray    slosh   mvbd
2       0       1       1
-------------------------------------------------------------------------
domain  xmin    xmax    ymin    ymax    zmin    zmax    slip*
1       -0.3    1.6     -.5     .5      -0.246  0.4     0
-------------------------------------------------------------------------
object  xmin    xmax    ymin    ymax    zmin    zmax    slip*
```

```
6       -1.5    1.5     -1.5    1.5     0.5     3.5     0
-------------------------------------------------------------------------
-------------------------------------------------------------------------
position wavemaker:
wvmkr   nsrs    numbch   xpos    ypos    zpos   xlength ylength zlength
0       1       0        -80.0   0.0     -2.5   100.0   0.0     8.0
-------------------------------------------------------------------------
motion wavemaker:
n       freq    ampl    phase
1       1       1       0
-------------------------------------------------------------------------
-------------------------------------------------------------------------
liqcnf  lqxmin  lqxmax  lqymin  lqymax  lqzmin  lqzmax
1       -200.0  200.0   -10.0   10.0    -25.0   -0.006
-------------------------------------------------------------------------
rho*    nu      sigma   theta
1.0     1e-6    0.0     90.0
-------------------------------------------------------------------------
imax    jmax    kmax    xc      yc      zc      sx      sy      sz
160     1       59      0.0     0.0     0.0     1.0     1.0     1.0
-------------------------------------------------------------------------
eps     omega   itmax   alpha   orde4*  feab1   feab2   nrintp  exact*
1.0E-5  1.3     10000   1.0     0       1.0     0.0     5       0
-------------------------------------------------------------------------
dt      tmax    cfl     cflmin  cflmax  divl
.01     2.2.    1       0.15    0.4     1
-------------------------------------------------------------------------
gravx   gravy   gravz   ginrt   finrt
0.0     0.0     -9.81   0       0
-------------------------------------------------------------------------
amplx   freqx   amply   freqy   amplz   freqz   u0*     v0*     w0*
0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
-------------------------------------------------------------------------
omex    omey    omez    tup*    tdown*  x0      y0      z0
0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
-------------------------------------------------------------------------
load    nsave
0       10
-------------------------------------------------------------------------
npavs   tbavs*  comavs  npm2d   tbmatl* npm3d   nprnt   ntcom
0       0.0     0       27      0.0     0       40      0
-------------------------------------------------------------------------
npmslic nyz     nxz     nxy
0       0       0       0
planeyz
```

```
planexz
planexy
------------------------------------------------------------------------
avs pathname:
AVS-Data/
matlab pathname:
Data/
------------------------------------------------------------------------
nfillb  ntfill
4       500
xl      xr      yl      yr      zl      zr
0       0       0       0       0       25
-10     -10     0       0       0       25
-20     -20     0       0       0       25
-29     -29     0       0       0       25
------------------------------------------------------------------------
nfrcb   ntfrc
1       200
xl      xr      yl      yr      zl      zr
22.5    23.5     0       0      -0.5    0.5
------------------------------------------------------------------------
nfluxb  ntflux
0       0
xl      xr      yl      yr      zl      zr
------------------------------------------------------------------------
npartp  npartl  npartc  ntpart
0       0       0       0
xpt     ypt     zpt     tstrt                           <- points
xl      xr      yl      yr      zl      zr      tstrt   <- lines
xc      yc      zc      radius  orient  tstrt           <- circles
------------------------------------------------------------------------
nmntrp  nmntrl  nmntrc  ntmntr  mvp
5       0       0       5000    0
xpt     ypt     zpt                                     <- points
5.      0.      -3.
10.     0.      -3.
15.     0.      -3.
20.     0.      -3.
24.     0.      0.
xl      xr      yl      yr      zl      zr              <- lines
xc      yc      zc      radius  orient                  <- circles
------------------------------------------------------------------------
```

A.5 outputfiles

The COMFLO kernel and rind produce a lot of output during a simulation. COMFLO writes this output to the output files. The most important output files are listed below:

`dthist.dat`	contains data about the change in time steps, denoted by +1 if doubled or −1 if halved.
`fill*.dat`	contains data about amount of liquid and liquid percentage.
`forces.dat`	contains time histories of the forces in the three coordinate directions and moments around the three axes.
`iter.dat`	contains time histories of the cycle number, the number of iterations per cycle and the total number of iterations.
`kinetic.dat`	This file contains data about time, cycle number and the kinetic energy in that cycle.
`moment.dat`	contains the total moments around the cġfor each time step around each direction
`normal.dat`	contains time histories of the normal force working from the edge on the quay on the c.g.
`position.dat`	contains time histories of the position of the cġin three coordinate directions
`rotation.dat`	contains data about the rotation of the c.g. around three axes
`velocity.dat`	contains time histories of the velocities of the cġin three coordinate directions
`waterfrc.dat`	contains the water forces working on the cġin the three coordinate directions
`watermom.dat`	contains the water moments working on the cġaround the three axes

A.6 Postprocessing

Most of the output files from the previous section can be visualized using Matlab. For this reason a graphical user interface is built in Matlab. The most important menus are shown in figure A.2 (the main menu) and A.3 (the snapshot menu).

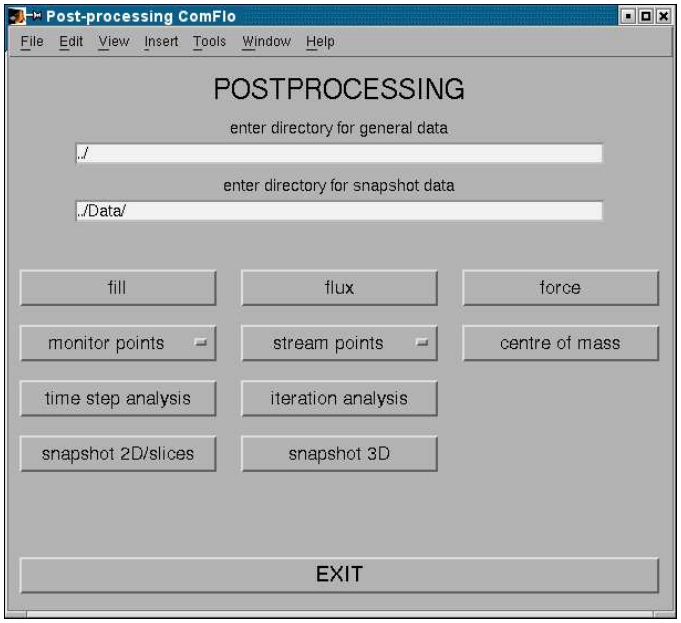

Figure A.2: The main menu

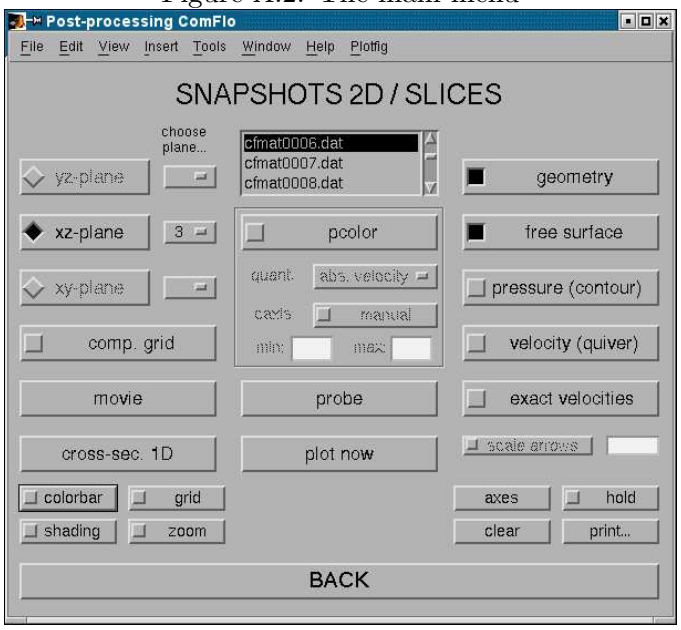

Figure A.3: The 2D snapshots menu

Appendix B

Scenario for recording the launch of a ship

This short manual is the result of experiences filming the launching of the "Suryawati" at the shipyard Bodewes-Volharding. This recording was the first of 4 attempts to create sufficient validation material for newly developed simulation software at the University of Groningen (Netherlands). This software will give the shipyard more precise parameters for launching its ships and minimizes failures.

Hardware needed

- 3 digital (mini-DV) camera's. At least one should have minimal the following specifications:
  - progressive recording
  - 3 ccd's
  - low f-number lens
- 1 stroboscope for synchronization or any other 'rather precise' method of synchronization
- 3 tripods
- 2 batteries per camera (both fully loaded)
- 3 mini-DV tapes
- at least 3 pair of Wellingtons
- 3 wireless communication devices

Preparations

Visit the shipyard at least one day before the actual launching to choose camera positions, place markings on the ship and measure the position of the ship on the quay.

You should use the best camera's to film bow and stern. Try to place the camera's precisely parallel to the quay (in a straight line from camera to the quays edge) on solid ground. In order to film the whole launching, try to get at least 5 times the ship's width on your view finder with the whole (vertical and horizontal) ship placed in a corner and the remaining 4 times of the ship's width (='water') in the other corner of your view finder. You can level the camera by zooming in on the wall of a building or checking the gauge on your tripod when available. The third camera should be located on the other side of the water at a sufficient distance to overview the whole scene. It should record the launching of the ship as well as the tidal wave crushing the shore. During the launch all camera's should be in a fixed position and focused on a fixed point to maintain a proper reference point. Besides this all camera's should not use any zoom to reduce any distortions of the lens. Avoid filming straight into the sun, because this could result in a unusable film.

During the whole launch, it's useful to have (football sized) reference points (in both horizontal as vertical direction) on the ship for tracking it's movement. Try to place markings on the bow (one at both sides) for horizontal reference. In vertical direction one could mark the tip of the mast (by fixing a waste disposal bag around it) and mark the top of the bow with a large sticker. The distance between these markings should be at least 5 meters. Write down these distances.

Try to measure the position of the ship to the edge of the quay (+distance in height), the distance in height from the quay to the water and the water depth.

Decide what kind of synchronization event will be generated. For example: after the launch, all camera's could turn and zoom in on a standing car or a stroboscope, which could then generate a light signal (with a duration of at least 0.12 seconds).

On the day of the launch

Arrive at least 2 hours before the actual launch. During this time you can get acquainted with the controls of the cameras and load the tapes. Check the battery status at all times. Take your positions at least 45 minutes before the launch, take the camera's out of their bags and place the camera's (let them acclimatize to the weather conditions). Start your recording 10 minutes before the launch and keep a close eye on your battery level and remove possible water droplets from the lens. See to it that all markings on the ship are visible during the recording and beware of tourists. After the launching, do the synchronization event and turn of your camera's. Write down the exact position and height of all camera positions (use known and mapped object for reference, e.g. sheds). These are needed for analyzing the data. Remove the tape from the camera and turn on their write-protection.

Information needed for analysis

- 3 mini-DV tapes with the recorded data
- type of recording: NTSC or PAL
- all camera positions (height,distance,angles) to a reference object
- map of the surroundings (with the water edge and with the objects used for reference)

- water height, distance in height from water to top quay.
- calculations and data (e.g. used ballast, center of gravity) for the launch
- position, height and length of the sliders
- position, height and length of the slides
- forces on each slider
- type of lubricant used between slider and slope

Bibliography

[1] Kendall E. Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, Inc., 2nd edition, 1989.

[2] E.F.F. Botta and M.H.M. Ellenbroek. A modified SOR method for the Poisson equation in unsteady free-surface flow calculations. *Journal of Computational Physics*, 60, 1985.

[3] G. Fekken. Numerical simulation of greenwater loading on the foredeck of a ship. Master's thesis, University of Groningen, August 1998.

[4] J. Gerrits. Three-dimensional liquid sloshing in complex geometries. Master's thesis, University of Groningen, June 1998.

[5] Ju. S. Jakovlev. Berekeningen en experimentele proeven van opdrijven, stabiliteit en tewaterlating (Een vertaling uit het Russisch). Technical report, 1947.

[6] E. Loots. Free surface flow in 3d complex geometries using enhanced boundary treatment. Master's thesis, University of Groningen, June 1998.

[7] Zdzislaw Meglicki. Advanced Scientific Computing, B673, February 2001.

[8] Chr. M. van Hooren. Dwarsscheeps te water laten, Berekening met behulp van een computer, Modelproeven. Technical Report 297-M, Laboratorium voor Scheepsbouwkunde, Technische Hogeschool Delft, January 1971.

[9] A.E.P. Veldman. Numerieke stromingsleer, march 1994.

[10] J. Versluis. Ontwerp modelopstelling voor systematies onderzoek dwarsaflopen van schepen. Technical report, Laboratorium voor Scheepsbouwkunde, Technische Hogeschool Delft, 1968.

[11] Ir. J. H. Vugts. The hydrodynamic coefficients for swaying, heaving and rolling cilinders in a free surface. Technical Report 194, Laboratorium voor Scheepsbouwkunde, Technische Hogeschool Delft.