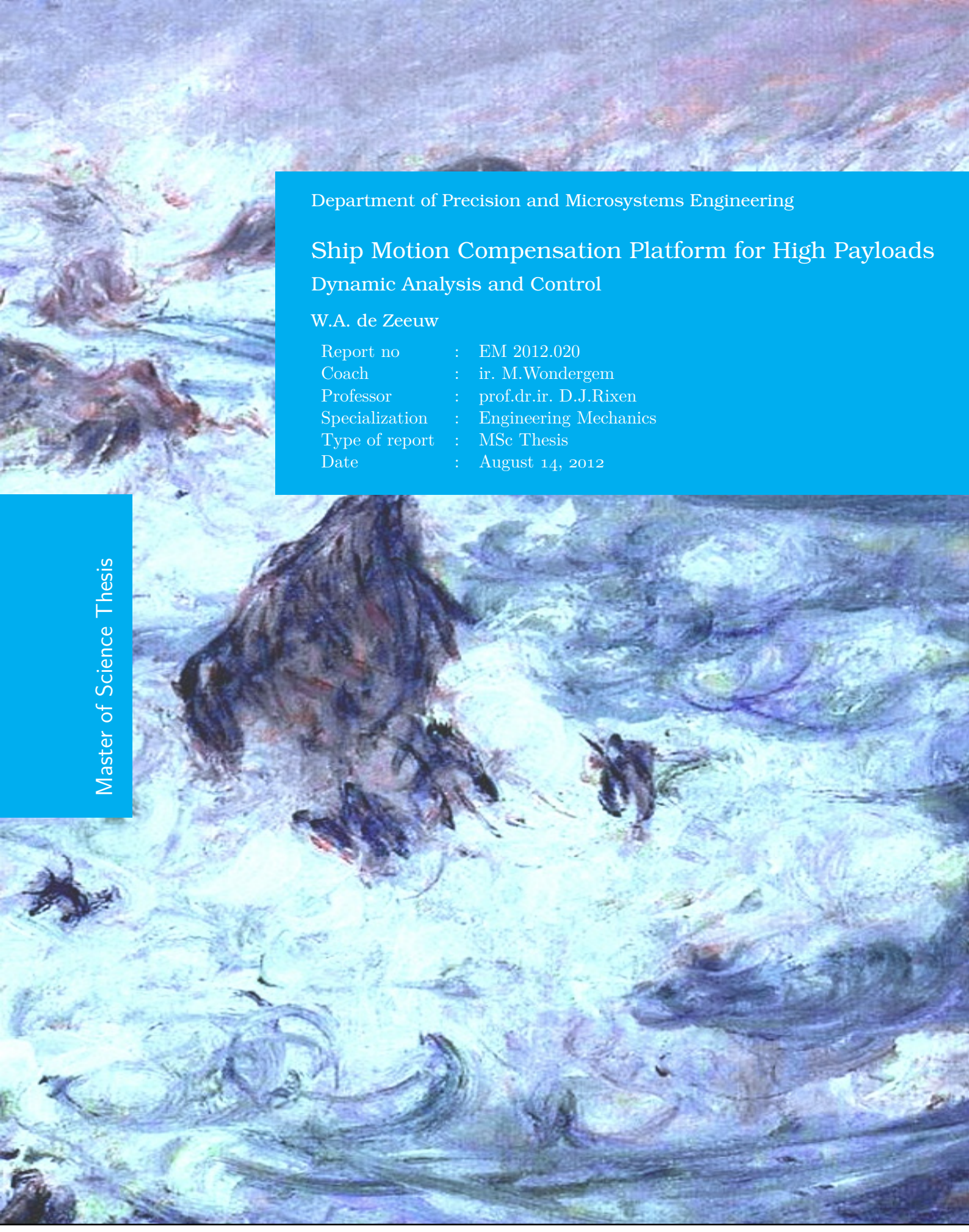

Department of Precision and Microsystems Engineering

Ship Motion Compensation Platform for High Payloads

Dynamic Analysis and Control

W.A. de Zeeuw

Report no	:	EM 2012.020
Coach	:	ir. M.Wondergem
Professor	:	prof.dr.ir. D.J.Rixen
Specialization	:	Engineering Mechanics
Type of report	:	MSc Thesis
Date	:	August 14, 2012

Master of Science Thesis

TU Delft Delft University of Technology

Challenge the future

Ship Motion Compensation Platform for High Payloads

Dynamic Analysis and Control

Master of Science Thesis

For the degree of Master of Science in Engineering Mechanics at Delft University of Technology

W.A. de Zeeuw

August 14, 2012

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work presented in this report was performed at, and supported by GustoMSC. The time and effort invested is very highly appreciated.

Abstract

When a ship motion compensation system carries very high payloads, in the size order of the displacement of the ship, the load dynamics and ship dynamics influence each other. What are these dynamics and how can an actuation system counteract or even benefit from it. Foundations to answer these questions are laid with a selection of literature focusing on non linear model predictive control, time domain hydrodynamics and motion compensation or generation mechanisms primarily of parallel robotic platform type. Next a physical scale experiment in which a heavy hexapod with quasi-static control is placed a ship is investigated. The instability shown in the experiments shows the severity and justify the topic. A planar model is estimated that quantifies sources of instability and shows that there are configurations of system parameters that result in instable behavior. Finally a new conceptual 3 degree of freedom platform mechanism is developed, and for the first time in literature a 3D coupled parallel robotic platform - ship dynamic simulation is derived. Two control techniques are applied, an unsophisticated quasi-static approach and a nonlinear model predictive control approach. Where for the latter real time capability with true online nonlinear optimization is achieved, which is also a first for this kind of systems. This model based approach outperforms the more naïve actuation scheme in stability and energy efficiency.

Cover: Detail of Claude Monet, *Tempête, côtes de Belle-Ile,* 1886, oil on canvas, 65,4 x 81,5 cm, musée d'Orsay, Paris. Photo: musée d'Orsay

Table of Contents

List of Figures

Acknowledgements

I would like to thank my graduation professor prof.dr.ir. D.J.Rixen and my supervisor at GustoMSC ir. M.Wondergem for their assistance during the writing of this thesis.

Hereby I would also like to thank the other graduation committee members: dr.ir. H.J. de Koning Gans, dr.ir. A.L. Schwab and dr.ir. D.J. Cerda Salzmann.

Delft, University of Technology W.A. de Zeeuw
August 14, 2012

Chapter 1

Introduction

Offshore to offshore load transfer is a common operation in the industry and especially common in the installation of offshore structures such as wind turbines. This transfer operation is in general sensitive to weather and sea state and therefore only possible at mild conditions. The waiting time for a so called weather window to allow for the operation to proceed is significant. As offshore time is expensive, all feasible projects to diminish it are of interest to the industry.

This thesis is focused on facilitating the open sea transfer of heavy components by means of an motion compensated platform. Actually heavy is kind of an understatement. The system has to be capable of carrying about 1000 metric tonnes with a center of gravity placed at about 60 meters above ground. This is about equivalent to all the cars parked at an Ikea establishment on an average Easter Sunday stacked on one big pile with a height equal to the length of a football field. Such a system would for example allow the installation of complete assembled wind turbines in a single lift. The crane could be placed on a fixed structure such as a jack-up unit Figure 1-1 and with the motion compensating system working as intended, this operation could perhaps even take place in storm conditions. Advantage of a stabilized platform as opposed to moving the components to the installation site on the jack-up itself is found in the unlimited operation water depth, as there is no ground contact, and higher possible travel speed as a dedicated component supply vessel could be employed with a hull optimized for traversing.

The main challenge in enabling such a system is found in the dynamics. As the loaded platform is a significant fraction of the total systems mass, the forces applied to the platform to compensate the movements, have reaction components that are high enough to influence the ship movements. This effect comparable to standing up straight in a rowing boat and trying to maintain balance. Pushing too hard with one leg rolls the boat to that side risking instability and a wet suit. As a result the dynamics of the forced platform and floating ship cannot be considered as two separate components. They have to be examined in a fully coupled framework.

All power used to drive the platform and its hydraulics must be developed on board. But Popey can eat only so many cans of spinach, therefore wherever possible handling the available

power efficient improves performance. Next to performance improvements, energy savings could also result in lower power requirements for the same compensation demand and thus result in a smaller (cheaper) system.

Overview of the thesis This work is divided in three chapters that discuss separate subjects and tackle the proposed problem. In Chapter 2 the theoretic foundations are laid upon which the latter derivations are based. This chapter cites and presents results from literature in the fields of Non Linear Model Predictive Control, a technique to incorporate model knowledge in an actuation scheme based on optimization, ship hydrodynamics used for time domain simulation and platform and other robotic motion compensating and generating mechanisms.

In Chapter 3 a documented movie of the scale model testing phase of the Ampelmann project is analyzed. In this experiment a relatively heavy platform is placed on a small boat. The interaction results in a resonating, unstable system. From this movie a linear model is estimated in order to obtain quantitative statements about the causes of this instability. The methods by which this model is estimated are non standard, as the model structure is fixed to a physical form. This application of the estimation theory and the method of obtaining quantitative statements about regions of stability are developed by the writer.

Chapter 4 derives the dynamics of a new kind of 3 degree of freedom motion compensating parallel platform coupled to a ship. An extension of the known literature of serial robots on non-inertial bases is accomplished by this parallel platform derivation. The forward and inverse kinematics of this new mechanism are found by methods known from flight simulator design theory, but extended by a instantaneous velocity solution of the linkages of this platform. The derived model is then used in a time domain simulation and a rather naive proportional integral derivative controller, based on the quasi-static assumption (the ship is fixed) is implemented. This controller is entered in a competition with an implementation of a true online optimizing non linear model predictive controller. Implementing a control strategy of this kind to a ship based robot has not been reported in literature.

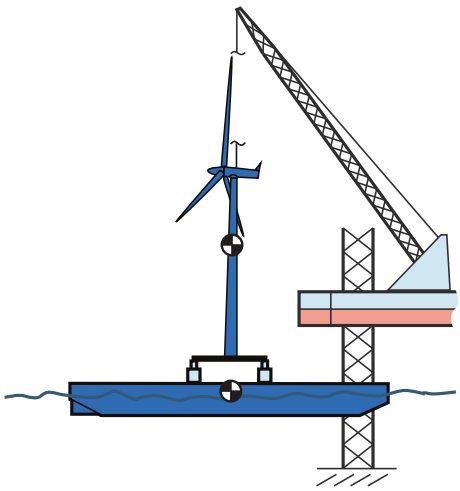

Figure 1-1: Visualization of ship motion compensating platform in action. A wind turbine that is placed on the platform is kept at a fixed position, therefore the crane placed on the jack-up unit can lift the structure gradually increasing tension in the main hoist.

Chapter 2

Foundations

A detailed literature and patent study is performed, focusing on three domains that each relate to the main goal: *enabling a ship motion compensation platform, stabilizing very high payloads*. Where high payloads should be understood as similar in order of magnitude to the displacement of the ship, so several hundreds of metric tonnes. The three domains are combined in one sentence as:

$$\underbrace{\textbf{Model predictive control}}_{\text{Section 2–1}} \text{ of a } \underbrace{\textbf{ship motion}}_{\text{Section 2–2}} \text{ compensating } \underbrace{\textbf{platform}}_{\text{Section 2–3}}$$

Each of these subjects is discussed in a separate section. In the "Model predictive control" section a selection of articles and books considering mainly nonlinear model predictive control is presented. This subject is chosen as a solution to our hypothesis that the similar mass motion compensation system requires more sophisticated control than standard proportional-integral-derivative (PID) control. In order to supply the controller with a useful prediction model, this model has to be obtained. The section "Ship motion" discusses modeling and some identification of ships for control purposes. The section "Platform" focuses more on the practical side and contains a patent search and a selection of (ship) motion compensating/-generating machines and mechanisms.

2-1 Model predictive control

2-1-1 Linear model predictive control

Introduction Mayne et al. (2000) [1] introduce the subject in their survey paper as: "Model Predictive Control (MPC) is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant." Ravi et al. [2] define

MPC a bit wider: "a family of control algorithms that employ an explicit model to predict the future behavior of the process over an extended prediction horizon."

The control signal is found via a minimization of an objective consisting of a combination of tracking performance and control effort called performance index or cost function. This methodology allows the controller to comply to constraints on the input (e.g. actuator force constraints) and on the predicted output, which is one of the virtues of MPC. The first control signal of the optimal control sequence found by the optimization is executed, and the optimization is re-initiated at the next time step. See Figure 2-1 for a graphical example of this "receding horizon" principle. As the control execution will always be performed by a computer and the optimization will require time, a discrete time formulation is a natural approach for MPC.

Standard Predictive Control Problem Boom [3] (Chapter 4&5) sketches a general mathematical framework to cope with linear model predictive control problems. Three performance indices are given: The Generalized Predictive Control (GPC) performance index, a quadratic form of output error and control increment (difference) signal, the Linear Quadratic Predictive Control (LQPC) performance index, a quadratic form of state and control signals, and the Zone performance index, a quadratic form of output error outside a set zone and control signals. These three are reformulated to fit inside the so called Standard Predictive Control Problem (SPCP) formulation. Quadratic performance index $\jmath$ if found by

$$\jmath = \sum_{j=0}^{N_m} \hat{z}_{k+j}^T \Gamma_j \hat{z}_{k+j} \quad \text{with} \quad \hat{z} = \begin{bmatrix} x & \text{or} & y-r \\ u & \text{or} & \Delta u \end{bmatrix} \tag{2-1}$$

$\hat{z}_{k+j}$ is a vector with a combination of <state (x) or output (y) and reference (r)> and <control level/effort (u) or differenced level (Δu)>. $\hat{z}_{k+j}$ is measured at the present time k and predicted for $j = 1$ to the prediction horizon (N_m). Γ_j is a selection matrix that allows for exclusion of certain samples from the performance index by having zeros instead of ones on the diagonal.[1] Performance index $\jmath$ at time k in Eq. (2-1) can be rewritten as a pure matrix formulation that replaces the sums over the predictions with larger matrix multiplications as $\jmath = \bar{\bar{z}}_k^T \bar{\bar{\Gamma}} \bar{\bar{z}}_k$. Where the extended matrices and vectors are marked with a double bar on top of the symbol.

At each time step an optimization problem in search of an control sequence that minimizes the cost function subject to the equality and inequality constraints (ϕ and ψ) is solved.

$$\begin{aligned} \jmath^* &= \min_{u_1..u_{N_c}} \jmath \\ \text{subject to} \quad & \phi = 0 \\ & \psi \leq \Psi \end{aligned} \tag{2-2}$$

The constraints can be functions of any of the system variables that could be included in $\hat{z}$ (Eq. (2-1)). The sequence of control signals for $j = 1$ to the control horizon N_c that results

[1] For example exclusion of the output in the first seconds of the prediction, which could be necessary if this part is already determined by the past control signals and no longer steerable. This particular example is called "dead time".

in the minimal cost is frequently denoted as u^*. Note that the control horizon N_c could be shorter than the prediction horizon N_m. The solution of the SPCP is given in [3] for the noiseless unconstrained case, the noisy unconstrained case, and the (in-)equality constrained cases. Shown is that for a Linear Time Invariant (LTI) system only the inequality constrained controller is nonlinear, the other controllers found from *the solutions of the* SPCPs *are* LTI *controllers.*

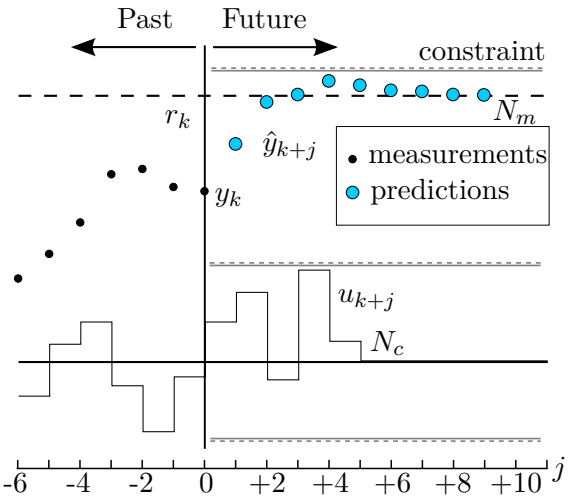

Figure 2-1: Receding horizon principle of model predictive control. u is the input signal, r is the reference and y is the output. At the present time k the first control signal is implemented and the entire horizon is shifted one sample and repeated. This principle is also called rolling optimization. N_m denotes the prediction horizon and N_c the control horizon. [3]

Properties There are several survey papers available on linear MPC that introduce the subject in depth, two frequently cited works are discussed. Mayne et al. (2000) [1] describes the development of the technology and searches for stability requisites. They sum three essential[2] ingredients: a terminal cost function, a terminal constraint set and a local stabilizing controller. The first two have to do with the predictions: the terminal cost adds a term to the performance index dependent on the situation at the prediction horizon, and the terminal constraint set can force the solution at the prediction horizon to be on the steady state value. The local stabilizing controller sets a local zone where a different controller takes over. The stability is theoretically proven via the direct Lyapunov method[3] and methods based on the monotonicity of the Lyapunov function. They discuss both continuous as discrete linear and nonlinear systems. Qin and Badgwell (2003) [5] gives next to a brief history of MPC, an overview of commercially available MPC technology, both linear and nonlinear, based primarily on data provided by MPC vendors (five). The main market for the technology is the process industry. The linear controllers work with a variety of linear state space, impulse and step response models, autoregressive models and transfer functions. And solve the

[2] Later these prove to be only sufficient, not necessary

[3] "For physical systems, one can often argue about stability based on dissipation of energy. The generalization of that technique to arbitrary dynamical systems is based on the use of Lyapunov functions in place of energy." - Åström and Murray [4]

(constrained) problem by Least Squares (LS) or by active set quadratic programming. The nonlinear software works with nonlinear state space, nonlinear neural net, and polynomial models and physics based so called first principle models. Optimization is typically done by nonlinear least squares, gradient based methods and Newton multi step algorithms. Important observation made by the writers is that most models are identified empirically and only one vendor is able to identify non linear models from first principles.

MPC is compared in several papers with other forms of control: Åström and Hägglund [6] discus the future of PID control and mention MPC as a companion/successor and praises its constraint handling. Sandino et al. [7] simulate a nonlinear model of a helicopter landing on a ship, which is in the same family of problems as ours, controlled by linearized MPC, Linear-quadratic-Gaussian (LQG) control and a loop shaping method. MPC outperforms the other methods but shows some oscillation in the control signal.

The true power becomes visible when the optimization is allowed to handle the full nonlinear system. Nonlinear Model Predictive Control (NMPC) is theoretically capable of steering a bike around a corner finding the solution that you first need to counter steer to the opposite direction. In Ge et al. [8] the term NMPC is not explicitly used but a full nonlinear system of a cat upside down in mid air, being able to flip over using minimal energy is solved.[4] This is done via a quadratic objective in control effort and a quasi-Newton optimization.

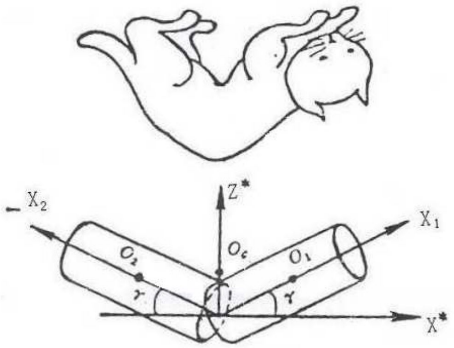

Figure 2-2: Nonlinear model based solution of a cat flipping over in mid air. Constrained by physics preventing a net momentum change. [8]

2-1-2 Non-linear model predictive control

Motivation The multiple input-output character of the boat-platform system due to the station keeping in several degrees of freedom, and the coupling between them, could be coped with by a variety of linear control techniques. Techniques like LQG, which is an algebraic solved control problem with a quadratic objective in state and control history, or H-infinity (H_∞) control, which actuation law is based on the feedback loop's maximum singular value[5], could take benefit of an available linear system model. But a feasible ship motion compensation

[4] The only difference with NMPC is the non-rolling optimization.

[5] The maximum singular value of the closed loop system is a robustness/model-uncertainty criterion. It should be as small as possible for maximum rejection to instability.

model should probably include several nonlinearities that are essential for accurate system representation, for example: the kinematics of the platform, the hydrodynamics frequency dependency and the hydrostatics. Perhaps even nonlinear actuator models. These system nonlinearities make NMPC strategies sensible candidates.

Basics nonlinear model predictive control NMPC is the nonlinear extension of it's linear brother. The basic scheme is shown in Figure 2-3. The system that has to be controlled, the plant, is nonlinear and therefore the prediction model inside the controller is also nonlinear.

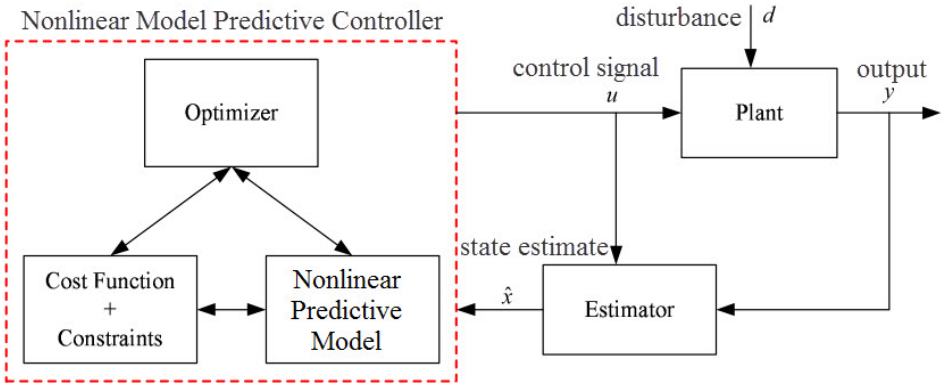

Figure 2-3: Basic structure nonlinear model predictive control. The controller has an internal model that is used to predict and optimize future outputs and find the corresponding optimal open loop control signals. [9]

There exist several surveys and books that are focused on NMPC that discuss its properties and difficulties. A selection is presented here: Henson (1998) [10] is one of the first to provide an overview of available NMPC technology and applications and proposes topics that required further research. He distinguishes fundamental/physics-based models from empirical models. Fundamental models are estimated efficient with little process data due to their constrained structure and are expected to remain valid in a broader range. Empirical models on the other hand allow for efficient formulations. The model can be constructed such that the optimization problem is convex and can be solved globally. Three solution approaches to the nonlinear optimization problem are mentioned: Successive linearization of the model equations, sequential model solution and optimization, and simultaneous solution of the dynamics and optimization. The writer suggests future research in algorithm development, in the stability and robustness area.

Schei and Johansen (1998) [11] deepen the subject of empirical modeling summarizing several nonlinear model types: fuzzy/multiple-linearized, Hammerstein[6], Volterra and Polynomial Nonlinear Autoregressive with Exogenous input (NARX) or Autoregressive Moving Average with Exogenous input (ARMAX) models. Muske and Rawlings (1993) [13] is referred in relation to the stability question. They derive a unified theoretical framework for a *linear infinite horizon controller that guarantees nominal stability regardless of weights in the cost function.*

[6]Hammerstein models consist of a nonlinear static mapping on the input state and then linear dynamics. Wei [12] discusses a lot of nonlinear model types in depth in Chapter 2. Including the output nonlinearty equivalent after the linear dynamics called Wiener model. His PhD thesis focuses on an efficient algorithm for simultaneous solution. The NMPC problem is changed to a 2 point (current time and prediction horizon) boundary value problem.

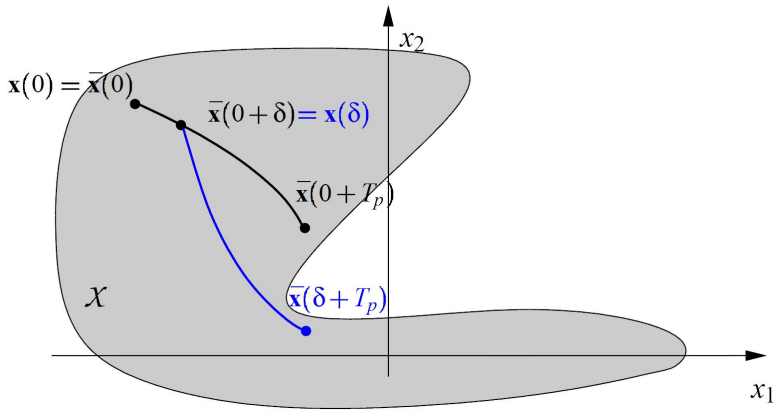

Figure 2-4: The difference between open-loop and closed-loop behavior in the finite horizon case. The optimal state trajectory at $t = 0$ results in a different optimum than the optimum found after re-optimizing at time $t = \delta$ for horizon T_p. The set of allowed states is given by χ. [9]

Stability developments Findeisen and Allgöwer (2002) [9] pick up where Henson stopped and discusses the stability of the (continuous) NMPC problem from a mathematical formulation. First the following important observation is presented: in a finite horizon formulation ($N_m < \infty$) the open loop optimal predicted trajectory, differs from the closed loop trajectory that results from closing the loop via the receding horizon principle by which the first of the sequence of optimal control signals is implemented. The difference in open and in closed loop trajectories is present even in the case of no model-plant mismatch, Figure 2-4. Therefore *stability (and optimality) in the open loop is no guarantee for closed loop stability (and optimality)*. Formulations that ensure closed loop stability are based on approximations of the infinite horizon problem by adding a penalty term (E) to the cost function that makes up for the missed part of the prediction and adding a terminal region constraint that forces the solution to steady state at the end of the prediction horizon. This terminal region for the state x is denoted by Ω.

$$\begin{aligned} \jmath(x, u|T_p) &= \int_t^{t+T_p} F(x(\tau), u(\tau)|T_p)d\tau + E(x(t+T_p)) \\ \text{subject to} \quad & x(t+T_p) \in \Omega \end{aligned} \tag{2-3}$$

Stage cost is denoted by F and is a function of state and control signal. The prediction and control horizon of the controller is T_p. The writers call this variant Quasi-Infinite Horizon NMPC (QIH-NMPC). Proofs are loosely based on using terminal penalty E as a local Lyapunov function of the system under a local controller $\kappa(x)$ in Ω. This results in the useful fact that feasibility in the open loop optimal control problem results in asymptotic stability in the closed loop system. In general κ, E and Ω are not easy to find. But if the nonlinear plant has a stabilizable Jacobian linearization[7] at the origin, a systematic approach to find the local controller, the terminal penalty and the terminal region is given in Chen and Allgöwer (1998) [14].

The last 10 years there was a fast development in research regarding NMPC stability. Less

[7]Nonlinear plant: $\dot{x} = f(x(t), u(t))$ has Jacobian linearisation $A \triangleq \frac{\partial f}{\partial x}(0,0)$ and $B \triangleq \frac{\partial f}{\partial u}(0,0)$. Which is stabilizable (controllable) if A is Hurwitz (stable) or if the controlability matrix $\begin{bmatrix} B & AB & .. & A^{n-1}B \end{bmatrix}$ is full rank.

restrictive stability requirements were derived. Grüne and Pannek's book (2011) [15] contains extensive mathematical proofs and results. In Chapter 4 it is derived that the infinite horizon optimal feedback law asymptotically stabilizes the system[8]. Their main result is shown in Chapter 6: Grüne and Pannek present a stability and suboptimality analysis for NMPC schemes *without stabilizing terminal constraints or penalties.* Stable control of this kind is valuable because the terminal conditions are a burden for efficiency and can compromise feasibility. As a spin-off they recover the well known[9] result that stability of the NMPC closed loop can be expected if the optimization horizon is sufficiently large.

Grüne and Pannek define the NMPC optimal open loop control problem (Algorithm 3.11 in [15]) in the following compact notation:

$$\boxed{\begin{aligned} &\text{minimize} \quad \jmath_N(n, x_0, u(\cdot)) \triangleq \sum_{k=0}^{N-1} \omega_{N-k}\ell(n+k, x_u(k,x_0), u(k)) \\ &\qquad\qquad + F(x_u(N,x_0)) \\ &\text{with respect to} \quad u(\cdot) \in \mathbb{U}^N_{\mathbb{X}_0}(x_0), \quad \text{subject to} \\ &x_u(0,x_0) = x_0\ , \quad x_u(k+1,x_0) = f(x_u(k,x_0), u(k)) \\ &\Rightarrow \text{solution optimal control sequence} \quad u^*(\cdot) \in \mathbb{U}^N_{\mathbb{X}_0}(x_0) \end{aligned}} \tag{2-4}$$

Where,
$\jmath_N$ performance index for finite horizon N.
ℓ stage cost, where the first argument $n+k$ is to allow time varying behavior.
$f(x,u)$ is the system state stepping function. (system dynamics)
x_u open loop solution trajectory resulting from initial state x_0 and control $u(k)$.
ω stage weights, F terminal cost, $\mathbb{X}_0$ the terminal state constraint set
and $\mathbb{U}^N(x_0)$ the set of allowed control signals from x_0 to N.

They define the optimal value function

$$V_N(x_0) = \min_{u \in \mathbb{U}} \jmath_N(x_0, u) \tag{2-5}$$

and apply the so called relaxed dynamical programming principle. This principle says that the same optimal trajectory should be found from all points along the prediction horizon.[10] It allows sub structuring the problem and allows the following inequality giving a posteriori *degree of suboptimality* $\alpha \in [0,1]$ *with respect to infinite horizon control.* [11]

$$V_N(x(n)) \geq V_N(x(n+1)) + \alpha\ell(x(n), \mu_N(x(n))) \tag{2-6}$$

[8] The nonlinear variant of the proof in [13]. It is shown that the infinite horizon optimal value function is a Lyapunov function for the closed loop system.

[9] A result from optimal control theory

[10] It shows that tails of optimal control sequences are again optimal control sequences for suitably adjusted optimization horizon, time instant and initial value.

[11] Proof. Positive running cost $\ell > 0$ gives positive optimal value function $V_N > 0$ and due to summation finite length optimal value is less or equal to infinite length $V_N \leq V_\infty$. Rearranging Eq. (2-6):
$\alpha\ell\big(x(n), \mu_N(x(n))\big) \leq V_N\big(x(n)\big) - V_N\big(x(n+1)\big)$
$\alpha \sum_{j=n}^{K-1} \ell\big(x(n), \mu_N(x(n))\big) \leq V_N\big(x(n)\big) - V_N\big(x(K)\big) \leq V_N\big(x(n)\big)$
$\alpha \sum_{j=n}^{\infty} \ell\big(x(n), \mu_N(x(n))\big) \leq V_N\big(x(n)\big)$
$\alpha V_\infty\big(x(n)\big) \leq V_N\big(x(n)\big) \leq V_\infty\big(x(n)\big)$

With $u = \mu_N(x)$ the optimal control law for horizon N. Then the infinite horizon cost under control μ_N is suboptimal to degree α as

$$\alpha V_\infty(x(n)) \leq \alpha V_\infty^{\mu_N}(x(n)) \leq V_\infty(x(n)) \tag{2-7}$$

Grüne and Pannek (2009) [16] find this degree of suboptimality a posteriori and a more conservative a priori estimate which allows adapting the prediction horizon beforehand.

α can be used to investigate stability. *Asymptotic stability can be concluded from suboptimality results if the running costs (stage costs ℓ) are positive and the degree suboptimality is positive* $\alpha > 0$.

Robustness and efficient calculation of the NMPC problem MPC possesses inherent robustness to model mismatch properties due to its resemblance to optimal control. Few explicit robust formulations exist [9], optimizing a feedback controller instead of a control (u) sequence is an example that is given. Findeisen and Allgöwer's discussion of the computational aspects results in the conclusion that on-line optimization is only possible via finite parametrization of controls and/or constraints. The sequential and simultaneous optimization are again [10] given as algorithm options. The large scale efficient simultaneous solution requires an additional constraint for every state node along the discretization in time to force a connected trajectory in the optimum. Downside is that only the finished optimization gives a feasible trajectory.

Wang and Boyd (2008) [17] describe a collection of methods to improve the speed of MPC using online optimization, as a contrary to calculating the entire control law offline and using a lookup strategy. Their three main techniques are: variable reordering to allow for efficient structured quadratic problems, warm-starting where the calculations for each step are initiated with the predictions made in the previous step, and early termination of the optimization. All three approaches show significant gains in speed. Note that for early termination, each optimization step should be at least feasible.

2-1-3 Applications of non-linear MPC

In this section several applications of NMPC are discussed. Chemical reactors due to extensive attention the literature gives to this application. Flight systems with their complex dynamical modeling. And the last two groups: robots/parallel manipulator platforms (a.o. Stewart type platforms) and ships because of the direct link to the goal of this work.

Chemical reactors The roots of MPC lay in the process industry. Therefore the lot of the applications documented are in chemistry. In literature reactors are the most popular use of constrained NMPC. Maximizing the production in the highly nonlinear polymerization process is discussed often in publications. Henson [10] refers to 20 papers on NMPC applications/simulation studies in chemical process industry. Magni et al. [18] applied a finite horizon NMPC controller with an auxiliary linear controller on model of nonlinear chemical reactor and arrive at a compromise between computational complexity and performance. Alamir et al. [19] discusses the computational verification and experimental implementation of nonlinear receding horizon control in a styrene polymerization reactor. Real time capability is enabled by

control parameterization in the form of a specific shape over the control horizon and penalizing the constraints in the cost function. Zhiying and Xianfang [20] controls a stirred tank reactor by GPC but estimates the model online. It uses online Support Vector Regression (SVR)[12] to determine a nonlinear model using recent model output data to track changes of a system model with time-varying and time-lagging characteristics. This is a method that origins from statistical machine learning and part of Support Vector Machines (SVM) theory. The nonlinear model obtained is linearized into an ARMAX form to obtain a convex open loop optimization.

Flight systems MPC has in the past only been applicable to systems with slow dynamics with characteristic times in the order of minutes. Due to increases in computing power and the development of numerical solution methods for NMPC, applications for systems with fast dynamics such as aircraft are looking more feasible. Khan et al. [23] proposes a NMPC strategy for unmanned air vehicles that can recover from actuator failure and continue to operate. Inherent system characteristics such as nonlinearities and cross-coupling effects can be exploited by the controller, rather than trying to minimize their influence. The dynamics are integrated via a Lagandre polynomial that is directly inserted into the cost function. The optimization of the control signals is solved by Sequential Quadratic Programming (SQP)[13] which allows a loop time as small as $1s$. The simulation results show that the controller allows adequate control authority to be returned to the aircraft in the event of a failure. Cheng et al. [24] derive a highly nonlinear model of a near-space hypersonic vehicle incorporating both engine as airframe dynamics. A time varying disturbance term is estimated by SVR and the total dynamical model is globally linearized via an input-output linearization and controlled by linear MPC. The model with the estimated disturbance term outperforms the nominal model showing less error and a smoother trajectory.

Robots/Parallel manipulator platforms Robotic manipulators are often categorized according to the number and type of Degrees of Freedom (DoFs) they possess. Robotic platforms are in general constructed as parallel manipulators with multiple kinematic chains, as opposed to serial manipulators such as robot arms, that consist of single chains. The DoFs of the end manipulator of a parallel manipulator, e.g. the platform surface, are linked to joint actuation via non linear kinematic transformations. A well known parallel manipulator platform is the "Stewart" platform. This is a 6 DoF robot with 6 actuated legs. An example is shown in the left photograph of Figure 2-5. A selection of publications that relate to the MPC of robots or platforms is given here.

Hedjar and Boucher [26]: Discusses the control a rigid link (no elasticity) serial robot arm with fast dynamics via NMPC. An approximation of a receding-horizon quadratic cost function, via Simpson's rule, allows for an analytical solution with one function evaluation. There is no need to perform an online optimization. The velocity is estimated by an observer as

[12] Zhao et al. [21] explains and applies SVR in combination with recursive subspace identification to identify a time varying Hammerstein model to use MPC. SVR identifies the nonlinear static part and the state space linear dynamics are estimated by singular value decomposition of the data Hankel matrix. Details are found in Overschee et al. [22]

[13] SQP is a nonlinear optimization strategy that first performs a Quasi-Newton step (approximated derivative terms) and then a line search in the direction of the quadratic solution of the first step. Constraints are typically implemented via penalty functions.

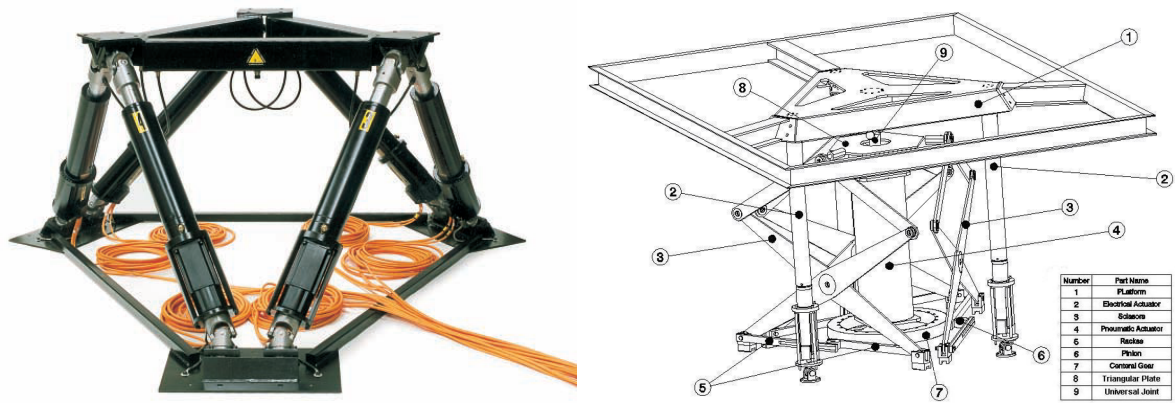

Figure 2-5: Left: Stewart type electric platform with 6 actuated legs mounted with universal joints, source: *http://www.boschrexroth.com* Right: 3 DoF platform with scissor type constraining mechanism, a central heaving cylinder and 3 cylinders that tilt the platform. [25]

measurements are too noisy. Robustness to model mismatch is enhanced by an integral action and asymptotic stability is proven via Lyapunov theory.

In Aminzadeh and Sabzehparvar [25][27] a configuration of a 3 DoF platform that allows heave, roll and pitch movement with scissors constraining the other DoFs is discussed. The dynamic relations are derived via the Newton-Euler method using Jacobian transformations to move from operational to joint space (leg lenghts).[25] The forward kinematics, that retrieve task space variables from joint space measurements, are solved by an iterative Newton-Rhapson scheme. The system is controlled by a model based controller via inversion of simplified dynamics in [27]. Control in task space outperforms the control in joint space due to the increase in error introduced by an additional transformation.

Koekebakker [28] wrote his PhD thesis on model based control of a 6 DoF flight simulator. He carefully models the Stewart platform including actuator inertia and a detailed hydraulics model. A systematical solution is presented for both the inverse and the iterative forward kinematics and the dynamics are derived. The control structure works with one inner loop per hydraulic leg and one outer loop that calculates the model based actuation signals. Additionally a detailed calibration method is explained that improves the precision by an order of magnitude.

Nadimi et al. [29] is one of the first that investigates the use of a GPC controller for the Stewart 6 DoF platform. The model is obtained by linearizing the MatlabSimMechanics toolbox benchmark Stewart platform model, and than stabilizing this system in two ways before applying MPC: first by velocity feedback, which proves hard to tune, and then by a LQG controller. The second method outperforms the first, but this conclusion might be affected by the settings. Nevertheless the control strategy appears suitable.

Ships and robots on ships This paragraph discusses the MPC of ships and robots on ships. The first two references are techniques to apply MPC to the control of a vessel, and the next five discuss control of cranes/robots/manipulators on ships and are of key importance to this thesis.

Zhang et al. [30] present the control of a submergence rescue vehicle. The model is mainly nonlinear in the state variables with coupling and higher order terms. The system state differential equation is split into a state and an actuation part, then approximated by a Taylor expansion and MPC controlled with an single step quadratic performance index with success.

Khan et al. [31] argue that due to the stochastic nature of the disturbances and the complex and ship specific dynamics it is preferred to use an artificial neural network to predict the ship motions instead of modeling the dynamics. The goal is to forecast 7 seconds of ship motions for launch and recovery of air vehicles such as missiles or helicopters. A practical experiment with about 10 minutes of ship roll data that is split into two thirds of training data and one third as validation set show a reasonable fit. Drawback of this method however is that forces induced by the air vehicle on the ship can not be included as their effects are not visible in the training set.

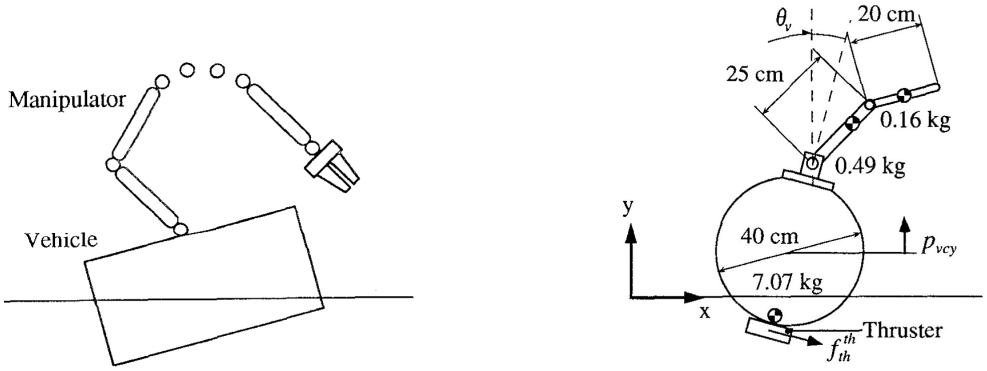

Figure 2-6: Robotic manipulator on ship utilizing restoring force. Left: [32] Right: [33]

An example of a well known manipulator control problem in offshore engineering is discussed by Kimiaghalam et al. [34]: The control of load swing in shipboard cranes. Their approach that decomposes of the crane model into linear state dynamics, and nonlinear static output mapping enables efficient use of MPC. They argue that the system dynamics, with a roll eigen-period of 10 seconds, are slow enough for live (real-time) optimization and show simulation results.

Instead of considering the buoyancy and ship movements as an enemy, Kosuge et al. [32] utilizes the buoyancy's restoring force/moment to reduce the number of actuators required to control a robot on a floating vehicle. A planar example of a serial robot on a ship with one horizontal thruster is discussed. The desired joint velocities are found via inversion of simplified dynamics. Kajita et al. [33] discusses this same vehicle manipulator system, and experimentally validates the reduction of actuator (thruster) force required for compensation of the robot's end effector reaction force. Figure 2-6.

Whereas in the last two references the actuation to compensate for the movements and reaction of the ship is performed on the ship, by means of thrusters, the last two writers discuss techniques that allow use the actuation of the robot itself to compensate for the ship movements. Love et al. [37] discusses the general modeling of robots operating on ships. They use the Denavit-Hartenberg matrix representation for the robotic joints and limit to serial

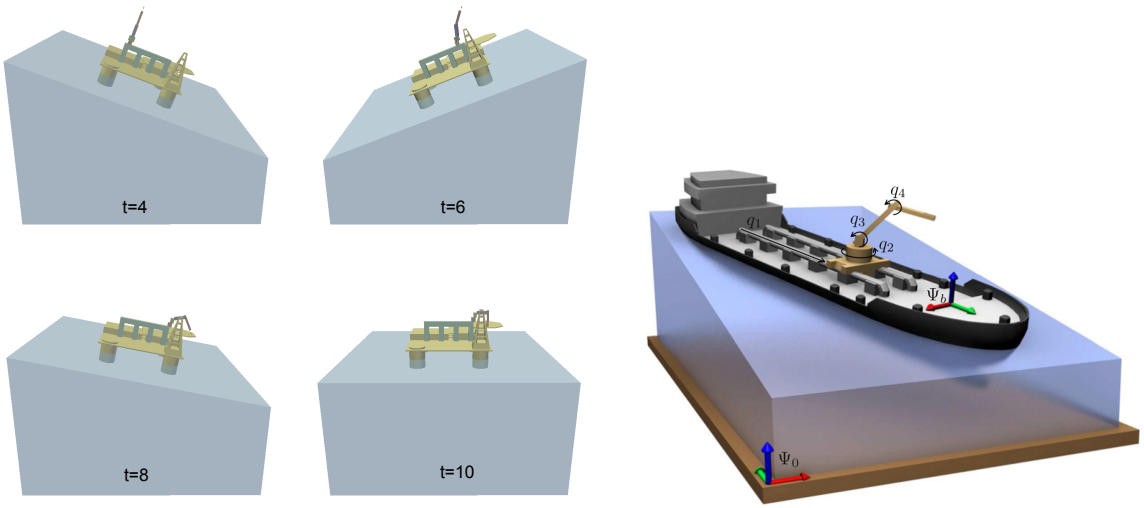

Figure 2-7: Robotic manipulator on ship/semisubmersible unit. Minimal torque solution snapshots in time. Left: [35] Right: [36]

(arm type) robots. This allows finding the end effector pose from a series of matrix multiplications that depend on the joint coordinates. Then, assuming constant and diagonal rigid body mass, they symbolical compute the Lagrange equations of motion and simulate the system imposing a base motion equal to a sea wave profile. This again reduces the problem to only force transfer from ship to actuator. The robot does not influence the base (ship) movements. Nevertheless the computations are useful, and a learning controller is able to control the robot arm. From created a series of publications regarding the control problem of robotic manipulators on ships.

From et al. [35] discusses the derivation of the equations of motion of a serial robot on a ship from a Lagrangian method based on partial derivation of the kinetic energy expressed via transformations in quasi-velocities. Quasi-velocities are not necessary time derivatives of their pose coordinates and allow nonlinearities. They arrive at dynamical relations that explicitly show the non inertial effects by means in terms of a Coriolis matrix. They use this knowledge to counteract these effects and use a MPC structure to control the system using minimal torque. In Figure 2-7 it is seen that the controller waits for translation of the robot to the right side until this side is lower. From et al. [38] discusses the boundedness properties of the mass matrix and the antisymmetric properties of the Coriolis matrix as these properties are used in stability poofs. The quasi-velocity vehicle-manipulator dynamics derivation satisfies both. The semi-velocity based method In From et al. [36] they adapt the prediction model in the controller to a stochastic estimation by means of an Auto Regressive (AR) model and a sum of sine waves model. Including a measure of uncertainty (a covariance matrix) into the performance index shows that the robot chooses "safer" paths, in the sense that they are more likely to be cheap.

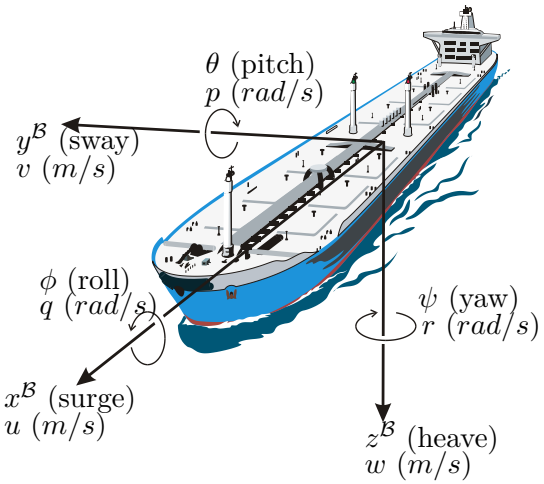

Figure 2-8: The ship direction and velocity nomenclature. The movements are defined in a ship based ($\mathcal{B}$) NED reference frame. Source: http://www.itk.ntnu.no/fag/gnc/Wiley/Ch2.pdf

2-2 Ship motion

In this section the necessary theory to generate a sufficiently realistic model for simulation and control purposes is gathered. First the nomenclature of ship movements is presented, then we start with the mixed frequency-time domain model and modify it to a pure time domain formulation appropriate for the stated goals and finally we approximate the radiation terms with a state space model.

Ship motions are generally described on a ship based frame $\mathcal{B}$ not necessary in center of gravity (CoG) with the x axis pointing forward, the y axis pointing to starboard and the z axis down. The rotation along this three axis is called "roll, pitch and yaw" respectively. This convention is chosen such that positive pitch rotation makes the bow move upward. See Figure 2-8. The forward, lateral and downward direction are named "surge, sway and heave".

2-2-1 Frequency to Time domain

Journée and Massie [39] describe the forced motion of a cylinder of mass m in the free surface of a water bassin as[14]:

$$\begin{aligned} z(t) &= z_a \sin t \\ (m + a(\omega))\,\ddot{z} + b(\omega)\dot{z} + cz &= F_a \sin(\omega t + \eta_{Fz}) \end{aligned} \tag{2-8}$$

Where z can be one of the DoFs of the ship, z_a is the amplitude of the forced oscillation. F_a and η_{Fz} are the amplitude and phase of the measured force. The components $a(\omega)$ and $b(\omega)$ are the frequency dependent added mass and damping. These coefficients can be found from potential theory and are written in matrix form for all 6 DoF as A(ω) and B(ω).

Potential theory assumes an ideal fluid (incompressible, inviscid, irrotational and without surface tension), steady state oscillation of the body, small velocities and amplitudes such

[14] [39]: Equation (6.51)

that the free surface conditions, the kinematic (water on the bodies boundary) conditions and the Bernoulli equation can be linearized. This allows splitting of the radiation (motion in undisturbed water) from the diffraction (wave excitation without movement) problem. The pressures are found from solution solving Bernoulli with the velocity potentials. Integration and in- and out- phase separation gives the mass synchronous and damping synchronous part of the force/moment and the added mass and damping can be found using the known mass, acceleration and velocity[15]. These so called hydro coefficients can be calculated using a hydrodynamic code (such as WAMIT). The calculation can be performed together with the Force Transfer Functions (FTFs) that find the wave loads for sinusoidal wave input.

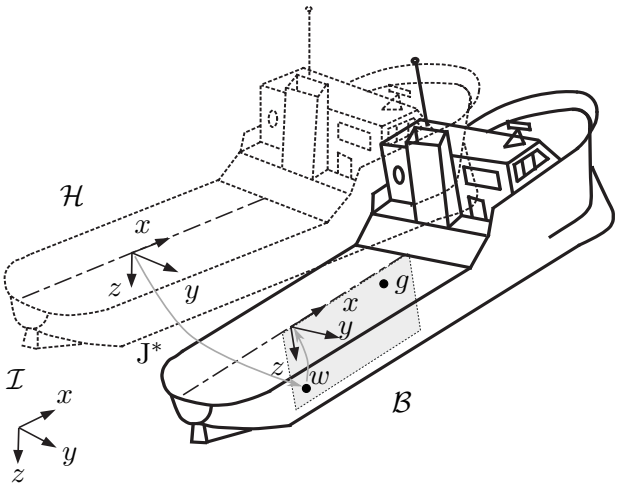

Figure 2-9: Seekeeping reference frame definitions. Ship fixed coordinate frame $\mathcal{B}$ on symmetry plane (not necessary in CoG) . Hydrodynamic/equilibrium reference frame $\mathcal{H}$ and inertial reference frame $\mathcal{I}$. [40]. J* transforms from the hydrodynamic frame through w, placed in the mean waterline on the ships symmetry plane (marked grey), to the ship frame $\mathcal{B}$. The ship's CoG is in g also on the symmetry plane. $z_{\mathcal{H}}$ is parralel to $z_{\mathcal{I}}$.

Cummins derived a pure time domain equation [41] expressed in the so called hydrodynamic equilibrium frame $\mathcal{H}$ that is placed in the mean waterline for a symmetric ship on the plane of symmetry (see Figure 2-9). This frame has coordinate vector $\bar{\xi}$ and the $\mathcal{H}$ frame aligned forces are in $\bar{\tau}^{\mathcal{H}}$. Using the infinite period added mass and damping matrices $\mathrm{A}(\infty) = \mathrm{A}_\infty$ and $\mathrm{B}(\infty) = \mathrm{B}_\infty$:

$$(\mathrm{M}^{\mathcal{H}}_{\mathrm{RB}} + \mathrm{A}_\infty)\ddot{\bar{\xi}}^{\mathcal{H}} + \int_{-\infty}^{t} \mathrm{K}(t-t')\dot{\bar{\xi}}^{\mathcal{H}}(t)dt' + \mathrm{G}\bar{\xi}^{\mathcal{H}} = \bar{\tau}^{\mathcal{H}} \tag{2-9}$$

With the retardation function $\mathrm{K}(t) = \frac{2}{\pi}\int_0^\infty \mathrm{B}(\omega)\cos(\omega t)d\omega$. Ogilvie rewrote the equation in a numerically better conditioned form, the Cummins-Ogilvie equation [42]:

$$(\mathrm{M}^{\mathcal{H}}_{RB} + \mathrm{A}_\infty)\ddot{\bar{\xi}}^{\mathcal{H}} + \mathrm{B}_\infty\dot{\bar{\xi}}^{\mathcal{H}} + \underbrace{\int_0^t \mathrm{K}(t-t')\dot{\bar{\xi}}^{\mathcal{H}}(t)dt'}_{\bar{\mu}^{\mathcal{H}}} + \mathrm{G}\bar{\xi}^{\mathcal{H}} = \bar{\tau}^{\mathcal{H}} \tag{2-10}$$

[15] Detailed derivation is presented in Chapter 7 of [39]

Now with the retardation function defined as $\mathrm{K}(t) = \frac{2}{\pi}\int_0^\infty (\mathrm{B}(\omega) - \mathrm{B}_\infty)\cos(\omega t)d\omega$. The vector $\bar{\mu}^{\mathcal{H}}$ is the damping term that is capturing the memory effects of the fluid. We recognize $\bar{\mu}$ as a convolution term with the velocity as input. $\mathrm{K}(t)$ is therefore the impulse response function. We used the assumption that the retardation is a causal function ($K(t<0)=0$), that is the effects of the delay are not present before to the velocity is/has been event, we shift the boundary of the convolution integral from $-\infty$ to 0.

2-2-2 Body frame

The dynamics of the ship are now found in the hydrodynamic frame, but we would like to know them in a body fixed frame. We now formally define the body fixed frame and transform the dynamics to this frame: $\mathcal{B}$ is placed on the deck of the ship with the x axis pointing forward, the y axis to starboard and the z axis down. The vector pointing from the origin of $\mathcal{B}$ to the CoG is denoted by $\bar{r}^{\mathcal{B}}_{bg}$ and the vector to the w is $\bar{r}^{\mathcal{B}}_{bw}$. Both of these vectors are fixed in the body frame and have a y component equal to 0.

The location and orientation of the ship relative to the inertial North-East-Down (NED) frame is described by the pose vector $\bar{\eta}$ consisting of three translations and three (roll-pitch-yaw) Euler angles. The generalized velocity in the $\mathcal{B}$ frame is denoted by $\bar{\nu}$. (notation according to [40])

$$\bar{\eta} = \begin{bmatrix} n & e & d & \phi & \theta & \psi \end{bmatrix}^T \in \mathbb{R}^3 \times \mathcal{S}^3 \tag{2-11}$$

$$\bar{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T \in \mathbb{R}^6 \tag{2-12}$$

The generalized velocity vector is not a straight forward time derivative of the pose vector. A transformation matrix is required. This transformation consists of a rotation and a so called attitude transformation. Construction is shown in Section 4-2-1.

$$R_{\mathcal{I}\to\mathcal{B}}(\phi,\theta,\psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\varphi s_\theta c_\psi - c_\varphi s_\psi & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & s_\varphi c_\theta \\ c_\varphi s_\theta c_\psi + s_\varphi s_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & c_\varphi c_\theta \end{bmatrix} = R^T_{\mathcal{B}\to\mathcal{I}} \tag{2-13}$$

$$[E'(\phi,\theta,\psi)]^{-1} = \frac{1}{c_\theta}\begin{bmatrix} c_\theta & s_\phi s_\theta & c_\phi s_\theta \\ 0 & c_\phi c_\theta & -s_\phi c_\theta \\ 0 & s_\phi & c_\phi \end{bmatrix} \tag{2-14}$$

The transformation from time derivative of pose to the body fixed velocities is now formed by:

$$\dot{\bar{\eta}} = \begin{bmatrix} R^T_{\mathcal{I}\to\mathcal{B}} & 0_{3\times3} \\ 0_{3\times3} & [E'(\phi,\theta,\psi)]^{-1} \end{bmatrix}\bar{\nu} = J_{\bar{\eta},\bar{\nu}}\bar{\nu} \tag{2-15}$$

Now to transform from the hydrodynamic equilibrium frame $\mathcal{H}$ to the boat frame $\mathcal{B}$ the transformation matrix J* from Fossen [43] is used.[16] Assuming small perturbations from the

[16] Matix J should not be confused with the scalar performance index J from MPC

equilibrium this matrix is constant with entries:

$$\mathrm{J}^* = \begin{bmatrix} 1 & 0 & 0 & 0 & z_w & 0 \\ 0 & 1 & 0 & -z_w & 0 & x_w \\ 0 & 0 & 1 & 0 & -x_w & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{I}_{3\times3} & -\begin{bmatrix} x_w & 0 & z_w \end{bmatrix}_\times \\ 0_{3\times3} & \mathrm{I}_{3\times3} \end{bmatrix} \tag{2-16}$$

Now the velocities and accelerations of the boat are now found from the equilibrium frame coordinates as:

$$\begin{aligned} \dot{\bar{\xi}} &= \mathrm{J}^* \delta\bar{\nu} \approx \mathrm{J}^* \bar{\nu} \\ \ddot{\bar{\xi}} &= \mathrm{J}^* \delta\dot{\bar{\nu}} \approx \mathrm{J}^* \dot{\bar{\nu}} \end{aligned} \tag{2-17}$$

Now premultiplying Eq. (2-10) with J^{*T} and substituting the coordinates via Eq. (2-17) gives the equations in a body fixed reference frame:

$$\underbrace{(\mathrm{M}^{\mathcal{B}}_{RB} + \mathrm{J}^{*T}\mathrm{A}_\infty\mathrm{J}^*)}_{\mathrm{M}^{\mathcal{B}}_B}\dot{\bar{\nu}} + \underbrace{\mathrm{J}^{*T}\mathrm{B}_\infty\mathrm{J}^*}_{\mathrm{D}^{\mathcal{B}}}\bar{\nu} + \underbrace{\int_0^t \mathrm{K}(t-t')\bar{\nu}(t)dt'}_{\bar{\mu}^{\mathcal{B}}} + \underbrace{\mathrm{J}^{*T}\mathrm{G}\mathrm{J}^*}_{\mathrm{G}^{\mathcal{B}}}\bar{\eta} = \mathrm{J}^{*T}\bar{\tau}^{\mathcal{H}} = \bar{\tau}^{\mathcal{B}} \tag{2-18}$$

The retardation function becomes $\mathrm{K}(t) = \frac{2}{\pi}\int_0^\infty \left(\mathrm{J}^{*T}\mathrm{B}(\omega)\mathrm{J}^* - \mathrm{J}^{*T}\mathrm{B}_\infty\mathrm{J}^*\right)\cos(\omega t)d\omega$. The radiation forces μ are now found in the body frame. In short the equations now look as:

$$\mathrm{M}^{\mathcal{B}}_B\dot{\bar{\nu}} + \mathrm{D}^{\mathcal{B}}\bar{\nu} + \bar{\mu}^{\mathcal{B}} + \mathrm{G}^{\mathcal{B}}\bar{\eta} = \bar{\tau}^{\mathcal{B}} \tag{2-19}$$

The right hand side forces from the equilibrium frame, such as wave forces, $\mathrm{J}^{*T}\bar{\tau}^{\mathcal{H}}_W = \bar{\tau}^{\mathcal{B}}_W$ also require a transformation from w to the origin of $\mathcal{B}$, opposed to for example the forces from the robotic platform. The forcing side can be expended by Taylor series in the state parameters or modulus (absolute valued) cross terms as given in Triantafyllou and Hover [44][17]. But these nonlinearities apply mainly to maneuvering problems with non zero forward speed thus only the hydrostatic components are included. Here they are written as linear matrix multiplications in G but they can be nonlinear functions of state.

2-2-3 State Space model of radiation forces

The main idea of this section is to replace the computationally expensive convolution terms in the equations of motion (in $\bar{\mu}$) with a state-space representation of low order. The convolution term but now from Eq. (2-9) (dropping the infinity added damping $\mathrm{J}^{*T}\mathrm{B}_\infty\mathrm{J}^*$) can be rewritten as:

$$\begin{aligned} \bar{\mu}^{\mathcal{B}} &= \int_0^t \mathrm{K}(t-t')\bar{\nu}(t)dt' \\ &= \int_0^t \left[\frac{2}{\pi}\int_0^\infty \left(\mathrm{J}^{*T}\mathrm{B}(\omega)\mathrm{J}^*\right)\cos\left(\omega(t-t')\right)d\omega\right]\bar{\nu}(t)dt' \end{aligned} \tag{2-20}$$

[17] Nonlinear vessel dynamics in Chapter 3 of [44]

We now rewrite this equation using indices and using the notation $b_{ij}^{\mathcal{B}}$ for the (i,j)th element of $\mathrm{J}^{*T}\mathrm{B}(\omega)\mathrm{J}^{*}$:

$$\bar{\mu}^{\mathcal{B}}(j) = \sum_i \mu_{ij}^{\mathcal{B}} = \sum_i \int_0^t \underbrace{\left[\frac{2}{\pi}\int_0^\infty \left(b_{ij}^{\mathcal{B}}(\omega)\right)\cos\left(\omega(t-t')\right)d\omega\right]}_{\mathrm{K}_{ij}} \nu_j(t)dt' \tag{2-21}$$

With a single component of the retardation function denoted by K_{ij}. We see that the computational cost of calculating this convolution in the time domain increase unbounded with the length of simulation. The system depends on the entire memory of velocities. Knowing that the retardation function dampens out, it is logical to attempt to simulate with a limited memory approximating $\mu^{\mathcal{B}}$ by $\hat{\mu}^{\mathcal{B}}$ in a state space realization. As suggested by Kristiansen and Egeland [45] and Kristiansen et al. [46]:

$$\begin{aligned} \dot{\bar{\chi}} &= \mathrm{A}_{rad}\bar{\chi} + \mathrm{B}_{rad}\bar{\nu} \\ \mu^{\mathcal{B}} \approx \hat{\mu}^{\mathcal{B}} &= \mathrm{C}_{rad}\bar{\chi} \end{aligned} \tag{2-22}$$

These last two references suggest identification in the time domain by calculating the step response function $\mathrm{K}_{ij}(t)$ (retardation) for discrete time steps and then applying singular value decomposition on the Hankel matrix. The Hankel matrix contains the values of the step response function for each row shifted one sample. Selecting certain (significant) components from the decomposition find the discrete time state space matrices. These have to be transformed to continuous time state space matrices to find the required state space realization of the radiation damping term. A practical approach to this method is discussed in Unneland [47] (pp.50-55). This method requires two approximating steps next to the identification: approximating the step response in time, and transforming from discrete time to continuous time. Pérez and Fossen [48] state that identifying the radiation terms from the frequency domain is superior due to this and some other reasons discussed briefly in the sequel.

Unneland also describes a frequency domain identification method for the state space matrices of the radiation terms. This method starts by considering the Cummins equation Eq. (2-9) in the frequency domain[18].

$$\left[-\omega^2[\mathrm{M}_{RB}^{\mathcal{H}} + \mathrm{A}(\omega)] + j\omega\mathrm{B}(\omega) + \mathrm{G}\right]\bar{\xi}^{\mathcal{H}} = \bar{\tau}^{\mathcal{H}} \tag{2-23}$$

Now the radiation forces can be expressed in the frequency domain as:

$$\begin{aligned} \bar{\tau}_{rad}^{\mathcal{H}} &= \omega^2\mathrm{A}(\omega)\bar{\xi}^{\mathcal{H}} - j\omega\mathrm{B}(\omega)\bar{\xi}^{\mathcal{H}} \\ &= \mathrm{A}_c(j\omega)[j\omega\bar{\xi}^{\mathcal{H}}] = \left[\frac{\mathrm{B}(j\omega)}{j\omega} + \mathrm{A}(j\omega)\right]\dot{\bar{\xi}}^{\mathcal{H}} \end{aligned} \tag{2-24}$$

The multiplication in the second equation is exactly the convolution with the retardation function in the time domain. Thus approximating the complex transfer function $\mathrm{A}_c(j\omega)$ with a LTI object (state space), results in the same result as directly estimating the retardation function K. We first write this transfer function in the reference frame of the barge by applying J^* from Eq. (2-16):

$$\bar{\tau}_{rad}^{\mathcal{B}} = \mathrm{A}_c^{\mathcal{B}}(j\omega)\bar{\nu}^{\mathcal{B}} = \left[\frac{\mathrm{J}^{*T}\mathrm{B}(j\omega)\mathrm{J}^*}{j\omega} + \mathrm{J}^{*T}\mathrm{A}(j\omega)\mathrm{J}^*\right]\bar{\nu}^{\mathcal{H}} \tag{2-25}$$

[18] pure frequency domain notation, not to be confused with Eq. (2-8)

Which has a known value for the discrete set of frequencies found via the hydrodynamic code. These are compared to the total radiation component (including infinite frequency added mass) of Eq. (2-9) transferred to the s domain via Laplace. Now the radiation component acting in direction k originating from a velocity component ν_l is aproximated with $\mathrm{K}(\omega)$ estimated by a rational transfer function with polynomial numerator $P(s)$ and denominator $Q(s)$:

$$\begin{aligned} \bar{\tau}^{\mathcal{B}}_{rad}(t) &= -\mathrm{A}_\infty \dot{\bar{\nu}} - \int_0^t \mathrm{K}(t-t')\bar{\nu}(t')dt \\ \tau_{rad_{kl}} \approx \hat{\tau}^{\mathcal{B}}_{rad_{kl}}(s) &= \left[\mathrm{A}_{\infty_{kl}} s + \frac{\mathrm{P}_{kl}(s)}{\mathrm{Q}_{kl}(s)}\right] \nu_l(s) \end{aligned} \tag{2-26}$$

Now the infinite frequency added mass, typically not available in 2D hydrodynamic codes, can be estimated in the same framework as the retardation by comparing the estimates from Eq. (2-26) with the known values from Eq. (2-25).[19] For each of the DoFs that shows coupling a rational transfer function can be estimated. The code from Perez and Fossen [49] employs a Gauss-Newton iteration for this. This gives the matrix of transfer functions as:

$$\mathrm{K}_{TF} = \begin{bmatrix} \frac{\mathrm{P}_{11}(s)}{\mathrm{Q}_{11}(s)} & \frac{\mathrm{P}_{12}(s)}{\mathrm{Q}_{12}(s)} & \cdots & \frac{\mathrm{P}_{16}(s)}{\mathrm{Q}_{16}(s)} \\ \frac{\mathrm{P}_{21}(s)}{\mathrm{Q}_{21}(s)} & \frac{\mathrm{P}_{22}(s)}{\mathrm{Q}_{22}(s)} & \cdots & \frac{\mathrm{P}_{26}(s)}{\mathrm{Q}_{26}(s)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\mathrm{P}_{61}(s)}{\mathrm{Q}_{61}(s)} & \frac{\mathrm{P}_{62}(s)}{\mathrm{Q}_{62}(s)} & \cdots & \frac{\mathrm{P}_{66}(s)}{\mathrm{Q}_{66}(s)} \end{bmatrix} \tag{2-27}$$

Conversion between LTI model forms (state space, polynomial, impulse) is discussed for example in [3] Appendix D and can be directly applied. The State-Space realization is non unique and scaled to improve numerical accuracy. Due to the relative degree 1 of the polynomials (number of poles n_p is one more than number of zeros n_z) there is no feed through D_{rad} matrix. Now the matrices from Eq. (2-22) can be found looking like:

$$\mathrm{A}_{rad} = \begin{bmatrix} \hat{\mathrm{A}}_{rad_{11}} & 0 & & \cdots & 0 \\ 0 & \ddots & & & \vdots \\ & & \hat{\mathrm{A}}_{rad_{kl}} & & \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & & 0 & \hat{\mathrm{A}}_{rad_{66}} \end{bmatrix} \quad \mathrm{B}_{rad} = \begin{bmatrix} \hat{\mathrm{A}}_{rad_{11}} \\ \vdots \\ \hat{\mathrm{A}}_{rad_{kl}} \\ \vdots \\ \hat{\mathrm{A}}_{rad_{66}} \end{bmatrix}$$
$$\mathrm{C}_{rad} = \begin{bmatrix} \hat{\mathrm{C}}_{rad_{11}} & \cdots & \hat{\mathrm{C}}_{rad_{kl}} & \cdots & \hat{\mathrm{C}}_{rad_{66}} \end{bmatrix} \tag{2-28}$$

With the dimension of dummy state $\bar{\chi}$ equal to the total number of poles in all transfer functions (typically about 2-4 per transfer function), and B_{rad} and C_{rad} both having one dimension equal to A_{rad} and the other to 6 (because of the 6 DoF).

[19] In this estimation we can use several facts about the retardation function and constrain the estimation to deliver feasible results: (proofs in [48])

- For zero and infinite frequency the retardation function is zero: → proper and zero(s) at $s = 0$
- Approaching $t = 0$ or $t = \infty$ does not make the retardation disappear or explode: → stability, relative degree of 1 (one more zero than pole)
- Passivity (no energy generation): → the retardation matrix is positive real.

2-3 Platform

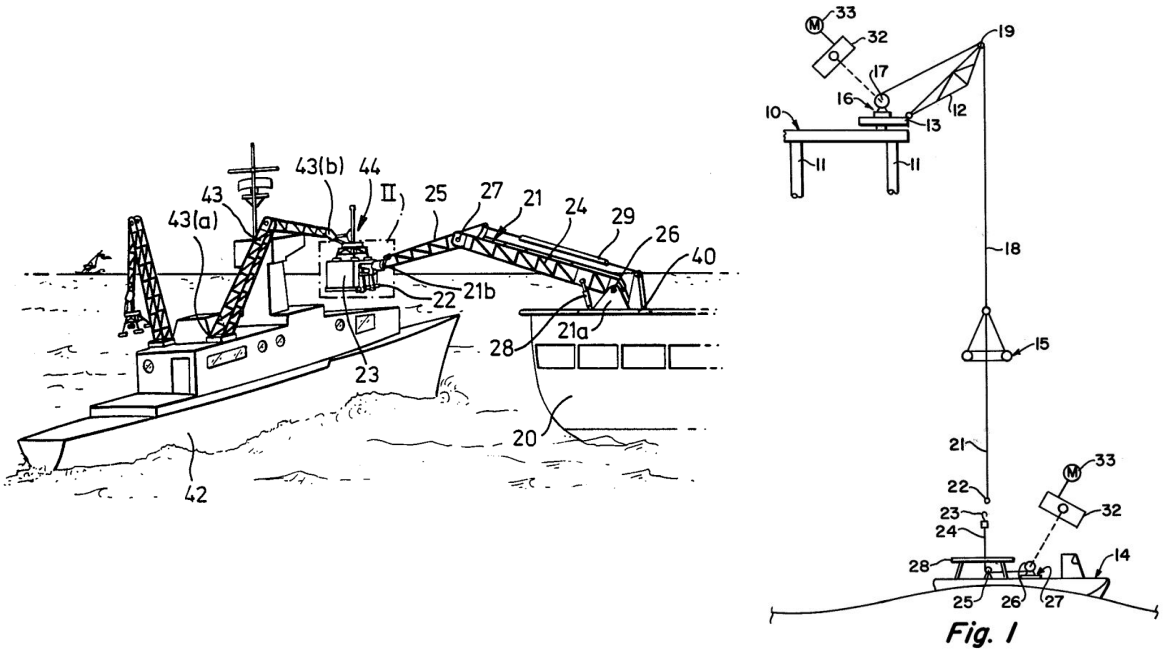

Figure 2-10: Open sea load transfer patent from 1989 [50] and from 1979 [51].

Open sea load transfer has been of interest for a long time. Methods of transloading cargo or passengers from one offshore structure to another, fixed of free floating, have been proposed and patented for over 30 years. Two dated patents are shown in Figure 2-10. "Open sea transfer of articles" [50] describes two robot arms handling load from one vessel to the next. And focusses mainly on the complex mechanical working principle of the end effector of the arms. "System to transfer cargo or passenger between platforms while undergoing releative motion" [51] is a cable bridge with a method of tensioning the wire to a constant stress and a carrier basket to perform the transfer.

Several mechanisms of ship based motion compensation platforms and systems are addressed in this section, where the last two are mechanism concepts that could be promising to scale up to use on a ship. For the parallel platform type compensators, the kinematic chains are inspected. The required degrees of freedom influence the choice of kinematic connections: Revolute joints (R) only permit one rotation, universal joints (U) permit 2 rotations (cardan type connection) and spherical (S) joints permit 3 rotations. For the hydraulics there is the decision of a axial constrained prismatic (P) cylinder and a free rotating cylindrical cylinder (C). The degrees of freedom can be counted by counting the number of bodies including the ground and substracting the constrained degrees of freedom by each joint ($R = -5$, $U = -5$, $S = -3$, $P = -5$, $C = -4$).

Ampelmann: [52] "Vessel, motion platform, method for compensating motions of a vessel and use of a stewart platform", 6 DoF hydraulic Stewart type platform. Universal-Cylindrical-Universal (U-C-U) configuration legs with internal static pneumatic gravity compensation. Designed for personel tranportation to and from for example offshore windturbines for maintainance, Figure 2-11
Bargemaster: [53] "Motion compensation device for compensating a carrier frame on a vessel for water motion", 3 DoF (roll-pitch-heave) load compensation platform with 3 vertically placed cylinders on spherical joints (S-C-S). Planar movements is constrained by 3 tension beams (U-R), with as a side effect limiting the maximal extention. This mechanism also incorporates quasistatic gravity compensation and is dimensioned to be easily containerized. Figure 2-12
Helipad: [54] "Helicopter landing platform having motion stabilizer for compensating ship roll and/or pitch", 1 DoF helicopter platform placed elevated from maindeck. Swaying motion of deck compensates the amplified roll motion of the ship experienced on the landing deck, Figure 2-14
Momac: [55] "Device for the safe transfer of personnel or material from an object configured as a boat to an object moving relative thereto", robotic arm (serial robot) for personnel transport. Can be equipped with a bridge or a basket, Figure 2-13
Pooltable: [56] "Motion Compensated Apparatus", 2 DoF (roll-pitch) compensated table. The table could be used as a pool table, but also as a bed or a operating theater, Figure 2-15
Ojdfell: 3 DoF (roll-pitch-heave) platform load/crane carrying platform with 3 2-linkage planar constraining mechanisms (Sarrus type) placed in a triangle configuration. The mechanism folds like a car jack and has a bottom to platform chain of R-R-S where the last spherical joint is a "Heim" type joint. The cylinder can be mounted in the scissors by a simple planar R-C-R connection, Figure 2-16
Royal Navy: 3 DoF (roll-pitch-heave) platform built for the royal navy with 3 Sarrus mechanisms constraining the planar movements again with a sperical joint at the platform. Difference is that the actuator is not a cylinder but a rotary push rod connection.Figure 2-17
Other: Two other concepts, a platform with a central heaving spring-cylinder, and a homemade flight simulator with a elastic "bungee cord" to compensate the gravity, Figure 2-18

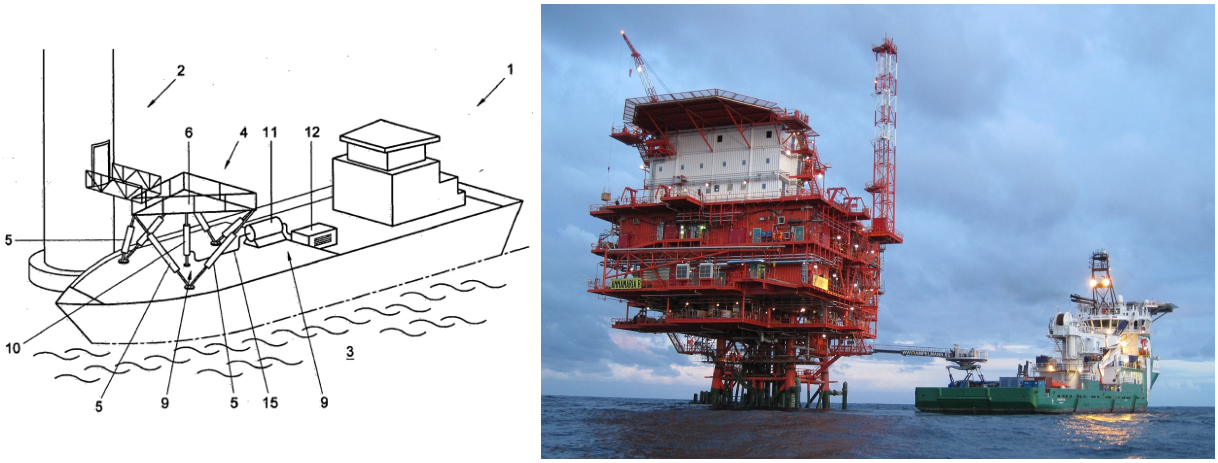

Figure 2-11: Graphics disclosed in Ampelmann patent (numbers are referenced in the patent) [52]

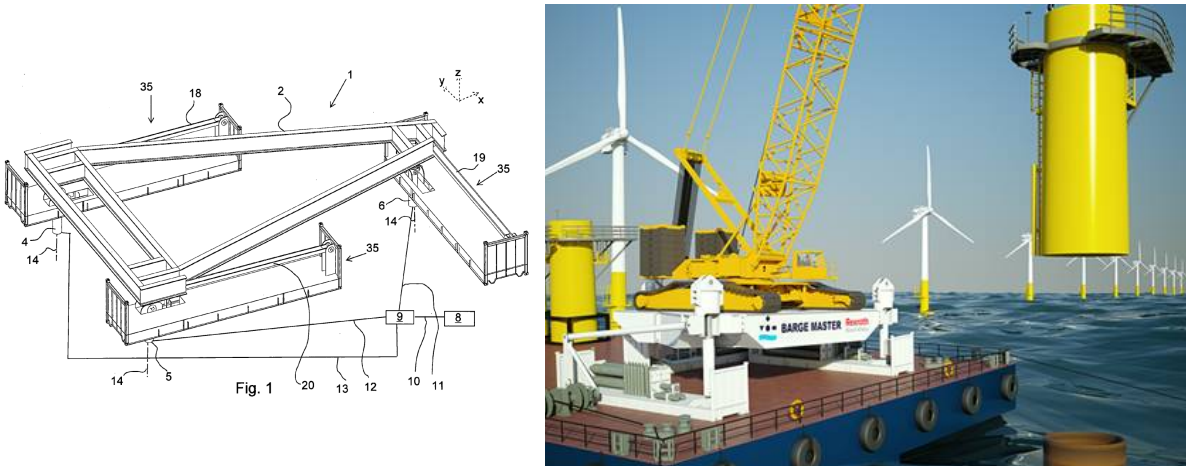

Figure 2-12: Graphics disclosed in Bargemaster patent (numbers are referenced in the patent) [53]

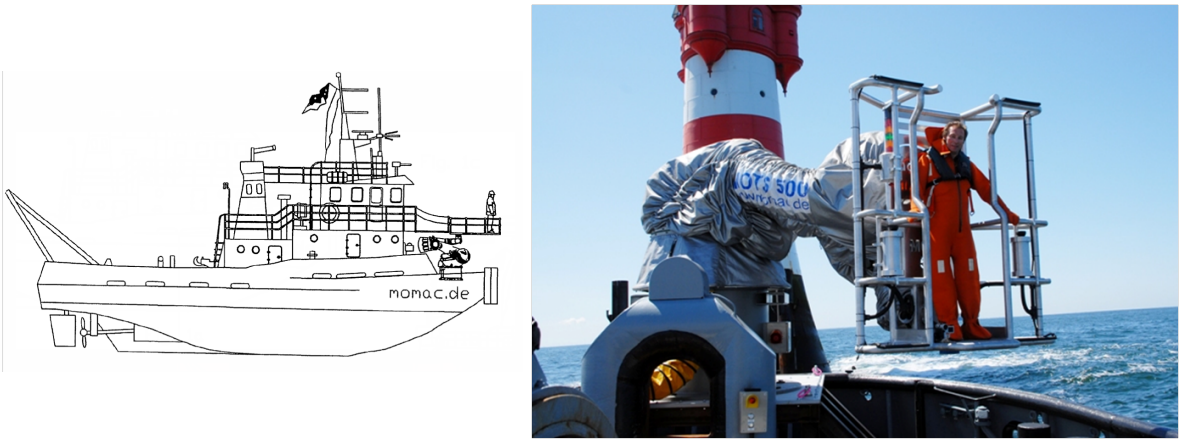

Figure 2-13: Graphics disclosed in Momac patent [55]

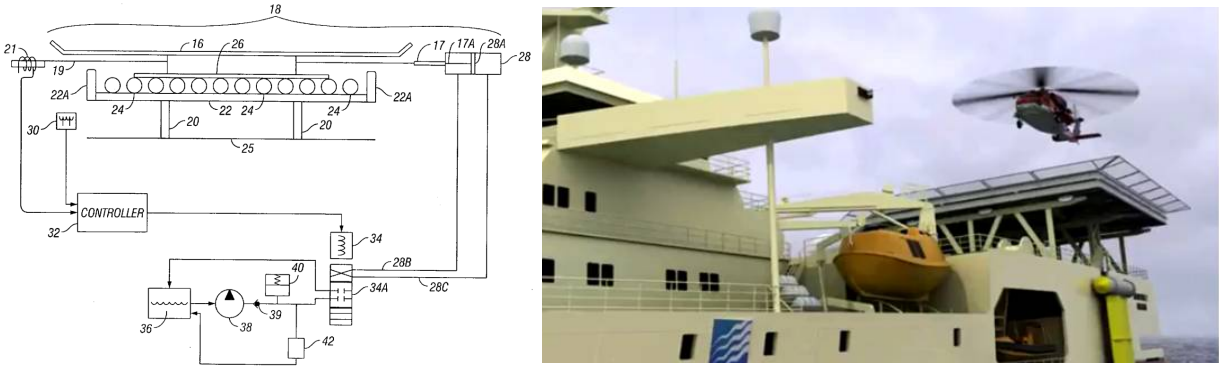

Figure 2-14: Graphics disclosed in the roll and/or pitch compensated helipad patent (numbers are referenced in the patent) [54]

Figure 2-15: Graphics disclosed in the stabilized billiard (or bed) patent (numbers are referenced in the patent) [56] and photograph of the "STable" on a "Radiance of the Seas" cruise ship.

Figure 2-16: Odfjell motion compensated platform Uptime on the Fob. Jr. Screenshots source: *www.youtube.com/watch?v=T8IujsZCRNc http://www.odfjellwind.com* (visited jun. 2012)

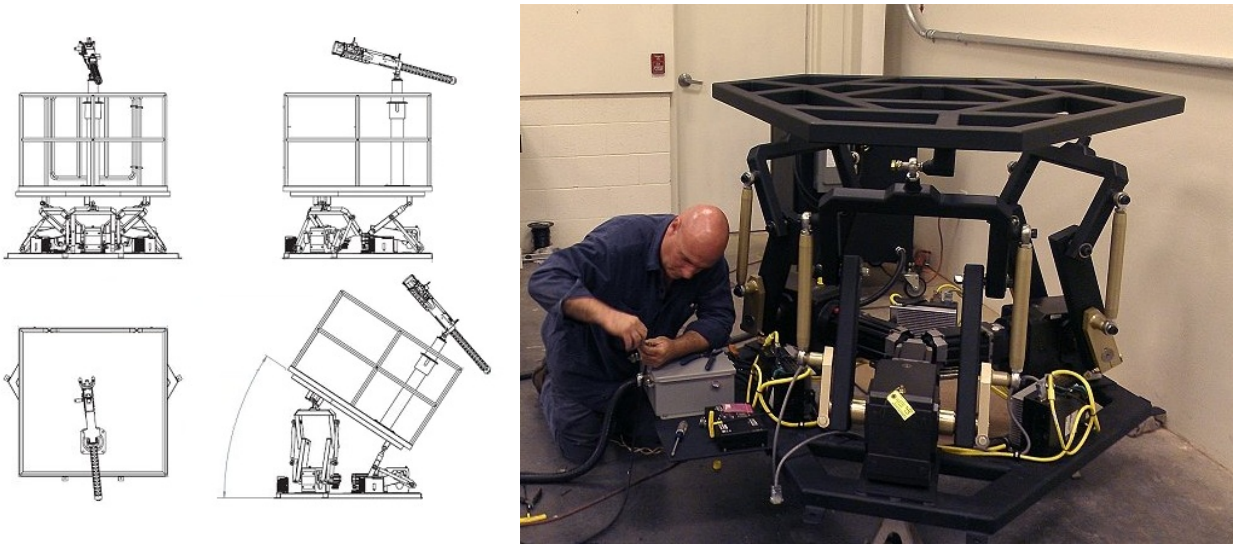

Figure 2-17: Sarrus type mechanism heave-roll-pitch platform designed for the royal navy. source: *http://www.inmotionsimulation.com* (visited jun. 2012)

Figure 2-18: Other concepts of roll-pitch-heave type platforms. The first row is a DIY flight simulator platform (source: *http://www.simprojects.nl*) with bungee cords to statically balance the payload. The second row is a platform type with big central spring/actuator to account for the heave (sources left to right: *http://www.inmotionsimulation.com*, *http://www.servos.com* and *http://www.intelmotion.com*) all visited jun. 2012

Chapter 3

Ampelmann scale tests - Quasistatic control

3-1 Introduction and scale model roll instability

The Ampelmann project and similar named company have resulted in a series of motion compensated platforms used for personnel transport to and from vessels. Their product is based on a hydraulic hexapod and its working principle is kind of similar to an inverted flight simulator. That is, not the cabin (platform) is moving but the foundation on which the hexapod is situated.

In the model phase of the Ampelmann project, the team faced stability problems in the water basin. The boat-platform assembly began a self induced oscillation in the roll rotational direction. This problem was carefully inspected and simulations in a multibody model, based on the Stewart Platform example from the Matlab SimMechanics toolbox, reproduced the phenomena (see Chapter 3.8 in [57]). It was found that the oscillation was related to the delay in the actuation, but a lag of zero did not solve the issue. The team suspected a relation to the inertia ratio of the platform and the ship and found that the system became more unstable if the top plate mass was increased [58], but it was not possible to stabilize the system by lowering the mass of the top plate. The ship mass was not altered as this is coupled to the stiffness and damping in a non trivial way. Their simulation also showed that if the hydrodynamic damping was increased the system became stable. In the model testing the keel of the ship was removed due to the limited depth of the water basin. This loss of roll damping was suspected to be the cause and roll dampers (plates) were welded to the sides of the ship. This solved the oscillation in the scale model [59]. For the full size model the mass of the platform was estimated to contribute only a factor 0.1% to 1% to the inertia of the ship-platform assembly. Therefore the problem was regarded to be not applicable to a full scale version.

Figure 3-1: Scale model testing of the Ampelmann, with roll dampers.

However considering that in our case the payload is not a couple of people but more close to the thousand fold of that, and the results of the 2D simulations performed in [60] showed stability problems the case is reopened here.

The hypothesis or suspected explanation of this roll instability is the fact that in controlling the platform, the ship's movements *due to* the control are ignored. The ship's position is considered as is, and the error regulated by the controller is the difference between leg length now and the leg length of the system with *the ship in the same position* and the platform in its target position. The ship is considered quasi static. For flight simulators this approach would certainly suffice as the actuator forces do not move the floor on which it is build, but for a ship-compensator system where the masses do not differ very much, this quasi static approach is expected to be insufficient.

In this chapter first a model experiment of the miniature Ampelmann on the ship without roll dampers is analyzed, than a linear model is identified from these experiments by means of fitting a parametric model, which is derived from theory and includes the suspected dominant effects. The found model is then interpreted and explained. The goal of this analysis is to understand where this instability comes from in order to prevent it from happening at a full scale system.

3-2 Movie analysis

The experiment with the scale model of the platform on the ship was filmed. The case without the roll dampers installed, and thus showing the oscillation, was recorded from the stern side of the ship and was available for analysis. From this movie the roll angles of platform and boat are estimated via computational image processing (see Figure 3-2) and plotted in Figure 3-3.

The maximal angle error due to limited image resolution is estimated as 1.5 degree. This is accurate enough for qualitative conclusions about the resonance and model fitting. The absolute value of the angles is probably a bit underestimated by the camera posed not exactly behind the stern. This is also a cause for asymmetry in the positive and negative roll angle. Also there are cables fed to the boat that could be providing some asymmetrical stiffness. The approximated roll angles are centered at their mean. Resulting in corrections of 5.9° and

Figure 3-2: Image processing of filmed experiments of an hexapod on a ship: Angle estimation

5.5° for the boat and the platform respectively.

In Figure 3-3 the results from the 307 frames (0.1Hz) that are measured are shown. The oscillation gradually increases in a rate that appears to be somewhere in between linear and logarithmic in time. At time $t = 15s$ the maximum amplitude is reached, and a limiting, or maybe long period beating, behavior is visible until the end of the movie at time $T = 30.6s$. The roll period is approximately $(24/30) = 0.8s$. The mean lead of the platform relative to the boat measured on zero crossing is $0.15s$ this is equivalent to a phase difference of about 67°.

3-3 Linear Model

In this section an attempt is made to explain the results observed from the experiment in the stability water tank by linear dynamics. The boat-platform assembly is represented by a 2 degree of freedom model. Only the roll angle of both is assumed of influence. Therefore the position vector is chosen as:

$$\mathbf{x} = \begin{bmatrix} \varphi_{platform} & \varphi_{boat} \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \tag{3-1}$$

For a damped fully coupled linear system there are eight unknowns, four in the damping matrix (C) and four in the stiffness matrix (K). The dynamics are given by Eq. (3-2).

$$\ddot{\mathbf{x}} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \mathbf{x} \tag{3-2}$$

Defining the state vector $\mathbf{x_s} = \begin{bmatrix} \dot{\mathbf{x}} & \mathbf{x} \end{bmatrix}^T$ gives the system matrix A_s (Eq. (3-3)) in a multiple output canonical form such that $\dot{\mathbf{x}}_\mathbf{s} = A_s \mathbf{x_s}$. All four entries are 4×4 matrices, I is the identity matrix and $\underline{0}$ represents the zero matrix.

$$A_s = \begin{bmatrix} C & K \\ I & \underline{0} \end{bmatrix} \tag{3-3}$$

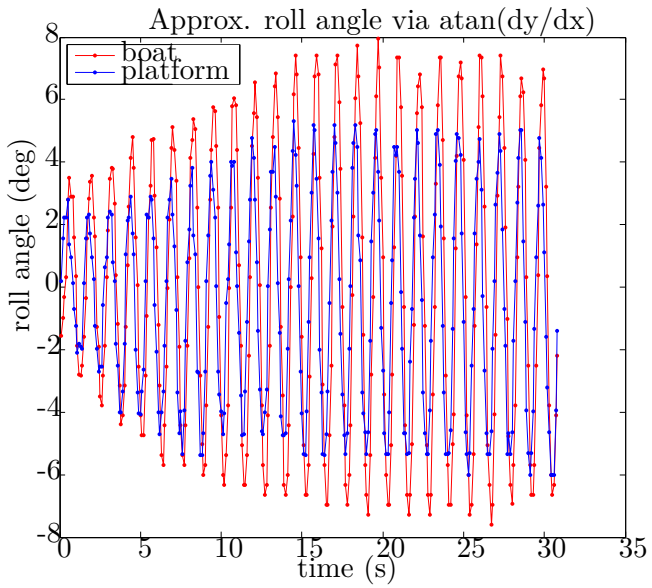

Figure 3-3: Roll angle tracking from video, development of oscillation in first 15 seconds, limiting behavior in second half. Notice the phase difference visible in the zero crossing: the platform leads the boat by about 67 degree. This value is estimated comparing the zero crossings of the dataset and averaging the result. (experiment: D. Cerda Salzmann)

The time forward solution of the initial value problem of the linear dynamics is given by Eq. (3-4). This solution has 12 variables, 8 in the system matrix, and four in the initial condition vector $\mathbf{x_0}$. The solution trajectories depend on the eigensystem of A_s. The eigenvalue problem for Eq. (3-3), $\det(A_s - \lambda I_{4\times4}) = 0$ is nonlinear in the 8 parameters and not trivially solved in general therefore the solution trajectories are no simple functions of the parameters.

$$\hat{\mathbf{y}}(t) = e^{A_s t}\mathbf{x_0} \tag{3-4}$$

3-3-1 Physics based model

In this section a linear dynamical model is derived from multibody dynamics. The two boxes in Figure 3-4 represent the platform's and the ship's rotational inertia. The springs and dampers (dashpots) are drawn as linear elements for clearness, but are torsional elements in interpretation. Equivalently when spoken of the masses, meant are the rotational inertias of platform and ship.

The proportional control spring is a spring with a free length ℓ_0 equal to the distance from the boat (x_2) to the reference (x_R): $\ell_0 = x_R - x_2$ When defining the error as Eq. (3-5) the actuation torque of the proportional control on the platform can be calculated as Eq. (3-6).

$$\varepsilon = x_R - x_1 \tag{3-5}$$

$$\tau_P = -k_P(x_1 - x_2 - \ell_0) = -k_P(x_1 - x_R) \tag{3-6}$$

For the derivative control dashpot a similar transformation is possible shown in Eq. (3-7) giving the torque on the platform.

$$\tau_D = -c_D(\dot{x}_1 - \dot{x}_2 - \dot{\ell}_0) = -c_D(\dot{x}_1 - \dot{x}_R) \tag{3-7}$$

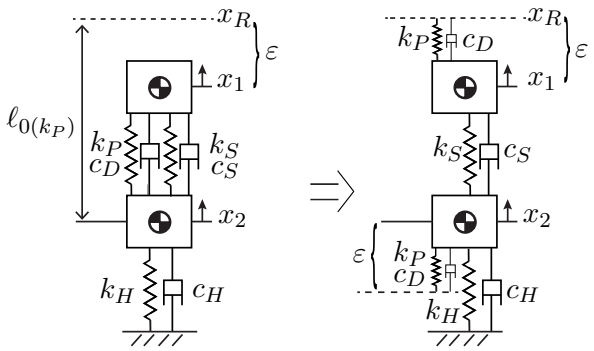

Figure 3-4: Linear model with proportional and derivative control x_1 (k_P and c_D) relative to the reference x_R, serial spring and damper (k_S and c_S), hydrostatic stiffness k_H and hydrodynamic damping c_H.

According to newton's third principle the control torques are felt in the opposite direction on the boat. The entire system including the hydrostatic, hydrodynamic and serial elements now becomes (Eq. (3-8)):

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \ddot{\mathbf{x}} = \begin{bmatrix} -(c_S + c_D) & c_S \\ (c_S + c_D) & -(c_S + c_H) \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} -(k_S + k_P) & k_S \\ (k_S + k_P) & -(k_S + k_H) \end{bmatrix} \mathbf{x} + \begin{bmatrix} c_D \\ -c_D \end{bmatrix} \dot{x}_R + \begin{bmatrix} k_P \\ -k_P \end{bmatrix} x_R \tag{3-8}$$

Or for with a non changing reference of zero ($x_R = 0$)[1]

$$\ddot{\mathbf{x}} = \begin{bmatrix} -\frac{(c_D+c_S)}{m_1} & \frac{c_S}{m_1} \\ \frac{(c_D+c_S)}{m_2} & -\frac{(c_H+c_S)}{m_2} \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} -\frac{(k_P+k_S)}{m_1} & \frac{k_S}{m_1} \\ \frac{(k_P+k_S)}{m_2} & -\frac{(k_H+k_S)}{m_2} \end{bmatrix} \mathbf{x} \tag{3-9}$$

This variant of the system suffers from an identification problem. It is known that a canonical form state space equation is unique, therefore Eq. (3-3) is unique. When the first and second matrix in Eq. (3-9) are compared to the first and second respectively in Eq. (3-2) we obtain the following equalities:

$$\begin{aligned} c_S &= c_{12} m_1 & k_S &= k_{12} m_1 \\ c_D &= -(c_{11} + c_{12}) m_1 & k_P &= -(k_{11} + k_{12}) m_1 \\ c_H &= \left(\frac{c_{11}}{c_{21}} c_{22} - c_{12} \right) m_1 & k_H &= \left(\frac{k_{11}}{k_{21}} k_{22} - k_{12} \right) m_1 \\ m_2 &= -\frac{k_{11}}{k_{21}} m_1 & m_2 &= -\frac{c_{11}}{c_{21}} m_1 \end{aligned} \tag{3-10}$$

This shows there is a free parameter in the parameterization. m_1 by the solution given in Eq. (3-10). For any given unique set of $k_{11}..k_{22}$ and $c_{11}..c_{22}$ the parameters of the physical

[1] Note that a non zero reference would require an additional term and for a changing reference a convolution term in the solution of the dynamics. Denoting $\begin{bmatrix} k_P & -k_P \end{bmatrix}^T x_R(t)$ as $g(t)$ constant reference $g(t) = g$ gives: $\hat{\mathbf{y}}(t) = e^{A_s t}\mathbf{x_0} + A_s^{-1}\left(e^{A_s t} - I\right)\begin{bmatrix} g & \vec{0} \end{bmatrix}^T$. For the varying reference case denoting $\begin{bmatrix} c_D & -c_D \end{bmatrix}^T \dot{x}_R(t)$ as $f(t)$, the solution is: $\hat{\mathbf{y}}(t) = e^{A_s t}\mathbf{x_0} + \int^t e^{A_s(t-\tau)} \begin{bmatrix} g(\tau) & f(\tau) \end{bmatrix}^T d\tau$

model of Figure 3-4 are unique if one parameter of these parameters is fixed beforehand. So estimated state space of Eq. (3-9) is unique if one mass (or any other value of an element in Figure 3-4) is given. *The top plate assembly is estimated as* $m_1 \triangleq 20kg$. *This estimate is used in the physics based fits.*

3-3-2 Stability

No serial terms Stability of a continuous system is guaranteed if the eigenvalues of A_s are distinct and real part of the eigenvalues is non positive. If the serial components in the Eq. (3-8) are zero, and the reference $x_R = \dot{x}_R = 0$, the system reduces to:

$$\ddot{\mathbf{x}} = \begin{bmatrix} -c_D/m_1 & 0 \\ c_D/m_2 & -c_H/m_2 \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} -k_P/m_1 & 0 \\ k_P/m_2 & -k_H/m_2 \end{bmatrix} \mathbf{x} \tag{3-11}$$

The system matrix becomes (with the mass normalization by m_1 and m_2 for improved readability denoted by subscript 1 resp. 2.):

$$A_s = \begin{bmatrix} -c_{D1} & 0 & -k_{P1} & 0 \\ c_{D2} & -c_{H2} & k_{P2} & -k_{H2} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-12}$$

The eigenvalues λ are found via the solution of the characteristic equation:

$$\begin{aligned} 0 =& \det |A_s - I\lambda| \\ =& \lambda^4 + (c_{D1} + c_{H2})\lambda^3 + (c_{D1}c_{H2} + k_{P1} + k_{H2})\lambda^2 + (k_{P1}c_{H2} + k_{H2}c_{D1})\lambda + k_{P1}k_{H2} \\ =& (\lambda^2 + c_{D1}\lambda + k_{P1})(\lambda^2 + c_{H2}\lambda + k_{H2}) \\ & \lambda_{1,2} = \frac{1}{2}(\pm\sqrt{c_{D1}^2 - 4k_{P1}} - c_{D1}),\ \lambda_{3,4} = \frac{1}{2}(\pm\sqrt{c_{H2}^2 - 4k_{H2}} - c_{H2}) \end{aligned} \tag{3-13}$$

Two demands for stability are found from the eigensolution.

1. Non positive real part: $\mathbb{R}(\lambda_j) \leq 0 \Rightarrow c_{D1} > 0, k_{P1} > 0, c_{H2} > 0, k_{P2} > 0$
2. Separation complex part: $c_{D1}^2 - 4k_{P1} \neq c_{H2}^2 - 4k_{H2}$

This second condition is needed as a marginal stable system with two identical (complex only) eigenvalues is unstable. A repeated pole on the imaginary axis makes the system unstable. The first condition is naturally fulfilled as *the masses, damping and stiffness are nonnegative.* The second condition, $(\frac{c_D}{m_1})^2 - 4(\frac{k_{P1}}{m_1}) \neq (\frac{c_H}{m_2})^2 - 4\frac{k_H}{m_2}$, is fulfilled if the *damped eigen frequencies of the two bodies are well separated.*

Time delay In order to investigate if a delay in the control action the following linear single time delayed system, with lag τ, is proposed:

$$\dot{x}(t) = A_0 x(t) + A_d x(t - \tau) \tag{3-14}$$

This system Eq. (3-14) is stable if: 1. the system $A_0 + A_d$ is stable (zero time delay stability, $\tau = 0$) and 2. the matrix $\mathcal{A}$ (Eq. (3-15), see [61]) is non-singular for all complex numbers s with a positive real part ($\mathbb{C}_+$):

$$\mathcal{A}(s, \tau) = [sI - A_0 - A_d e^{-s\tau}] \tag{3-15}$$

If the controlled part from A_s (c_D and k_P) is placed in the delayed A_d and the hydrodynamical part into A_0 such that $A_s = A_0 + A_d$, using again subscripts for mass standardization we get:

$$\mathcal{A} = \begin{bmatrix} s + c_{D1}e^{-\tau s} & 0 & k_{P1}e^{-\tau s} & 0 \\ -c_{D2}e^{-\tau s} & s + c_{H2} & -k_{P2}e^{-\tau s} & k_{H2} \\ -1 & 0 & s & 0 \\ 0 & -1 & 0 & s \end{bmatrix} \tag{3-16}$$

for singularity/instability it is required that the determinant is zero:

$$\det |\mathcal{A}| = 0 \tag{3-17}$$

This gives the following solution for s:

$$\begin{aligned} s_{1,2} &= \frac{1}{2}\left(\pm\sqrt{(c_{D1}e^{-\tau s})^2 - 4k_{P1}e^{-\tau s}} - c_{D1}e^{-\tau s}\right) \\ s_{3,4} &= \frac{1}{2}\left(\pm\sqrt{c_{H2}^2 - 4k_{H2}} - c_{H2}\right) \end{aligned} \tag{3-18}$$

The square root is always smaller than the second term for positive mass, damping and stiffness. This makes clear that $\nexists\ s \in \mathbb{C}_+$ such that $\mathcal{A}$ is singular. This shows that *the model without serial terms cannot be destabilized by time delays in the control.* Off course the solution and perhaps the growth rate of an already unstable solution are influenced. Note that a delay on the hydrodynamic damping, a kind of memory or retardation effect, is also incapable of destabilizing this model.

Gravity Adding a gravitational term to the equation would alter the behavior of the response and could influence stability. Consider an inverted pendulum on a hinge of length l and mass m and inertia J in a gravitational field g. A angular deviation from standing up straight φ results in a gravitational force component in the rotating direction of the pendulum. $F_g \sin\varphi = mg \sin\varphi$. The dynamics for this system are simply:

$$J\ddot{\varphi} = \tau_g = lmg \sin\varphi \approx lmg\varphi = k_g\varphi \tag{3-19}$$

The gravitational term works as an unstable spring that would appear in the k_{11} top left block of the stiffness matrix in Eq. (3-8) or in Eq. (3-2). This term is easily compensated *provided that the actuation force is sufficient* to allow for increase of k_P.

3-3-3 Fitting

Method Even though the dynamics are linear in the system matrices, the predicted response in Eq. (3-4) is not linear in the parameters that form the state matrix A_s according to Eq. (3-9). To fit the experimental data from the movie image processing (Section 3-2) numerical methods must be used. Results from realization theory (e.g. Appendix A.4 in [22]) can be used to find general linear system matrices but as this method does not allow constraints or a parameterized format for the output this method is unsuitable. The fitted model (denoted with a hat) plus an assumed exogenous error term denoted by ε_t are equal to the measured value (y_t in Eq. (3-20)).

$$\begin{aligned} \mathbf{y}_t &= \hat{\mathbf{y}}(t) + \varepsilon_t \\ &= e^{\hat{A}_s t}\hat{\mathbf{x}}_\mathbf{0} + \varepsilon_t \end{aligned} \tag{3-20}$$

It is assumed that this error term is independent and normally distributed with a mean of zero, as the data is already centered, and a (unknown) fixed standard deviation σ. The noise is thereby implicitly assumed identical for platform and boat.

$$\varepsilon_t = \mathbf{y}_t - \hat{\mathbf{y}}(t) \sim \mathcal{N}(0, \sigma I) \tag{3-21}$$

For a given parameter vector θ:

$$\theta = \begin{cases} \begin{bmatrix} k_{11} & k_{12} & k_{21} & k_{22} & c_{11} & c_{12} & c_{21} & c_{22} & \sigma \end{bmatrix}^T & \text{(Eq. (3 – 2))} \\ \begin{bmatrix} m_1 & m_2 & k_P & k_S & k_H & c_D & c_S & c_H & \sigma \end{bmatrix}^T & \text{(Eq. (3 – 8))} \end{cases} \tag{3-22}$$

The parameter estimation is performed by Maximum Likelihood as follows. (See e.g. Dekking [62] and Heij et al. [63]) The probability of a certain residual for one scalar state is found by:

$$\begin{aligned} \mathrm{P}[\varepsilon_t|\theta] &= \mathrm{P}[y_t - \hat{y}(t)|\theta] \\ &= \phi(y_t - \hat{y}(t)|\theta) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y_t - \hat{y}(t))^2/2\sigma^2} \end{aligned} \tag{3-23}$$

With $\mathrm{P}[Y|\theta]$ the probability of event Y conditional on θ, and $\phi()$ the Gaussian probability density function. Now the likelihood of the entire vector of residual observations is defined as:

$$\begin{aligned} \mathcal{L} &= \mathrm{P}[\mathbf{y} - \hat{\mathbf{y}}|\theta] \\ &= \prod_{t=0}^{T} \phi(\mathbf{y}_t - \hat{\mathbf{y}}(t)|\theta) \end{aligned} \tag{3-24}$$

This likelihood is transformed via the monotonic logarithmic transformation into the log-likelihood:

$$\begin{aligned} \log \mathcal{L} &= \sum_{t=0}^{T} \log \phi(\mathbf{y}_t - \hat{\mathbf{y}}(t)|\theta) \\ &= -\frac{N}{2}\log 2\pi - \frac{N}{2}\log \sigma^2 - \frac{1}{2}\sum_{t=0}^{T} (\mathbf{y}_t - \hat{\mathbf{y}}(t))^T (\mathbf{y}_t - \hat{\mathbf{y}}(t))/\sigma^2 \end{aligned} \tag{3-25}$$

With N the number of time observations. This $\log \mathcal{L}$ is maximized[2] via numerical optimization. As optimization algorithm a quasi newton method is used that numerically estimates the gradient via finite difference and uses an approximate inverse Hessian[3] (H^{-1}) via the BFGS algorithm[4], see for example Papalambros and Wilde [64]. The optimization step for a newton algorithm is:

$$d\theta = -\mathrm{H}(\theta)^{-1}\nabla\theta \tag{3-26}$$

The algorithm is iterated until convergence to the maximum likelihood parameters which are asymptotically normal distributed. Calculating this Hessian is convenient. The expected value of the Hessian (H) evaluated at the maximum likelihood estimate of parameters θ_{ml} is via the Fisher Information theory equal to the information matrix $\mathrm{E}[\mathrm{H}] = \mathcal{I}(\theta_{ml})$. And the inverse of the information matrix, and thus the expected inverse of the Hessian, is equal to the covariance matrix of the parameters ($\Sigma_\theta = \mathrm{H}^{-1}$). This covariance matrix contains information about the accurateness and significance of the estimation of the parameters. The standardized distance from zero of the ith parameter estimate $\hat{\theta}_i$ is given by:

$$z_i \triangleq \frac{\hat{\theta}_i}{\sqrt{\Sigma_\theta[i,i]}} \tag{3-27}$$

Fitting non physical model The model from Eq. (3-2) is estimated. The non convexity of the optimization results in slow convergence and sensitivity to the initial guess of the optimization.

$$A_s = \begin{bmatrix} -0.90 & 0.24 & -21.21 & -3.92 \\ -1.71 & 0.91 & 1.07 & -28.87 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-28}$$

The fit in Figure 3-5 is visually quite successful, the parameters and their z-value in parenthesis (standarized parameter distance from zero by the positive definite Hessian/covariance matrix) are given in Table 3-1. The diagonal elements in the stiffness matrix are as expected from the calculations in the stability section (Section 3-3-2) of similar magnitude hence the controlled platform's and the hydrostatic supported boat's eigenfrequencies are close but they are not matching. This is also visible in the "slow beating" in the oscillation.

The residuals of the platform and boat fit are analyzed for normality as non normality is a sign of misspecification. A Gaussian fit gives for the platform: $\sigma_1 = 0.642$, $\mu_1 = 0.016$ and for the boat: $\sigma_2 = 0.406$, $\mu_2 = 0.042$. The kurtosis (K), skewness (S), Jarque-Bera (JB)[5] and outlier corrected ($JB_{(-i)}$), without the i biggest residuals on both sides, for both fits are:

$$\begin{aligned} K_1 &= 2.51, \quad S_1 = -0.23, \quad JB_1 = 192.16, \quad JB_{1(-3)} = 3.31 \\ K_2 &= 2.46, \quad S_2 = -0.14, \quad JB_2 = 2.08, \quad\;\; JB_{2(-1)} = 0.80 \end{aligned}$$

[2]Note that maximizing this objective is very similar to minimizing the sum of squared residuals (LS): $SSR = \sum_1^N (\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}})$ via $\min_{\mathbf{x_0},C,K} SSR$ With as upside additional information about the error in estimation and the residual distribution.

[3]Matrix of second derivatives of likelihood with respect to the parameters.

[4]BFGS stands for Broyden-Fletcher-Goldfarb-Shanno

[5]The Jarque-Bera normality test is defined as: $JB \triangleq \frac{\text{smpl.size}}{6}\left(S^2 + \frac{(K-3)^2}{4}\right) \sim \mathrm{H}_0 : \chi^2_{(2)}$

$\dot{\phi}_{p0}$	9.38	(18.52)	$\dot{\phi}_{b0}$	5.20	(5.43)
ϕ_{p0}	-0.72	(-5.77)	ϕ_{b0}	-2.88	(-17.06)
k_{11}	-21.21	(-40.96)	k_{12}	-3.92	(-4.10)
k_{21}	1.07	(1.29)	k_{22}	-28.87	(-39.29)
c_{11}	-0.90	(-3.29)	c_{12}	0.24	(2.65)
c_{21}	-1.71	(-3.81)	c_{22}	0.91	(3.35)
σ^2	0.78	(23.17)			

Table 3-1: Non physics based fit of Eq. (3-2) with $\log \mathcal{L} = -352.61$. z-values of the parameters are given in parentheses.

The JB test for normality has a $\chi^2_{(2)}$ distribution under the null hypothesis of normality. The 5% critical value is approxmately 5.99 so normality is not rejected if the extreme residuals are omitted.

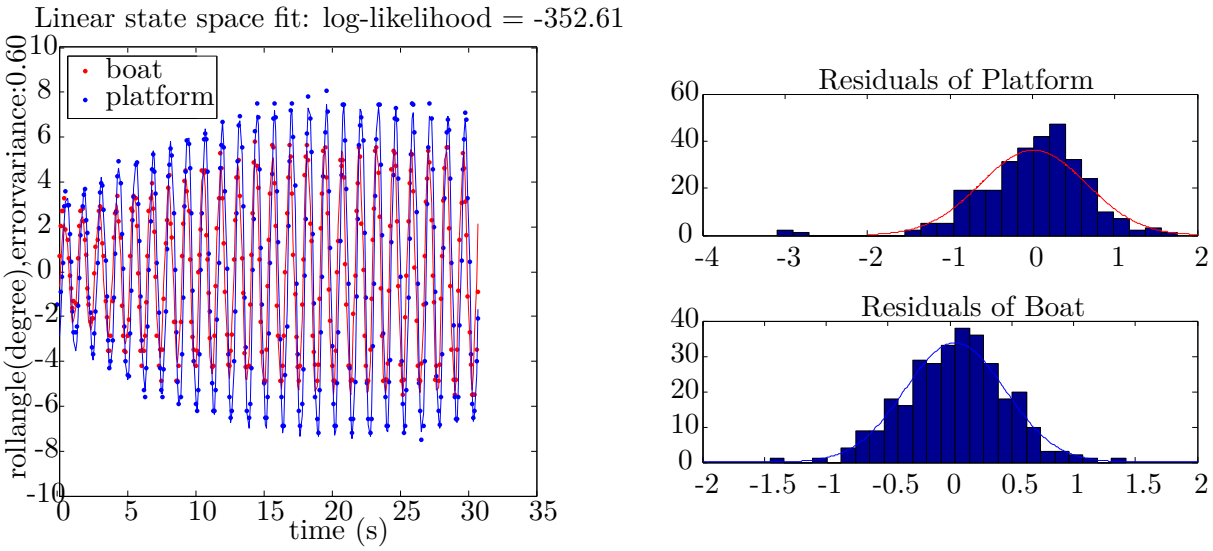

Figure 3-5: Fit for the full dataset, non physical model. Direct estimation of the stiffness matrix (K) and damping matrix (C) components. The dots are the measured data points and the full line is the estimated model.

The z-values are estimated with the variance matrix given by the numerical Hessian. Almost all parameters are estimated with fair significance ($|z| > 2$). What is interesting is that the positive damping in c_{22} is feeding the oscillation with energy. As this is the damping on the movement of the ship it is not logical that this is positive. But when the model is used for out of sample prediction (Figure 3-6) a illogical burst of oscillation is visible and the estimated model is assumed either lacking an important effect, is over fitted or is non linear.

Fitting physics based parametric model Now the model from Eq. (3-8) with $x_R = 0$ is fitted to the data with a new maximum likelihood estimation. This results in the optimum given in Table 3-2. And in the system matrix given in Eq. (3-29) where it is seen that the stiffnesses have very similar values as the ones in Eq. (3-28) only differing about 6%. This would be a

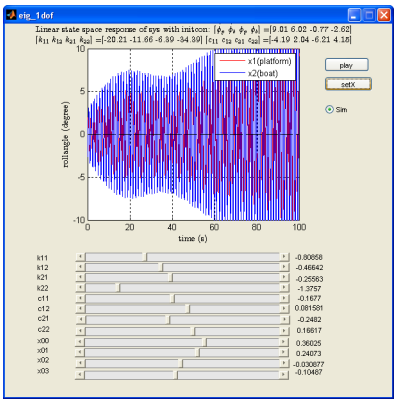

Figure 3-6: Prediction non physical fit 100 seconds. User interface tool designed to create feasible estimates for the initialization of the optimization algorithm, look at the mode shapes and visualize the simulation.

logical result if the fitting was unconstrained, as the data is the same, the parametric model is in essence is a one-on-one transformation. But the difference is that the sign of the mass, spring and dashpot parameters is fixed to be positive. The optimization finds an optimum without derivative damping c_D which is probably not true.

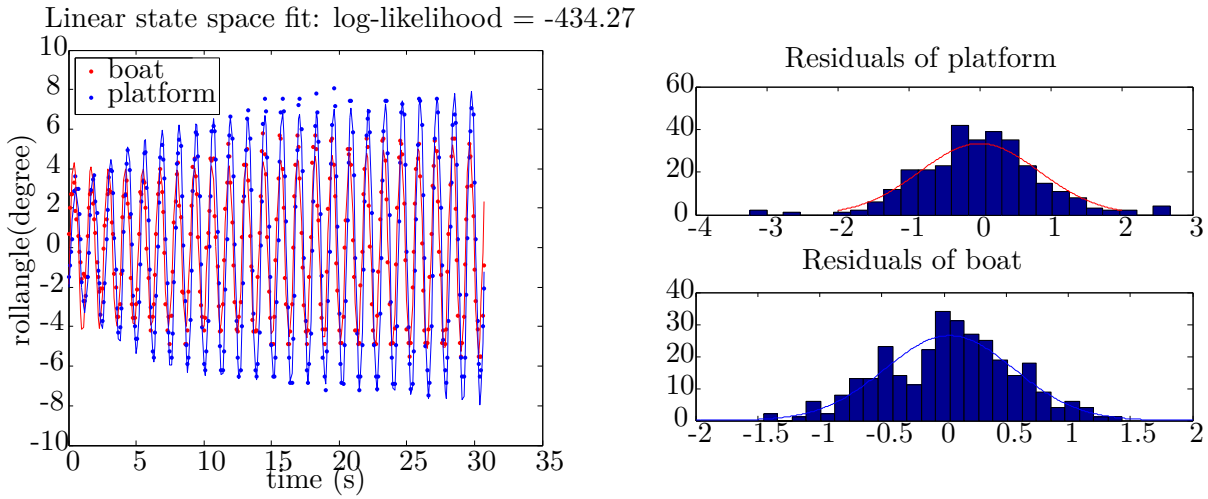

Figure 3-7: Physics based parametric fit of linear mass spring dashpot system from Figure 3-4. $m_1 = 20kg$ and the full set $t = 0..T$ is used. The dots are the measured data points and the full line is the estimated model.

$$A_s = \begin{bmatrix} -0.11 & 0.11 & -24.87 & 0.00 \\ 0.02 & -0.27 & 3.79 & -26.49 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-29}$$

Residual statistics:

$$K_1 = 2.71, \quad S_1 = 0.19, \quad JB_1 = 29.31, \quad JB_{1(-4)} = 2.75$$
$$K_2 = 2.39, \quad S_2 = -0.10, \quad JB_2 = 1.76$$

The fit is reasonable for the second half and the JB values are not rejecting normality when the bottom and top 4 extreme residuals are removed for the platform: $JB_1 = 29.31$, $JB_{1(-4)} = 2.75 < 5.99$ and for the boat directly: $JB_2 = 1.76 < 5.99$. With all residuals included, especially the large ones at the beginning, make the JB test reject normality.

The build up of the oscillation is not properly fit as the first 15 seconds show large residuals. In fact, the fit contracts in the first 8 seconds whereas the data shows diverging maxima from the beginning. Out of sample prediction shows unbounded growth in both observed angles. Considering that nonlinearities are more dominant with larger deviations from the upright position, it is possible that the second half of the oscillation, the kind of limiting behavior, is due to the nonlinearities. But as the fitting does not include this effect, and the absolute values of the angles are greater in the second half, the optimization focuses on this part and the quality of the fit in the first half is lost.

$\dot{\phi}_{p0}$	19.41	(104.66)	$\dot{\phi}_{b0}$	7.86	(6.87)
ϕ_{p0}	-1.87	(-18.41)	ϕ_{b0}	-2.00	(-10.09)
m_1	20.00	(fixed)	m_2	131.12	(893.89)
k_P	497.30	(585.76)	k_S	0.00	(0.00)
k_H	3473.04	(1059.02)	c_D	0.00	(0.00)
c_S	2.13	(8.70)	c_H	33.80	(171.85)
σ	0.98	(25.54)			

Table 3-2: Physical parameterized fit of figure Figure 3-7 with $\log \mathcal{L} = -434.27$. This is a local optimum as the initial guess needed to be quite close to arrive at this setting.

First half fitting physics based parametric model As the linear model is not capable of explaining the limited angles, the assumption of constant parameters is dropped and only the first 15 seconds (oscillation build up) are fitted with the model from Eq. (3-9) and plotted in Figure 3-8.

The system matrix is found to be:

$$A_s = \begin{bmatrix} -3.32 & 3.15 & -37.78 & 0.00 \\ 0.91 & -1.35 & 10.36 & -28.35 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-30}$$

The fit is visually good, and the eigenvalues of As: $\lambda_1 = -2.3847 \pm 6.1673i$ and $\lambda_2 = 0.0497 \pm 4.9492i$ show a stable (negative real) and an unstable solution. And both eigenfrequencies are of the same order of magnitude shown by the complex part.

Residual statistics:

$$K_1 = 2.32, \quad S_1 = -0.24, \quad JB_1 = 2.71,$$
$$K_2 = 2.49, \quad S_2 = 0.31, \quad JB_2 = 25.49, \quad JB_{2(-4)} = 3.72$$

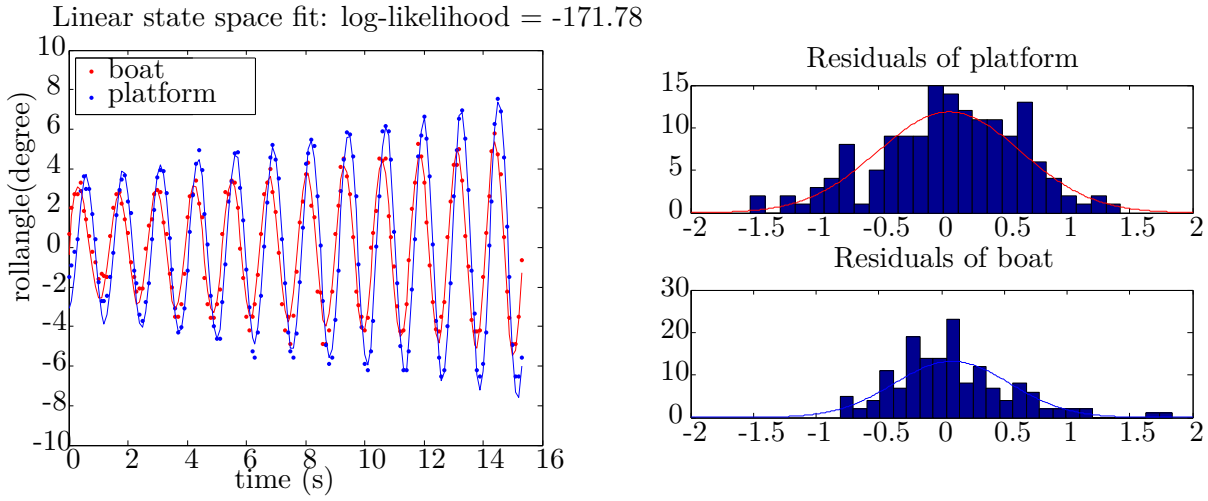

Figure 3-8: Fit for the first 15 seconds using the parametric physical model. $m_1 = 20kg$ and the first half of the dataset is used so $t = 0..T/2$. The dots are the measured data points and the full line is the estimated model.

$\dot{\phi}_{p0}$	23.25	(4.77)	$\dot{\phi}_{b0}$	-1.64	(-0.76)
ϕ_{p0}	-0.37	(-0.71)	ϕ_{b0}	-3.09	(-10.01)
m_1	20.00	(fixed)	m_2	72.97	(275.16)
k_P	755.63	(54.57)	k_S	0.00	(0.01)
k_H	2068.72	(164.39)	c_D	3.37	(1.40)
c_S	63.00	(43.21)	c_H	35.39	(18.46)
σ	0.74	(16.85)			

Table 3-3: Physical parameterized fit of first 15 seconds with $\log \mathcal{L} = -171.78$.

Second half fitting physics based parametric model Now the second half of the dataset (the limit cycling) is fitted with the model from Eq. (3-9). The optimization is initiated with the found solution from previous section where the first 15 seconds were fitted. The results are shown in Figure 3-9.

$$A_s = \begin{bmatrix} -3.64 & 3.56 & -41.02 & 0.00 \\ 0.92 & -1.47 & 10.38 & -28.10 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-31}$$

The eigenvalues of As are $\lambda_1 = -2.5532 \pm 6.3470i$ and $\lambda_2 = -0.0020 \pm 4.9375i$ show two stable (negative real) solutions. The second solution persists long due to the small absolute value of the real part. The estimation of the hydrostatic stiffness (k_H) and control stiffness (k_P) increase a bit (both about 6%) and the control damping c_D is still not significantly diffrent from zero. Now a serial spring is found with a z value of four and a half, but in comparison with the other estimations the power of this estimation is still moderate. The estimation of the serial damping increases a bit more, about 12%, but the hydrostatic damping estimate increases a lot: 28%.

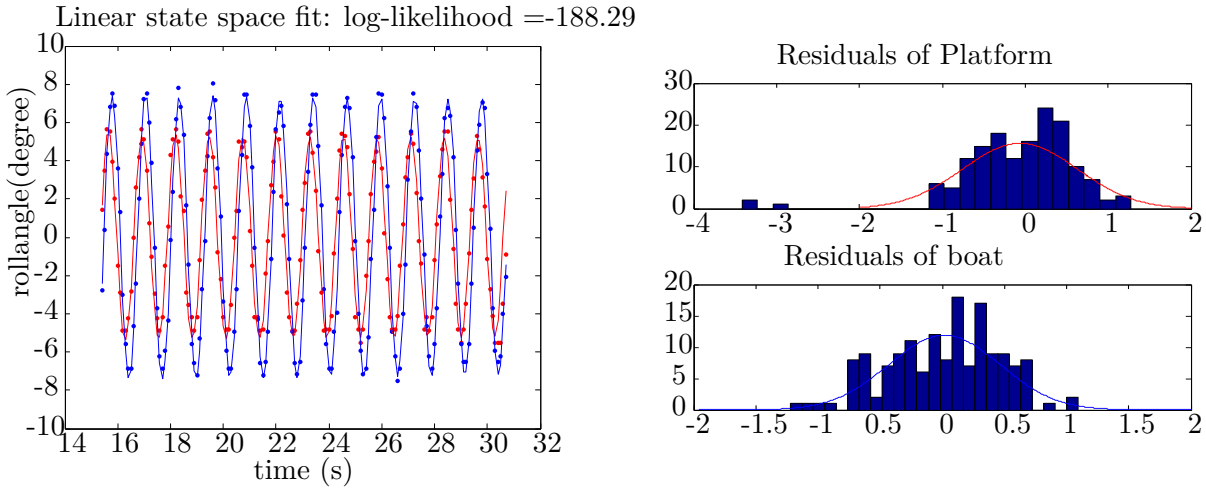

Figure 3-9: Fit for the second 15 seconds using the parametric physical model. $m_1 = 20kg$ and the second half of the dataset is used so $t = T/2..T$. The dots are the measured data points and the full line is the estimated model.

$\dot{\phi}_{p0}$	26.92	(4.48)	$\dot{\phi}_{b0}$	33.91	(13.47)
ϕ_{p0}	1.68	(2.76)	ϕ_{b0}	−2.43	(−7.22)
m_1	20.00	(fixed)	m_2	79.03	(113.52)
k_P	798.18	(70.18)	k_S	22.19	(4.54)
k_H	2198.47	(152.53)	c_D	1.53	(0.87)
c_S	71.19	(70.12)	c_H	45.36	(27.07)
σ	0.82	(17.31)			

Table 3-4: Physical parameterized fit of second 15 seconds with $\log \mathcal{L} = -188.29$.

Residual statistics:

$$K_1 = 2.13, \quad S_1 = 0.01, \quad JB_1 = 406.00, \quad JB_{1(-3)} = 3.61$$
$$K_2 = 2.06, \quad S_2 = -0.24, \quad JB_2 = 2.92,$$

The residuals show a strong indication for an underlying normal distribution. Only the last three big residuals at $t = 30$ have to be ignored to take care of the excess kurtosis. This is a sign of correct model specification.

Parameter sweeps, eigenvalues A kind of sensitivity analysis in order to get a grip on the severity of the nonlinearities is made, see Figure 3-10 The found solution for the $20kg$ parametric model for the first half T is used, shown for the x axis value 100%, and all eight parameters are varied. A positive real part gives an unstable response of the system. The resulting curves show that the hydodynamic damping and the derivative control have a monontonic stabilizing effect on increase, whilst the other parameters (when varied solitary) have a range of instability bound for most parameters by one half to to times the value. k_S was estimated zero hence the horizontal line.

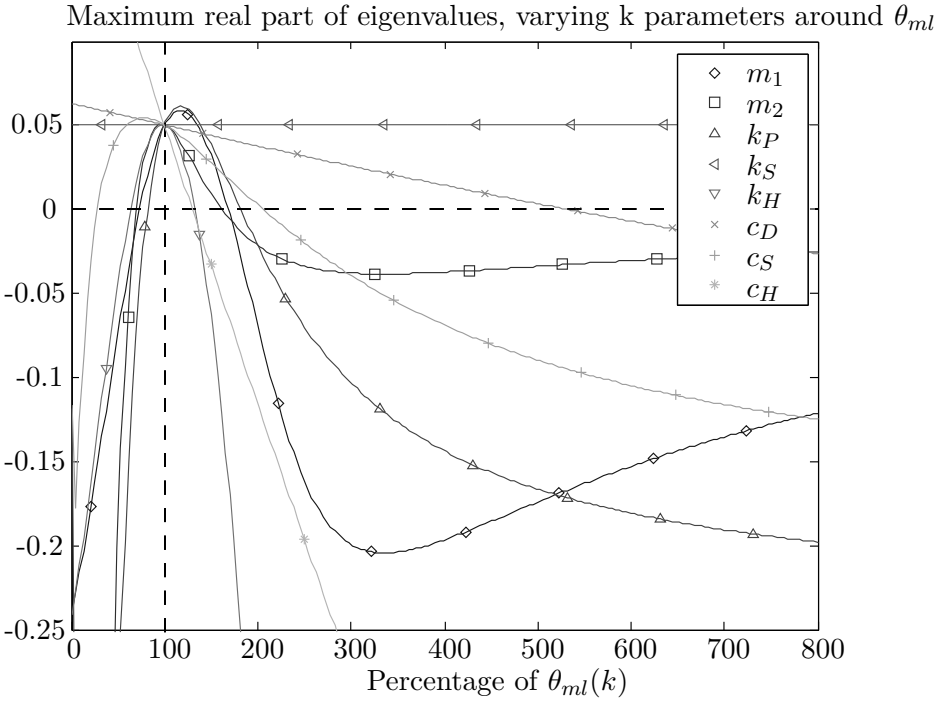

Figure 3-10: Influence of the 8 parameter values on stability (positiveness of the eigenvalues) around the solution of the fit for the first 15 seconds. The parameters of the maximum likelihood estimation are the set θ_{ml}. In every line in the plot, one of these estimates is varied rationally to its value. The maximum real part at θ_{ml} (100%) is about 0.05. It is visible that there is a unstable regeon that can be turned stable by various ways: increasing or decreasing the serial damping, increasing the hydrostatic damping, changing the masses. Note that changing the masses would also change the hydrostatics in a real world application.

3-4 Conclusion

The build up and limit cycling behavior visible in the Ampelmann scale model testing movie is reproducible by a split linear model for the two regimes. The fit matches the measurements with high accuracy and the parameters have logical values. The coupling between the two bodies via the serial damping term c_S is a key ingredient to create an unstable pole in this model. The cause for this term could be hydraulic damping present in the system or a derivative error that is not incorporated as $\dot{\epsilon} = \dot{x}_R - \dot{x}_1$ but as $\dot{\epsilon} = \dot{x}_2 - \dot{x}_1$. This conclusion is not supported very strong as multiple other nonlinearities could be present (such as nonlinear actuation forces for example) and this conclusion is based on a estimated linear model.

However it is shown that there always is a configuration of parameters possible such that the eigenfrequencies collide and the double, marginally stable (mainly complex), pole destabilizes the system. This situation could be reached, or even searched for by the nonlinearities. Therefore it is important to protect the system against this potential danger in the actuation strategy.

Chapter 4

3D simulation

4-1 Mechanism description

In this chapter the working principle, dynamics and control of a new ship motion compensation mechanism is discussed. The mechanism is based on three planar linkages, so called "Sarrus" mechanisms, that constrain the movements in the ship's horizontal plane. The platform allows only heaving (vertical) translations and roll and pitch rotations. This decision is made for several reasons. The ship's accelerations in these three directions are much larger than the planar movements[1]. The planar movements are more easy to counteract via other methods such as Dynamic Positioning (DP) and anchoring. The construction can be made more rigid by the planar constraining mechanism and finally less Degrees of Freedom (DoFs) requires less actuators. This provides a cost reduction and improves the robustness advantage to more complex systems.

The mechanism allows a force-reach trade off via the linkage as the actuator position on the top linkage can be altered. Shifting the actuator connection towards(/away from) the platform increases(/decreases) the available force and decreases(/increases) the achievable heave distance and roll and pitch angles. See the raw sketch in Figure 4-1.

4-2 Mechanism kinematics

The approach to find the forward kinematics, that is the relations to find the platform pose given the hydraulic leg lengths, is divided in the following steps:

- Develop necessary (mathematical) tools: coordinate systems, rotation and translation definitions. Section 4-2-1

[1] Table 2 in [37]

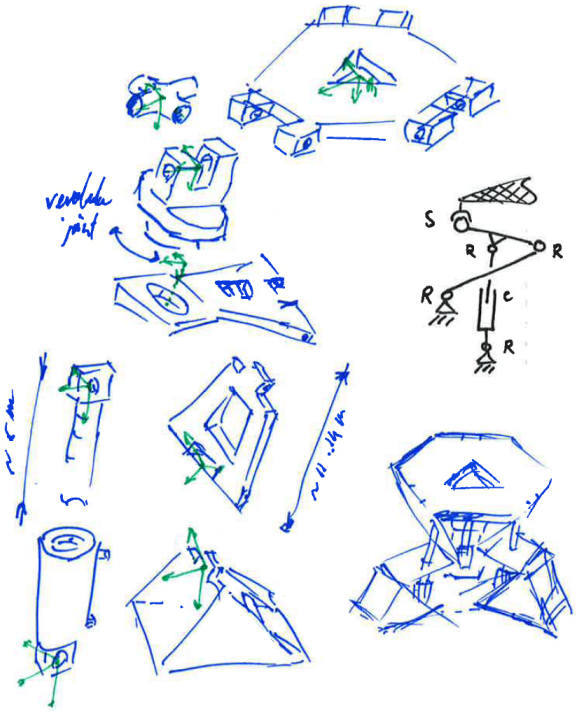

Figure 4-1: Sketch of the 3 DoF parallel manipulator platform with 3 planar linkage systems with a leveraged hydraulic transmission. The three planar linkages are placed $120°$ apart constraining the system to move only in heave(z), roll(ϕ) and pitch(θ) directions. The linkage consists of two arms attached with a revolute joint (R) to each other and to the ground. The top linkage is connected to the platform by a rotating cardan joint permitting three rotations (acting as a spherical (S) joint) and with a revolute joint to the hydraulic cylinder (C). The hydraulic cylinder is in turn attached to the ground by a revolute joint. Note that the cylinder is only loaded axially due to the linkage.

- Inverse kinematics (κ^{-1}): given the platform pose vector ($\bar{x}_\beta$ composed of 1 translation and two rotations, formally defined in Eq. (4-2)), what are the leg lengths $\ell_{1..3}$, and leg directions $\bar{l}_{1..3}$. Section 4-2-2
- Time rate of change of inverse kinematics: finds a relation between $\dot{\bar{x}}_\beta$ and $\dot{\bar{\ell}}$ called the Jacobian. Section 4-2-3
- Forward kinematics: Newton-Rhapson iterations with Jacobian find the platform pose from given leg lengths. Section 4-2-4

4-2-1 Mathematical tools

Coordinates The platform pose is determined relative to the ship by a translation and a rotation. See Figure 4-3. The boat's origin is placed on deck on the symmetry plane, not necessarily above the CoG with the attached coordinate system called $\mathcal{B}$. The platform's origin is placed in the loaded platform's CoG and called $\mathcal{P}$. The vector pointing from the origin of the boat frame to the origin of the platform's frame is denoted by $\bar{c}$ and the rotation from

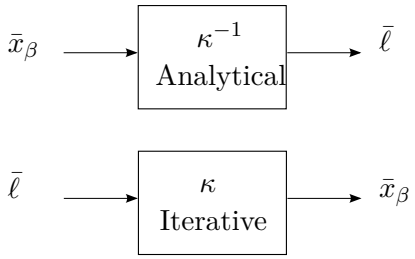

Figure 4-2: Inverse and forward kinematical relations. The inverse kinematics find the joint coordinates (leg lengths) from a given end effector translational and rotational state is a unique analytical relation. The forward kinematics is not guaranteed unique for parallel manipulators, and the solution must be found by iterations.

the $\mathcal{B}$ to $\mathcal{P}$ by $R_{\mathcal{B}\to\mathcal{P}}$. Coordinate free vectors can be described in a reference frame of choice. This is denoted by a superscript, e.g. $\bar{c}^{\mathcal{B}}$ or $\bar{c}^{\mathcal{P}}$.

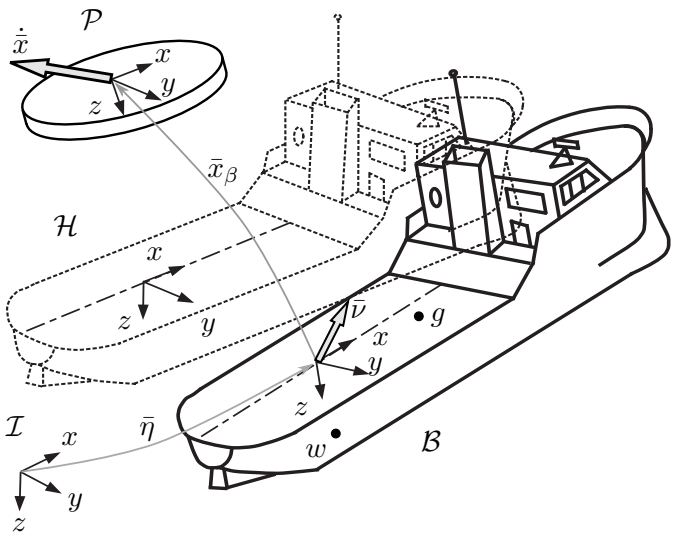

Figure 4-3: The main four reference frames. The platform $\mathcal{P}$, the ship $\mathcal{B}$, the inertial $\mathcal{I}$ and the hydrodynamic equilibrium frame $\mathcal{H}$. The pose of the boat in the inertial frame is found via $\bar{\eta}$ and the relative pose of the platform via $\bar{x}_\beta$. The body fixed velocities for boat and platform are $\bar{\nu}$ and $\dot{\bar{x}}$ respectively.

Rotation We use the compact notation $c_\phi \triangleq \cos(\phi)$, $s_\theta \triangleq \sin(\theta)$ for the sine and cosine functions. A coordinate rotation around a single axis finds the coordinates of a vector (z) in a rotated coordinate system (z') that has one axis in common with the originating system (e.g the X axis $z' = R_X z$). The rotation from the $\mathcal{B}$ to the $\mathcal{P}$ frame is found by thee consecutive coordinate rotations shown in Figure 4-4. Combined in the roll-pitch-yaw Euler angle list $\bar{\beta}$

they form the rotation matrix:

$$R_{\mathcal{B}\to\mathcal{P}}(\bar{\beta}) = R_X(\varphi)R_Y(\theta)R_Z(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\varphi & s_\varphi \\ 0 & -s_\varphi & c_\varphi \end{bmatrix} \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\varphi s_\theta c_\psi - c_\varphi s_\psi & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & s_\varphi c_\theta \\ c_\varphi s_\theta c_\psi + s_\varphi s_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & c_\varphi c_\theta \end{bmatrix} = R^T_{\mathcal{P}\to\mathcal{B}}(\bar{\beta}) \tag{4-1}$$

The translation vector and Euler angle list can be combined into one vector defining the pose of the platform as:

$$\bar{x}_\beta = \begin{bmatrix} \bar{c}^{\mathcal{B}} & \bar{\beta} \end{bmatrix}^T \tag{4-2}$$

The Euler angle rates[2], see Figure 4-4, are found from the instantaneous rotation rate vector of the platform expressed in the boat frame $\bar{\omega}^{\mathcal{B}}$ as (Eq. 295 Diebel [65] and pp. 3-4 Schwab and Meijaard (2006) [66]). As the platforms frame $\mathcal{P}$ is expressed relative to $\mathcal{B}$, the boat is the platform's frame "ground". Hence we use the "ground"-to-euler E and not the "body"-to-euler E' as used in the ship attitude transformation Eq. (2-14).

$$\dot{\bar{\beta}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{c_\theta} \begin{bmatrix} c_\psi & s_\psi & 0 \\ -c_\theta s_\psi & c_\theta c_\psi & 0 \\ c_\psi s_\theta & s_\psi s_\theta & c_\theta \end{bmatrix} \bar{\omega}^{\mathcal{B}} = [E(\bar{\beta})]^{-1}\bar{\omega}^{\mathcal{B}} \tag{4-3}$$

We can define a generalized velocity vector as:

$$\dot{\bar{x}} = \begin{bmatrix} \dot{\bar{c}}^{\mathcal{B}} & \bar{\omega}^{\mathcal{B}} \end{bmatrix}^T \tag{4-4}$$

Note that $\frac{d}{dt}(x_\beta) \neq \dot{x}$. We can transform from the time derivative of Eq. (4-2) to the generalized velocity vector by:

$$\dot{\bar{x}}_\beta = \begin{bmatrix} \dot{\bar{c}}^{\mathcal{B}} \\ \dot{\bar{\beta}} \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & [E(\bar{\beta})]^{-1} \end{bmatrix} \begin{bmatrix} \dot{\bar{c}}^{\mathcal{B}} \\ \bar{\omega}^{\mathcal{B}} \end{bmatrix} \triangleq \mathrm{J}_{\bar{x}_\beta,\bar{x}}\dot{\bar{x}} \tag{4-5}$$

4-2-2 Inverse kinematics

In this section the the inverse kinematical relations of the mechanism from Figure 4-1, that find the leg lengths from the pose of the platform, are derived. The mechanism is described by

[2] A few useful relations are given in Diebel (2006) [65] Section 8.2 (p.24).
The linearized variant of Eq. (4-1) is: $L\{R_{\mathcal{B}\to\mathcal{P}}\} = \begin{bmatrix} 1 & \psi & -\theta \\ -\psi & 1 & \phi \\ \theta & -\phi & 1 \end{bmatrix}$
The relation between the rates, their transformations and conjugate transformations:
$\omega^{\mathcal{P}} = R_{123}(\bar{\beta})\omega^{\mathcal{B}}$, $\omega^{\mathcal{B}} = E_{123}(\bar{\beta})\dot{\bar{\beta}}$ and $\omega^{\mathcal{P}} = E'_{123}(\bar{\beta})\dot{\bar{\beta}}$
Where the inverse of the conjugate transformation is: $[E'(\bar{\beta})]^{-1} = \frac{1}{c_\theta}\begin{bmatrix} c_\theta & s_\phi s_\theta & c_\phi s_\theta \\ 0 & c_\phi c_\theta & -s_\phi c_\theta \\ 0 & s_\phi & c_\phi \end{bmatrix}$

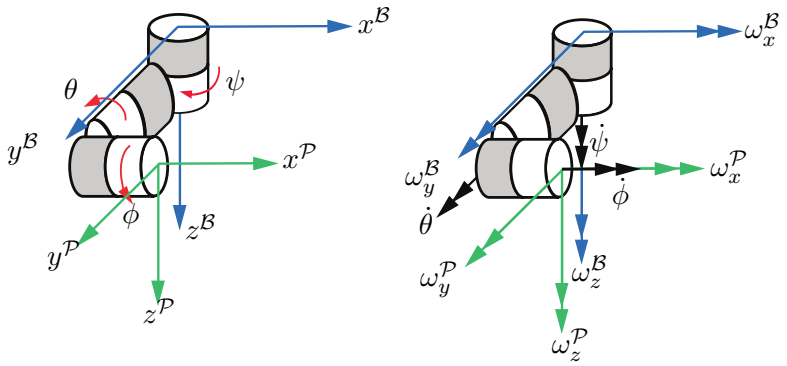

Figure 4-4: The North-East-Down, xyz-Euler angles and rotation rates drawn as revolute joints or cans in series as in [66]. The grey part of the joint is fixed and the white part can rotate. The chain of rotations defines the rotational pose of the secondary base relative to the first.

a double chain of vectors shown in Figure 4-5. The methods used to derive at the kinematical relations are analogous (not similar) to the methods used by Koekebakker (2001) [28] in his PhD thesis considering model based control of a flight simulator, Li and Salcudean (1997) [67] in their publication considering a hanging 6 legged Stewart platform and Hsu and Fong (2001) [68] with their paper on computed force feedback of a 6 legged hydraulic platform.

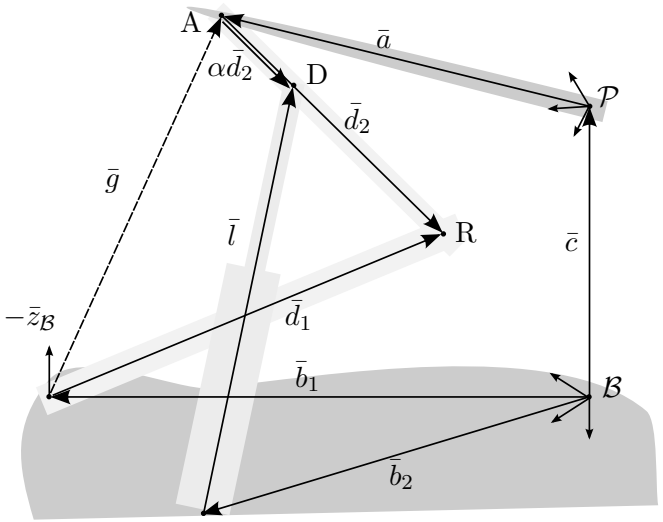

Figure 4-5: Vectorial representation of the planar linkage system of one leg. The mechanism is shown in Figure 4-1.

Hydraulic leg vectors The hydraulic cylinder is represented by the vector $\bar{l}_i$ where $i = 1..3$. For the sake of readability and because all three legs are identical, the subscript is dropped. The vector from the platform axis base to joint A is constant and known in $\mathcal{P}$ and called $\bar{a}$. The vectors defining the mounting points on the deck $\bar{b}_1$ and $\bar{b}_2$ are constant and known in $\mathcal{B}$.

The hydraulic leg is mounted on fractal distance $\alpha = [0..1]$ along linkage $\bar{d}_2$. The kinematic loop with the leg included in the base frame leads to:

$$\begin{aligned} &\bar{b}_2 + \bar{l} - \alpha\bar{d}_2 - \bar{a}^{\mathcal{B}} - \bar{c} = 0 \\ &\bar{b}_2 + \bar{l} - \alpha\bar{d}_2 - R^T_{\mathcal{B}\to\mathcal{P}}\bar{a}^{\mathcal{P}} - \bar{c} = 0 \\ &\bar{l} = \alpha\bar{d}_2 + R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}} + \bar{c} - \bar{b}_2 \end{aligned} \tag{4-6}$$

The length[3] and direction of the hydraulic leg are found and denoted by:

$$\begin{aligned} \ell &= |\bar{l}| = |\alpha\bar{d}_2 + R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}} + \bar{c} - \bar{b}_2| \\ \bar{l}_n &= \frac{\bar{l}}{\ell} \end{aligned} \tag{4-7}$$

Linkage vectors To find ℓ and $\bar{l}_n$ in Eq. (4-7) from pose vector $\bar{x}_\beta$ defined by Eq. (4-2), $\bar{d}_2$ needs to be found. We arrive at an expression for this vector using the following know facts and values:

$\bar{b}_1^{\mathcal{B}},\ \bar{b}_2^{\mathcal{B}},\ \bar{a}^{\mathcal{P}}$	(known basis and platform connection points)
$R_{\mathcal{B}\to\mathcal{P}},\ \bar{c}^{\mathcal{B}}$	(known transformation from boat to platform axis)
$\alpha,\ \lvert\bar{d}_1\rvert,\ \lvert\bar{d}_2\rvert$	(known linkage dimensions)
$\{\bar{a}, \bar{b}_1, \bar{b}_2, \bar{c}, \bar{d}_1, \bar{d}_2, \bar{l}, \bar{g}\} \in \mathbb{R}^2$	(planar mechanism)
$\angle\text{-}\bar{z}_{\mathcal{B}}\bar{g} > 0,\ \angle\bar{g}\bar{l} > 0,\ \angle\text{-}\bar{z}_{\mathcal{B}}\bar{d}_1 \in (0, \frac{\pi}{2})$	(no singular pose)
$\bar{b}_1 \perp \bar{c} \parallel \bar{z}_{\mathcal{B}}$	(height in z-direction and forms right angle with deck)

Auxiliary vector $\bar{g}$ together with the angles from the barge's z-axis ($\bar{z}_{\mathcal{B}}$) and this vector, and from $\bar{g}$ to the linkage are found by:

$$\begin{aligned} &\bar{g} = \bar{c} + R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}} - \bar{b}_1, \text{ unit direction: } \bar{g}_n = \frac{\bar{g}}{|\bar{g}|} \\ &\angle\text{-}\bar{z}_{\mathcal{B}}\bar{g} = \cos^{-1}(\bar{g}_n^T\bar{e}_z), \text{ with: } \bar{e}_z = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T \\ &\angle\bar{g}\bar{d}_1 = \cos^{-1}\left(\frac{|\bar{g}|^2 + |\bar{d}_1|^2 - |\bar{d}_2|^2}{2|\bar{g}||\bar{d}_1|}\right) \end{aligned} \tag{4-8}$$

The total angle from the z-axis to the linkage now becomes:

$$\angle\text{-}\bar{z}_{\mathcal{B}}\bar{d}_1 = \angle\text{-}\bar{z}_{\mathcal{B}}\bar{g} + \angle\bar{g}\bar{d}_1 \tag{4-9}$$

Now the vectors defining the linkage are given by the lengths and the found angle, and using the second kinematic loop:

$$\begin{aligned} \bar{d}_1 &= |\bar{d}_1|\left(\text{-}\bar{z}_{\mathcal{B}}\cos(\angle\text{-}\bar{z}_{\mathcal{B}}\bar{d}_1) + \frac{-\bar{b}_1}{|\bar{b}_1|}\sin(\angle\text{-}\bar{z}_{\mathcal{B}}\bar{d}_1)\right) \\ \bar{d}_2 &= \bar{b}_1 + \bar{d}_1 - \bar{c} - R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}} \end{aligned} \tag{4-10}$$

Every part in Eq. (4-7) for the hydraulic legs can now be found from the dimensions of the mechanism and the pose given in $\bar{x}_\beta$.

[3] the length is found using the definition of the norm of a vector $\bar{v}$: $|\bar{v}|^2 = \bar{v}^T\bar{v}$

4-2-3 Time derivatives of inverse kinematics

In this section a relation between the velocity of the platform and the rate of change of the joint coordinates, the leg lengths, is found. We use the generalized velocity vector from Eq. (4-4). The velocity of joint A (V_A) in the $\mathcal{B}$ frame, Figure 4-5, can be found from the generalized velocity by[4]:

$$\begin{aligned} V_A &= \dot{\bar{c}} + \bar{\omega} \times R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}} \\ V_A &= \dot{\bar{c}} + [\bar{\omega}]_\times R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}} \end{aligned} \tag{4-11}$$

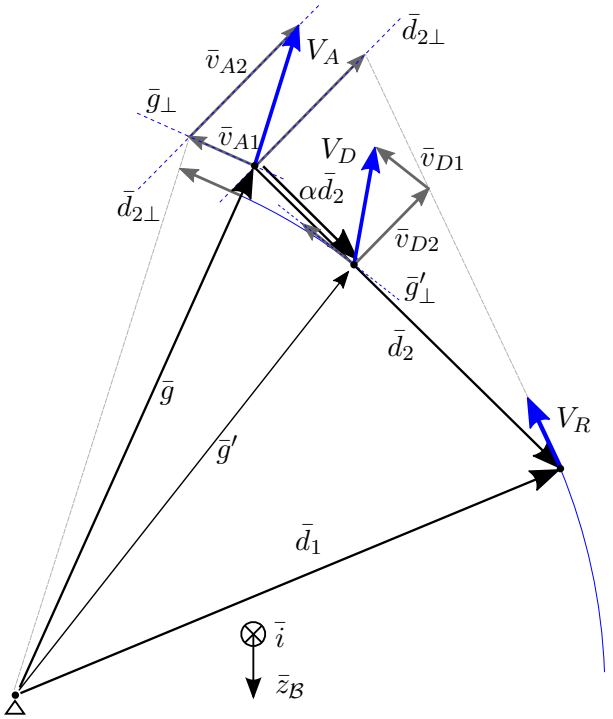

Figure 4-6: Velocity of joints in planar linkage system. The linkage is connected to the platform in A and to the hydraulics in D. The vectors $\bar{d}_1$ and $\bar{d}_2$ form the parallel mechanism of the linkage, and the auxiliary vectors $\bar{g}$ and $\bar{g}'$ are used in calculation and represent no parts. The main strategy of the velocity transformation is to decompose velocity vector V_A in two independent directions that represent the two rotations around the revolute joints at the floor and in R.

Decomposition of V_A The velocity in the revolute joint between the linkage and the hydraulic cylinder, denoted by V_D in Figure 4-6, is found by noticing that the revolute joint in R, connecting the two arms of the linkage, can only move in a circular motion. Therefore the velocity in this point V_R must be tangential to $\bar{d}_1$. The hinge on the boat deck can also only rotate around its own axis. The velocity of point A (V_A) can therefore be described by two rotation rates around the two hinges (denoted by $\dot{\bar{\gamma}}_{1,2}$) pointing out of plane, and the arm of

[4] $[\bar{v}]_\times$ is the skew symmetric cross product matrix such that $\begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \bar{r} = \bar{v} \times \bar{r}$

the two rotations:

$$\begin{aligned} V_A &= \dot{\bar{\gamma}}_1 \times \bar{d}_1 + \dot{\bar{\gamma}}_2 \times (\bar{d}_1 - \bar{d}_2) \\ &= \dot{\bar{\gamma}}_1 \times \bar{d}_1 + \dot{\bar{\gamma}}_2 \times \bar{g} \end{aligned} \tag{4-12}$$

V_A can therefore be decomposed into two components, both in plane, and orthogonal to the rotation arms.

$$\begin{aligned} V_A &= \bar{v}_{A1} + \bar{v}_{A2} \\ &= v_{A1}\bar{g}_{n\perp} + v_{A2}\bar{d}_{2n\perp} \end{aligned} \tag{4-13}$$

Where the magnitudes are denoted without bar as they are scalar valued, and the subscript n is used for normalization. The $\bar{g}$ and $\bar{d}_2$ vectors are found respectively in Eq. (4-8) and Eq. (4-10). The subscript $\perp$ denotes "perpendicular to" whose direction is fixed defining the out of plane direction by:

$$\bar{i} \triangleq \bar{z}_{\mathcal{B}} \times \bar{b}_{1n} \tag{4-14}$$

or in words as the outer product of the unit Z direction vector of the boat frame (a NED frame) and the unit vector in the direction pointing to the deck mounted hinge of the linkage. Note that i is normalized by construction. A perpendicular vector is found by taking the outer product of the vector with $\bar{i}$. ($\bar{g}_\perp = \bar{g} \times \bar{i}$) This allows finding the magnitude of the decomposition of V_A via inversion of:

$$V_A = \begin{bmatrix} \bar{g}_{n\perp} & \bar{d}_{1n\perp} & \bar{i} \end{bmatrix} \begin{bmatrix} v_{A1} \\ v_{A2} \\ 0 \end{bmatrix} \tag{4-15}$$

Using the subscript x,y and z to denote the $\mathcal{B}$ based Carthesian component of the vectors in the decomposition matrix, the inverse of Eq. (4-15) is found as[5][6]:

$$\Delta = \frac{1}{\underbrace{i_x^2 + i_y^2 + i_z^2}_{=1}} \frac{1}{(g_z d_{1y} - g_y d_{1z})i_x + (g_x d_{1z} - g_z d_{1x})i_y + (g_y d_{1x} - g_x d_{1y})i_z}$$

$$\begin{bmatrix} v_{A1} \\ v_{A2} \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{g}_\perp & \bar{d}_{1\perp} & \bar{i} \end{bmatrix}^{-1} V_A = \begin{bmatrix} (\bar{g} \times \bar{i}) & (\bar{d}_1 \times \bar{i}) & \bar{i} \end{bmatrix}^{-1} V_A =$$

$$\Delta \begin{bmatrix} d_{1x}i_y^2 - d_{1y}i_x i_y + d_{1x}i_z^2 - d_{1z}i_x i_z & d_{1y}i_x^2 - d_{1x}i_y i_x + d_{1y}i_z^2 - d_{1z}i_y i_z & d_{1z}i_x^2 - d_{1x}i_z i_x + d_{1z}i_y^2 - d_{1y}i_y i_z \\ -g_x i_y^2 + g_y i_x i_y - g_x i_z^2 + g_z i_x i_z & -g_y i_x^2 + g_x i_y i_x - g_y i_z^2 + g_z i_y i_z & -g_z i_x^2 + g_x i_z i_x - g_z i_y^2 + g_y i_y i_z \\ i_x/\Delta & i_y/\Delta & i_z/\Delta \end{bmatrix} V_A$$

$$\triangleq \mathrm{J}_{v_A, V_A}(\bar{x}_\beta) V_A \tag{4-16}$$

For a known pose in the $\mathcal{B}$ frame, Eq. (4-2), the decomposition Jacobian matrix $\mathrm{J}_{v_A,V_A}(\bar{x}_\beta)$ can be found and the velocity components in the two rotation directions are found via a linear transformation. The bottom row of Eq. (4-16) finds the out of plane velocity of V_A, which should be zero as the linkage only allows planar movements.

[5] Normalization subscript n is omitted for better readability.

[6] The inverse exists because of the angle conditions in Section 4-2-2 force a non-singular configuration.

Velocity at hydraulic connection joint D The v_{A2} component that is linked to the second hinge (in R) scales linearly with the arm length of d_2. The component that is left in joint D is found via fractional length α.

$$\bar{v}_{D2} = (1-\alpha)\bar{d}_{2n\perp} v_{A2} \tag{4-17}$$

The magnitude of the component of the first hinge is dependent on the relative length of $\bar{g}$ and the vector from the floor joint to D called $\bar{g}'$.

$$v_{D1} = \frac{|\bar{g}'|}{|\bar{g}|} v_{A1} = \frac{|(\bar{d}_1 - (1-\alpha)\bar{d}_2)|}{|\bar{g}|} v_{A1} \tag{4-18}$$

The direction of the velocity component rotates a little, the new direction and the total velocity component is found by[7]:

$$\begin{aligned} \bar{v}_{nD1} &= \bar{g}'_{\perp} = \frac{\bar{g}'}{|\bar{g}'|} \times \bar{i} \\ \bar{v}_{D1} &= v_{D1}\bar{v}_{nD1} = \frac{|\bar{g}'|}{|\bar{g}|} v_{A1} \left(\frac{\bar{g}'}{|\bar{g}'|} \times \bar{i} \right) \end{aligned} \tag{4-19}$$

The velocity in joint D is now found by summing Eq. (4-19) and Eq. (4-17):

$$\begin{aligned} V_D &= \bar{v}_{D1} + \bar{v}_{D2} \\ &= \frac{|\bar{g}'|}{|\bar{g}|} \left(\frac{\bar{g}'}{|\bar{g}'|} \times \bar{i} \right) v_{A1} + (1-\alpha)(\bar{d}_{2n} \times \bar{i}) v_{A2} \\ &= \begin{bmatrix} \frac{|\bar{g}'|}{|\bar{g}|} \left(\frac{\bar{g}'}{|\bar{g}'|} \times \bar{i} \right) & (1-\alpha)(\bar{d}_{2n} \times \bar{i}) & 0_{3\times 1} \end{bmatrix} \begin{bmatrix} v_{A1} \\ v_{A2} \\ 0 \end{bmatrix} \\ &= \mathrm{J}_{V_D, v_A}(\bar{x}_\beta) \begin{bmatrix} v_{A1} \\ v_{A2} \\ 0 \end{bmatrix} \end{aligned} \tag{4-20}$$

The total Jacobian transformation matrix (3×3) from a velocity in A to a velocity in D can now be formed by recombining Eq. (4-16) and Eq. (4-20):

$$\begin{aligned} \mathrm{J}_{D,A}(\bar{x}_\beta) &\triangleq \mathrm{J}_{V_D, v_A}(\bar{x}_\beta) \mathrm{J}_{v_A, V_A}(\bar{x}_\beta) \\ V_D &= \mathrm{J}_{D,A} V_A \end{aligned} \tag{4-21}$$

Leg length change rate The rate of change of the length of a single leg is found by using the velocity of the joint in D (see Figure 4-6) and the unit direction of a leg from Eq. (4-7) and the Jacobian from Eq. (4-21) (p.44 [28]):

$$\begin{aligned} \dot{\ell} &= \frac{d}{dt}|\bar{l}| = \frac{d}{dt}\sqrt{\bar{l}^T \bar{l}} = \frac{\bar{l}^T V_A}{|\bar{l}|} \\ &= \bar{l}_n^T V_D = \bar{l}_n^T \mathrm{J}_{D,A}(\bar{x}_\beta) V_A \end{aligned} \tag{4-22}$$

[7] Note that if $\alpha = 1$ then $V_D = V_R$ in Figure 4-6.

Using the velocity of point A from Eq. (4-11) in Eq. (4-22) using the triple product identity[8] yields :

$$\begin{aligned}\dot{\ell} &= \bar{l}_n^T(\bar{x}_\beta)\mathrm{J}_{D,A}(\bar{x}_\beta)\left(\dot{\bar{c}} + \bar{\omega} \times R^T_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})\bar{a}^{\mathcal{P}}\right)\\ &= \bar{l}_n^T\mathrm{J}_{D,A}\dot{\bar{c}} + \left(\bar{l}_n^T\mathrm{J}_{D,A}\right)\cdot\left(\bar{\omega}\times R^T_{\mathcal{B}\to\mathcal{P}}\bar{a}^{\mathcal{P}}\right)^T\\ &= \bar{l}_n^T\mathrm{J}_{D,A}\dot{\bar{c}} + \bar{\omega}^T\cdot\left((R^T_{\mathcal{B}\to\mathcal{P}}\bar{a}^{\mathcal{P}})^T\times\left(\bar{l}_n^T\mathrm{J}_{D,A}\right)\right)\\ &= \bar{l}_n^T\mathrm{J}_{D,A}\dot{\bar{c}} + \left(\left[R^T_{\mathcal{B}\to\mathcal{P}}\bar{a}^{\mathcal{P}}\right]_\times \mathrm{J}^T_{D,A}\bar{l}_n\right)^T\bar{\omega}\\ &= \begin{bmatrix}\bar{l}_n^T\mathrm{J}_{D,A} & \left(\left[R^T_{\mathcal{B}\to\mathcal{P}}\bar{a}^{\mathcal{P}}\right]_\times \mathrm{J}^T_{D,A}\bar{l}_n\right)^T\end{bmatrix}\begin{bmatrix}\dot{\bar{c}}\\ \bar{\omega}\end{bmatrix}\end{aligned} \tag{4-23}$$

For all three legs, the first of the matrices on the right side of the last equation from Eq. (4-23) can be found, and stacked into one 3×6 Jacobian matrix that transforms the generalized velocity vector (Eq. (4-4)) to the leg change rates:

$$\dot{\bar{\ell}} = \begin{bmatrix}\dot{\ell}_1 & \dot{\ell}_2 & \dot{\ell}_3\end{bmatrix}^T = \mathrm{J}_{\bar{\ell},\bar{x}}\dot{\bar{x}} \tag{4-24}$$

4-2-4 Solution to forward kinematics

The solution of the forward kinematical problem of the mechanism is found iteratively by applying a Newton-Rhapson scheme. (Similar to [27], [28] and [67] and many others) The "leg rate - generalized velocity" Jacobian from Eq. (4-24) is used in combination with the transformation from the time derivative of the pose vector to generalized velocity (Eq. (4-5)). The superscript + indicates the generalized inverse[9]:

$$\mathrm{J}_{\bar{x}_\beta,\bar{\ell}} = \left[\mathrm{J}_{\bar{\ell},\bar{x}_\beta}\right]^+ = \mathrm{J}_{\bar{x}_\beta,\bar{x}}\left[\mathrm{J}_{\bar{\ell},\bar{x}}\right]^+ \tag{4-25}$$

$$\bar{x}_{\beta_{j+1}} = \bar{x}_{\beta_j} + \left[\mathrm{J}_{\bar{\ell},\bar{x}_\beta}(\hat{x}_{\beta_j})\right]^+\left(\bar{\ell}_{\text{measured}} - \hat{\bar{\ell}}(\hat{\bar{x}}_{\beta_j})\right) \tag{4-26}$$

Selecting only the heave-roll-pitch DoFs from $\bar{x}_\beta$ shrinks the dimension from $\mathrm{J}_{\bar{\ell},\bar{x}_\beta}(\hat{x}_{\beta_j}) \in \mathbb{R}^{6\times 3}$ to a square matrix in $\mathbb{R}^{3\times 3}$ making the problem exactly determined (if the reduced $\mathrm{J}_{\bar{\ell},\bar{x}_\beta}$ is full rank).

The number of iterations required to converge to the correct pose is dependent on the initial guess and is typically in the order of 1 to 3. Starting the iteration at the pose from the previous timestep improves the convergence. Execution times of the forward kinematics calculation scripts as presented in this section are 1 to 2 kHz, and are suitable for real time application where the update frequency requirement is lower.

[8] Triple product identity $\bar{a}\cdot(\bar{b}\times\bar{c}) = \bar{b}\cdot(\bar{c}\times\bar{a}) = \bar{c}\cdot(\bar{a}\times\bar{b})$

[9] The Moore-Penrose pseudoinverse solves the (real valued) ordinary least squares problem $y = Xb$ by $X^+y = (X^TX)^{-1}X^Ty = \hat{b}$. If the problem is exactly determined and nonsingular the pseudo inverse is equal to the matrix inverse X^{-1}.

4-3 Dynamics

4-3-1 Platform dynamics in an inertial frame

If it is assumed that the boat is fixed in space, the body frame of the boat can be regarded as an inertial reference frame ($\mathcal{B} = \mathcal{I}$). Then the dynamics of the platform can be derived via the Newton-Euler method. The forces and moments are combined into the vector $\bar{\tau}^{\mathcal{B}}$ and consist of three linear terms and three angular terms.

$$\begin{aligned} \sum \tau^{\mathcal{B}}_{\text{linear}} &= \frac{d}{dt}\left(\mathrm{M}\dot{\bar{c}}\right) \\ &= \mathrm{M}\ddot{\bar{c}} = mI_{3\times 3}\ddot{\bar{c}}^{\mathcal{B}} \end{aligned} \tag{4-27}$$

Where m is the platform and load combined mass and $\bar{c}$ is the vector pointing from the origin of the boat fixed reference frame $\mathcal{B}$ to the origin of the platform frame in $\mathcal{P}$. In this paragraph it is assumed that *the origin of $\mathcal{P}$ is in the* CoG *of the loaded platform.* The time derivative is expanded into an angular acceleration part and an angular rate dependent part. This gives the Euler equations as:

$$\begin{aligned} \sum \tau^{\mathcal{B}}_{\text{angular}} &= \frac{d}{dt}\left(R_{P\to B}\mathrm{I}^{\mathcal{P}}_p\bar{\omega}^{\mathcal{P}}\right) \\ &= R_{P\to B}\mathrm{I}^{\mathcal{P}}_p\dot{\bar{\omega}}^{\mathcal{P}} + R_{P\to B}\left(\bar{\omega}^{\mathcal{P}} \times (\mathrm{I}^{\mathcal{P}}_p\bar{\omega}^{\mathcal{P}})\right) \\ &= \mathrm{I}^{\mathcal{B}}_p\dot{\bar{\omega}}^{\mathcal{B}} + [\bar{\omega}^{\mathcal{B}}]_\times \mathrm{I}^{\mathcal{B}}_p\bar{\omega}^{\mathcal{B}} \end{aligned} \tag{4-28}$$

With $\mathrm{I}^{\mathcal{B}}_p = R_{P\to B}\mathrm{I}^{\mathcal{P}}_p R^T_{P\to B}$ the rotational inertia matrix of the platform expressed in the boat fixed coordinates and therefore dependent on the angular pose $\bar{\beta}$. Combined and rearranged and using the Jacobian projection to find the force vector from the leg forces the system becomes:

$$\begin{bmatrix} mI_{3\times 3}\ddot{\bar{c}}^{\mathcal{B}} \\ \mathrm{I}^{\mathcal{B}}_p\dot{\bar{\omega}}^{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} 0 \\ -[\bar{\omega}^{\mathcal{B}}]_\times \mathrm{I}^{\mathcal{B}}_p\bar{\omega}^{\mathcal{B}} \end{bmatrix} + \bar{f}_{\text{gravity}} + \mathrm{J}^T_{\ell,x}\bar{f}_{\text{legs}} \tag{4-29}$$

4-3-2 Ship dynamics

The boat's reference frame $\mathcal{B}$ is expressed relative to the inertial "North-East-Down" frame. The inertia matrix is not constant in an inertial frame $\mathcal{I}$ if $\mathcal{B}$ rotates with respect to $\mathcal{I}$. Hence, it is convenient to express the equations of motion in a body-fixed coordinate system. To derive the ship dynamics the method and coordinates presented in Fossen (2005) [43][10] are employed. A positional vector ($\bar{\eta}$) and a velocity vector ($\bar{\nu}$) are defined that are not simply related by time differentiation but also by a rotation dependent transformation (Eq. (2-15)):

$$\dot{\bar{\eta}} = \mathrm{J}_{\bar{\eta},\bar{\nu}}(\bar{\eta})\bar{\nu} \tag{4-30}$$

For time domain simulation the Oglivie transformation of the Cummins equation Section 2-2 is used to fit the frequency dependent hydrocoefficients in the time domain equation. These hydrocoefficients are for example calculated from strip theory in a hydrodynamic code, such as

[10] Section 6.2

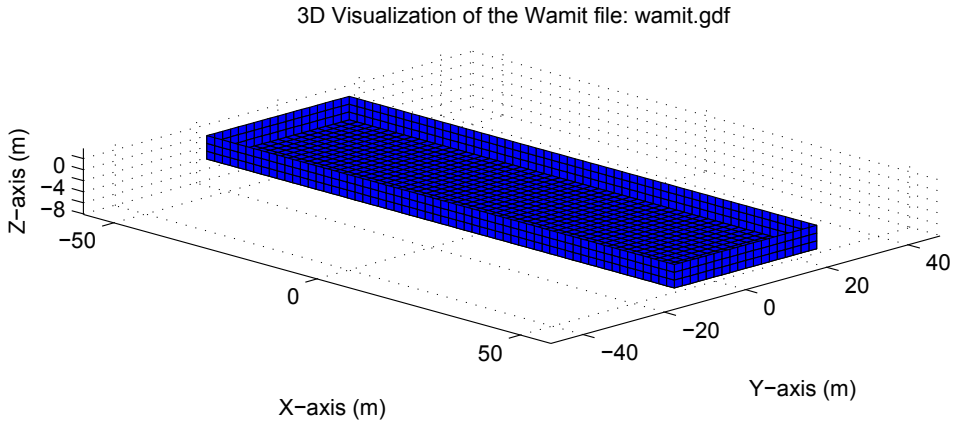

Figure 4-7: Visualization of the barge panel model used in the WAMIT hydrodynamic calculations.

WAMIT and are mainly dependent on hull shape. The radiation forces ($\bar{\mu}$) are approximated by a state space model ($\mathrm{A}_{rad}, \mathrm{B}_{rad}, \mathrm{C}_{rad}$) that estimates the retardation function. The system of equations describing the ship motion is:

$$\begin{aligned} \dot{\bar{\eta}} &= \mathrm{J}_{\bar{\eta},\bar{\nu}}(\bar{\eta})\bar{\nu} \\ \mathrm{M}_s\dot{\bar{\nu}} + \bar{\mu} + g(\bar{\eta}) &= \tau \\ \dot{\bar{\chi}} &= \mathrm{A}_{rad}\bar{\chi} + \mathrm{B}_{rad}\bar{\nu} \\ \bar{\mu} &= \mathrm{C}_{rad}\bar{\chi} \qquad \text{where } \bar{\chi}(0) = \bar{0} \end{aligned} \tag{4-31}$$

With the mass matrix,

$$\mathrm{M}_s = \mathrm{M}_{RB} + \mathrm{J}^{*T}\mathrm{A}_\infty\mathrm{J}^* \tag{4-32}$$

Where J^* is the geometry defined linear transformation from the hydrodynamic equilibrium frame $\mathcal{H}$ to the body frame $\mathcal{B}$ and A_∞ and B_∞ are the infinite period hydrodynamic added mass and damping. These last two constants are jointly estimated (if not provided from WAMIT) with the state space model for radiation terms. Note that the infinite period damping term from the Oglivie transformation of the Cummins equation $\mathrm{D}\bar{\nu} = \mathrm{J}^{*T}\mathrm{B}_\infty\mathrm{J}^*\bar{\nu}$ is dropped because the radiation term is estimated in the form of the original Cummins equation (See Section 2-2-3). The rigid body mass matrix M_{RB} is diagonal if the origin is placed in the CoG. If it is placed in another place, say a body fixed vector $\bar{r}_{o_b g_s}$ away from the origin (see Figure 4-3), the rigid body mass matrix becomes[69]:[11]

$$\mathrm{M}_{RB} = \begin{bmatrix} m\mathrm{I}_{3\times3} & -m[\bar{r}_{o_b g_s}]_\times \\ m[\bar{r}_{o_b g_s}]_\times & \mathrm{I}_g - m[\bar{r}_{o_b g_s}]_\times^2 \end{bmatrix} \tag{4-33}$$

Where m is the mass, I_g is the inertia seen from the CoG that is described in radii of inertia as $\mathrm{I}_g(i,j) = mr_{ij}^2$ with $r_{ij} = 0$ for $i \neq j$, $\mathrm{I}_{3\times3}$ is a identity matrix and the subscript $\times$ denotes the skew symmetric cross product matrix.

[11] Proof of symmetry: $\bar{r} \triangleq \begin{bmatrix} x & y & z \end{bmatrix}^T$, $m[\bar{r}]_\times = m\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$ so $m[\bar{r}]_\times^T = -m[\bar{r}]_\times$ and $m[\bar{r}]_\times^2 = m\begin{bmatrix} -(y^2+z^2) & xy & xz \\ xy & -(x^2+z^2) & yz \\ xz & yz & -(x^2+y^2) \end{bmatrix} = m[\bar{r}]_\times^{2^T}$ Therefore $\mathrm{M}_{RB} = \mathrm{M}_{RB}^T$

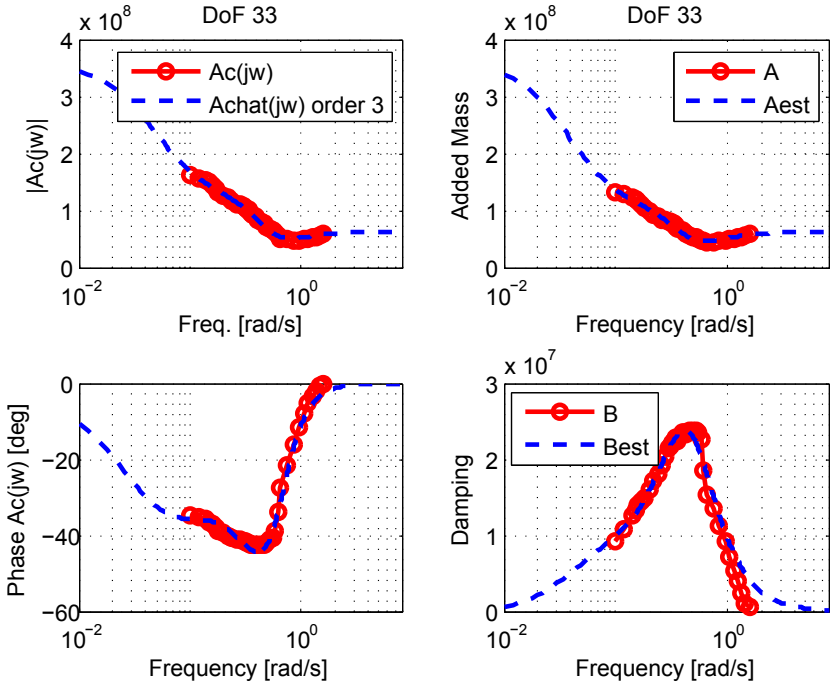

Figure 4-8: Estimation heave-heave retardation function. Left the known values from the complex retardation transfer function $A_{c33}(j\omega) = \left[\frac{B_{33}(j\omega)}{j\omega} + A_{33}(j\omega)\right]$ and the estimation of this function via a LTI state space object is shown. Right the reconstructed added mass and damping from approximated retardation. Estimation order is chosen such that $\hat{B}(\omega) > 0$ to ensure passivity and the transfer function is stable. Methods from [48], barge WAMIT modeling by GustoMSC.

4-3-3 Coupled ship and robot dynamics

Considering the platform on the floating boat and describing the dynamics from the moving reference frame of the ship introduces fictitious forces in the dynamics.[12] The platform and boat have a relative velocity and acceleration, this introduces forces if the system is described from the ship's reference frame $\mathcal{B}$. Therefore the dynamics found for the ship and for the platform in Section 4-3-1 and Section 4-3-2 cannot be coupled straightforward.

Next to these gyroscopic forces, the velocities and the pose are not simply linked through time differentiation. They require pose dependent linear transformations, for the platform described in Eq. (4-5) and for the boat described in Eq. (4-30). Both of these transformation are based on the Euler angles and are therefore prone to singularities and need careful handling.

We want to derive a description for the dynamics of the coupled system that has two important properties: *1. We want to use the minimal set of coordinates 2. We do not want Lagrange multipliers in the equations.* The first condition is because simulation speed considerations,

[12] Rixen (2011) [70] Chapter 2 on rotor dynamics Remark in Section 2.1.1. regarding the Coriolis/gyroscopic and centrifugal terms: "Physical interpretation of Coriolis forces - Coriolis forces (also sometimes called gyroscopic forces) and centrifugal forces are in fact fictitious forces in the sense that they appear in the equation of motion of the system when they are expressed for positions measured in a moving (i.e. rotating) frame (similarly to deceleration forces experienced in a lift)."

whereas the second requires a formulation that implicitly complies to the constraints and is not prone to constraint drift in forward time simulation. Hence larger step sizes should be possible. The method employed to link these two systems mainly follows the Lagrangian formulation of the Boltzmann-Hamel equations of motion for single and serial multi body systems presented in From (2012) [71] and [72]. The key observation required to utilize this theory, is that by considering only two bodies, and not all the linkage masses, *the system essentially becomes a serial mechanism* with a nonlinear linkage. But regardless the complex linkage, it is a single chain with two masses.

The dynamics are derived are very similar to the Boltzmann-Hamel equations of motion for single bodies. Calculating the Lagrange equations in terms of local position and velocity, and a mapping to the corresponding quasi-velocities, that are not necessary a time derivative of the position variables, gives the relations for the global system in Lagrangian form. The key points in the derivation of the method are stated here.

Kinetic energy First we require an expression for the kinetic energy of the system ($\mathcal{K}$). To arrive at an expression for this energy we calculate the kinetic energy for both bodies individually and sum them. Because we want a minimal set of coordinates for various reasons (simulation speed for instance) and the platform motion relative to the ship is only in the $z^{\mathcal{B}}$ and rolling and pitching movement (no rotation around $z^{\mathcal{B}}$) we define the selection matrix H and conjugate selection matrix $\tilde{\mathrm{H}}$ as:

$$\mathrm{H} \triangleq \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T \quad \tilde{\mathrm{H}} \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{4-34}$$

We now redefine $\bar{x}_\beta$ from Eq. (4-2) and $\dot{\bar{x}}$ Eq. (4-4) to only contain the heave, roll and pitch.

$$\begin{aligned} \mathrm{H}^T \bar{x}_\beta &\rightarrow \bar{x}_\beta \triangleq \begin{bmatrix} z & \phi & \theta \end{bmatrix}^T \\ \mathrm{H}^T \dot{\bar{x}} &\rightarrow \dot{\bar{x}} \triangleq \begin{bmatrix} w & p & q \end{bmatrix}^T \end{aligned} \tag{4-35}$$

This reduces the rotation from Eq. (4-1) and the "ground"-to-euler transformation E' (conjugate of E in Eq. (4-3)) to:

$$R_{\mathcal{B}\rightarrow\mathcal{P}}(\varphi,\theta) = \begin{bmatrix} c_\theta & 0 & -s_\theta \\ s_\varphi s_\theta & c_\varphi & s_\varphi c_\theta \\ c_\varphi s_\theta & -s_\varphi & c_\varphi c_\theta \end{bmatrix} \quad [E'(\varphi,\theta)]^{-1} = \begin{bmatrix} \frac{1}{c_\theta} & 0 & 0 \\ 0 & 1 & 0 \\ \tan\theta & 0 & 1 \end{bmatrix} \tag{4-36}$$

The kinetic energy of a body i with rigid body inertia I_i and velocity $\bar{V}_{0i}^{\mathcal{Y}_i}$ with respect to the inertial frame, denoted by the subscript 0, expressed in body fixed local frame $\mathcal{Y}_i$ is found as:

$$\begin{aligned} \mathcal{K}_i &= \frac{1}{2}(\bar{V}_{0i}^{\mathcal{Y}_i})^T \mathrm{I}_i \bar{V}_{0i}^{\mathcal{Y}_i} \\ &= \frac{1}{2}(\bar{V}_{0i}^{\mathcal{I}})^T \mathrm{Ad}_{\mathcal{I}\mathcal{Y}}^T \mathrm{I}_i \mathrm{Ad}_{\mathcal{I}\mathcal{Y}} \bar{V}_{0i}^{\mathcal{I}} \end{aligned} \tag{4-37}$$

Where the adjugate matrix (with capital Ad) transforms the velocity from one frame to another using twists.[13] For example the velocity reference frame transformation is performed by the adjugate matrix and is constructed for the transformation from $\mathcal{B}$ to $\mathcal{P}$ as:

$$\begin{aligned}\dot{\bar{x}}^{\mathcal{P}} &= \mathrm{Ad}_{\mathcal{BP}}(\bar{x}_\beta)\dot{\bar{x}}^{\mathcal{B}} = \begin{bmatrix} R_{\mathcal{B}\to\mathcal{P}} & R_{\mathcal{B}\to\mathcal{P}}[\bar{c}]_\times^T \\ 0 & R_{\mathcal{B}\to\mathcal{P}} \end{bmatrix}\dot{\bar{x}}^{\mathcal{B}} \\ &= \begin{bmatrix} \begin{bmatrix} c_\theta & 0 & -s_\theta \\ s_\varphi s_\theta & c_\varphi & s_\varphi c_\theta \\ c_\varphi s_\theta & -s_\varphi & c_\varphi c_\theta \end{bmatrix} & \begin{bmatrix} c_\theta & 0 & -s_\theta \\ s_\varphi s_\theta & c_\varphi & s_\varphi c_\theta \\ c_\varphi s_\theta & -s_\varphi & c_\varphi c_\theta \end{bmatrix}\begin{bmatrix} 0 & z & 0 \\ -z & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ 0_{3\times 3} & \begin{bmatrix} c_\theta & 0 & -s_\theta \\ s_\varphi s_\theta & c_\varphi & s_\varphi c_\theta \\ c_\varphi s_\theta & -s_\varphi & c_\varphi c_\theta \end{bmatrix} \end{bmatrix}\dot{\bar{x}}^{\mathcal{B}}\end{aligned} \tag{4-38}$$

The kinetic energy of the platform is now found using the platform rigid body inertia matrix $\mathrm{M}_{RBp} \in \mathbb{R}^{6\times 6}$ with the reduction matrices H from Eq. (4-34) and the velocity of the platform with respect to the inertial frame expressed in the platform frame $\mathcal{P}$ ($\bar{V}_{0p}^{\mathcal{P}}$) as:

$$\begin{aligned}\mathcal{K}_{\text{platform}} =& \frac{1}{2}(\bar{V}_{0p}^{\mathcal{P}})^T \mathrm{M}_{RBp}\bar{V}_{0p}^{\mathcal{P}} \\ =& \frac{1}{2}(\bar{V}_{0b}^{\mathcal{P}} + \bar{V}_{bp}^{\mathcal{P}})^T \mathrm{M}_{RBp}(\bar{V}_{0b}^{\mathcal{P}} + \bar{V}_{bp}^{\mathcal{P}}) \\ =& \frac{1}{2}(\bar{V}_{0b}^{\mathcal{B}} + \bar{V}_{bp}^{\mathcal{B}})^T \mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}}(\bar{V}_{0b}^{\mathcal{B}} + \bar{V}_{bp}^{\mathcal{B}}) \\ =& \frac{1}{2}(\bar{\nu} + \mathrm{H}\dot{\bar{x}})^T \mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}}(\bar{\nu} + \mathrm{H}\dot{\bar{x}}) \\ =& \frac{1}{2}(\bar{\nu}^T + \dot{\bar{x}}^T\mathrm{H}^T)\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}}(\bar{\nu} + \mathrm{H}\dot{\bar{x}}) \\ =& \frac{1}{2}\bar{\nu}^T \underbrace{\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}}}_{\mathrm{M}_{vv}}\bar{\nu} + \bar{\nu}^T \underbrace{\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}}\mathrm{H}}_{\mathrm{M}_{vc}}\dot{\bar{x}} \\ &+ \frac{1}{2}\dot{\bar{x}}^T \underbrace{\mathrm{H}^T\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}}\mathrm{H}}_{\mathrm{M}_{cc}}\dot{\bar{x}}\end{aligned} \tag{4-39}$$

Where we've recognised the velocity of the ship expressed in the ship frame (only heave, roll and pitch) as $\bar{V}_{0b}^{\mathcal{B}} = \mathrm{H}^T\bar{\nu}$ and the velocity of the platform relative to the ship expressed in $\mathcal{B}$ as $\bar{V}_{bp}^{\mathcal{B}} = \dot{\bar{x}}$. Now the total mass matrix for the platform dynamics becomes:

$$\mathrm{M}_p = \begin{bmatrix} \mathrm{M}_{cc} & \mathrm{M}_{vc} \\ \mathrm{M}_{vc}^T & \mathrm{M}_{vv} \end{bmatrix} = \begin{bmatrix} \mathrm{H}^T \\ \mathrm{I}_{6\times 6} \end{bmatrix} \mathrm{M}_{vv} \begin{bmatrix} \mathrm{H} & \mathrm{I}_{6\times 6} \end{bmatrix} \in \mathbb{R}^{9\times 9} \tag{4-40}$$

And the kinetic energy and massmatrix for the ship are found by the rigid body mass plus the infinite period added mass in the body frame (Eq. (4-32)):

$$\begin{aligned}\mathcal{K}_{\text{ship}} &= \frac{1}{2}\bar{V}_{0b}^{\mathcal{B}^T}\mathrm{M}_s\bar{V}_{0b}^{\mathcal{B}} \\ &= \frac{1}{2}\bar{\nu}^T\mathrm{M}_s\bar{\nu}\end{aligned} \tag{4-41}$$

[13] Denoting a body frame by ' and the transformation from base to body by $\bar{x}' = \mathrm{R}\bar{x}$. Translating a velocity vector $\dot{\bar{x}}$ along $\bar{r}$ while the base rotates with $\bar{\omega}$ gives body velocity $\dot{\bar{x}}' = \mathrm{R}(\dot{\bar{x}} + \bar{\omega}\times\bar{r}) = \mathrm{R}(\dot{\bar{x}} - \bar{r}\times\bar{\omega}) = \mathrm{R}\dot{\bar{x}} - \mathrm{R}[\bar{r}]_\times\bar{\omega} = \mathrm{R}\dot{\bar{x}} + \mathrm{R}[\bar{r}]_\times^T\bar{\omega}$ and body rotation rate $\omega' = \mathrm{R}\omega$

The total mass matrix is now found by combining the kinetic energies and using Eq. (4-40) and Eq. (4-41) and defining velocity state $\bar{v}^T = \begin{bmatrix}\dot{\bar{x}}^T & \bar{\nu}^T\end{bmatrix}$ and orientation state $\bar{q}^T = \begin{bmatrix}\bar{x}_\beta^T & \eta^T\end{bmatrix}$ as:

$$\begin{aligned}\mathcal{K} = \mathcal{K}_{\text{platform}} + \mathcal{K}_{\text{ship}} &= \frac{1}{2}\begin{bmatrix}\dot{\bar{x}}^T & \bar{\nu}^T\end{bmatrix}\underbrace{\left(\begin{bmatrix}\mathrm{M}_{cc} & \mathrm{M}_{vc}\\ \mathrm{M}_{vc}^T & \mathrm{M}_{vv}\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & \mathrm{M}_s\end{bmatrix}\right)}_{\mathrm{M}(\bar{q})}\begin{bmatrix}\dot{\bar{x}}\\ \bar{\nu}\end{bmatrix}\\ &= \frac{1}{2}\bar{v}^T\mathrm{M}(\bar{q})\bar{v}\end{aligned} \tag{4-42}$$

4-3-4 Lagrange - Coriolis terms

The Lagrange equations for a configuration ($\bar{q}$) dependent mass matrix and quasi-velocity $\bar{v}$ that is not a direct time derivative of $\bar{q}$ consist of the following derivatives of the Lagrangian L:

$$\begin{aligned}L(\bar{q},\bar{v}) &= \frac{1}{2}\bar{v}^T\mathrm{M}(\bar{q})\bar{v} - \mathcal{U}(\bar{q})\\ \frac{\partial L}{\partial \bar{v}} &= \mathrm{M}(\bar{q})\bar{v}\\ \frac{d}{dt}\left(\frac{\partial L}{\partial \bar{v}}\right) &= \mathrm{M}(\bar{q})\dot{\bar{v}} + \dot{\mathrm{M}}(\bar{q})\bar{v}\\ \frac{\partial L}{\partial \bar{q}} &= \frac{1}{2}\frac{\partial^T\mathrm{M}(\bar{q})\bar{v}}{\partial \bar{q}}\bar{v} - \frac{\partial \mathcal{U}(\bar{q})}{\partial \bar{q}}\end{aligned} \tag{4-43}$$

Now using the velocity transform coupling the position variables to the quasi-velocities defined as:

$$\bar{v} = \mathrm{S}(\bar{q})\dot{\bar{q}} \tag{4-44}$$

Gives the Lagrangian $\tilde{L}$ expressed in the derivative of the position variables ($\bar{q}$):

$$\begin{aligned}\tilde{L}(\bar{q},\bar{v}) &= \frac{1}{2}\dot{\bar{q}}^T\mathrm{S}^T(\bar{q})\mathrm{M}(\bar{q})\mathrm{S}(\bar{q})\dot{\bar{q}} - \mathcal{U}(\bar{q})\\ \frac{\partial \tilde{L}}{\partial \bar{v}} &= \mathrm{S}^T(\bar{q})\frac{\partial L}{\partial \bar{v}}\\ \frac{d}{dt}\left(\frac{\partial \tilde{L}}{\partial \bar{v}}\right) &= \dot{\mathrm{S}}(\bar{q})\frac{\partial L}{\partial \bar{v}} + \mathrm{S}^T(\bar{q})\frac{d}{dt}\left(\frac{\partial L}{\partial \bar{v}}\right)\\ \frac{\partial \tilde{L}}{\partial \bar{q}} &= \frac{\partial L}{\partial \bar{q}} + \frac{\partial^T(\mathrm{S}(\bar{q})\dot{\bar{q}})}{\partial \bar{q}}\frac{\partial L}{\partial \bar{v}}\end{aligned} \tag{4-45}$$

Now Lagrange's equations are found as:

$$\frac{d}{dt}\left(\frac{\partial \tilde{L}}{\partial \bar{v}}\right) - \frac{\partial \tilde{L}}{\partial \bar{q}} = \mathrm{S}^T\tau \tag{4-46}$$

$$\begin{aligned}&\mathrm{M}(\bar{q})\dot{\bar{v}} + \underbrace{\dot{\mathrm{M}}(\bar{q})\bar{v} - \mathrm{S}^{-T}(\bar{q})\frac{1}{2}\frac{\partial^T\mathrm{M}(\bar{q})\bar{v}}{\partial \bar{q}}\bar{v}}_{\text{Multibody Coriolis terms}}\\ &+ \underbrace{\mathrm{S}^{-T}(\bar{q})\left(\dot{\mathrm{S}}^T(\bar{q}) - \frac{\partial^T(\mathrm{S}(\bar{q})\dot{\bar{q}})}{\partial \bar{q}}\right)\mathrm{M}(\bar{q})\bar{v}}_{\text{Coriolis terms identical to single body case}} + \mathrm{S}^{-T}(\bar{q})\frac{\partial \mathcal{U}(\bar{q})}{\partial \bar{q}} = \tau\end{aligned} \tag{4-47}$$

Now for further evaluation of these Coriolis terms without regarding singularities in the mapping from position to quasi-velocity, we define a local position coordinate ϕ that maps to the velocity variables $\bar{v}$.

$$\begin{aligned} \bar{v} &= \mathrm{S}(\phi)\dot{\phi} \\ \mathrm{S}(\phi) &= \left(I - \frac{1}{2}ad_\phi + \frac{1}{6}ad_\phi^2 - ...\right) \end{aligned} \tag{4-48}$$

With ad_ϕ the Lie bracket in matrix notation that looks for a the $SE(3)$ group (3 translations and 3 rotations) in which 3 dimensional rigid body movements are described (with velocity directions defined according to Eq. (2-12)) as:

$$ad_V = \begin{bmatrix} [\bar{\omega}_{0b}^{\mathcal{B}}]_\times & [\bar{V}_{0b}^{\mathcal{B}}]_\times \\ 0_{3\times 3} & [\bar{\omega}_{0b}^{\mathcal{B}}]_\times \end{bmatrix} = \begin{bmatrix} 0 & -r & q & 0 & -w & v \\ r & 0 & -p & w & 0 & -u \\ -q & p & 0 & -v & u & 0 \\ 0 & 0 & 0 & 0 & -r & q \\ 0 & 0 & 0 & r & 0 & -p \\ 0 & 0 & 0 & -q & p & 0 \end{bmatrix} \tag{4-49}$$

By evaluating the local velocities at $\phi = 0$, S^{-1} becomes identity. Now the Coriolis terms from Eq. (4-47) without the multiplication with $\bar{v}$ can be captured in one Coriolis matrix C such that $\tau_{coriolis} = \mathrm{C}\bar{v}$. The entries of this matrix are given by [72]:

$$\mathrm{C}_{ij}(v) = \sum_k \left(\frac{\partial \mathrm{M}_{ij}}{\partial \phi_k} - \frac{1}{2}\frac{\partial \mathrm{M}_{jk}}{\partial \phi_i}\right)\Bigg|_{\phi=0} v_k + \sum_k \left(\frac{\partial \mathrm{S}_{ki}}{\partial \phi_j} - \frac{\partial \mathrm{S}_{kj}}{\partial \phi_i}\right)(\mathrm{M}\bar{v})_k\Bigg|_{\phi=0} \tag{4-50}$$

First part: partials of M_{vv} The first part of Eq. (4-50) consisting of the partial derivatives of the mass matrix given in Eq. (4-42). As only the partial derivatives with respect to pose vector q of the mass matrix are required, the only nonzero terms are in the orientation dependent part of the mass matrix Eq. (4-40).

$$\begin{aligned} \frac{\partial \mathrm{M}_{ij}}{\partial \phi_k} = \frac{\partial \mathrm{M}_{p_{ij}}}{\partial \phi_k} &= \left\{ \begin{bmatrix} \mathrm{H}^T \\ \mathrm{I}_{6\times 6} \end{bmatrix} \frac{\partial \mathrm{M}_{vv}}{\partial \phi_k} \begin{bmatrix} \mathrm{H} & \mathrm{I}_{6\times 6} \end{bmatrix} \right\}_{i,j} \\ &= \left\{ \begin{bmatrix} \mathrm{H}^T \\ \mathrm{I}_{6\times 6} \end{bmatrix} \frac{\partial \left(\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}}\right)}{\partial \phi_k} \begin{bmatrix} \mathrm{H} & \mathrm{I}_{6\times 6} \end{bmatrix} \right\}_{i,j} \end{aligned} \tag{4-51}$$

As the adjoint velocity transformation $\mathrm{Ad}_{\mathcal{BP}}$ only depends on the platform position coordinate ($\bar{x}_\beta$) only the partial derivatives in $\phi_{1..3}$ are non zero. The central part of Eq. (4-51) is expanded using the chain rule of differentiation (M_{RBp} is the platform's rigid body mass matrix which is in itself invariant under pose change):

$$\begin{aligned} \frac{\partial \left(\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}}\right)}{\partial \phi_k} &= \left(\frac{\partial \mathrm{Ad}_{\mathcal{BP}}^T}{\partial \phi_k} \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}}\right) + \left(\mathrm{Ad}_{\mathcal{BP}}^T \mathrm{M}_{RBp} \frac{\partial \mathrm{Ad}_{\mathcal{BP}}}{\partial \phi_k}\right) \\ &= \left(\frac{\partial \mathrm{Ad}_{\mathcal{BP}}^T}{\partial \phi_k} \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}}\right) + \left(\frac{\partial \mathrm{Ad}_{\mathcal{BP}}^T}{\partial \phi_k} \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}}\right)^T \\ &\triangleq \left(\frac{\partial \mathrm{Ad}_{\mathcal{BP}}^T}{\partial \phi_k} \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}}\right)^{\mathrm{sym}} \end{aligned} \tag{4-52}$$

For the second step the symmetry of the rigid body mass is exploited by: $((\mathrm{M}_{RBp}\mathrm{Ad}_{\mathcal{BP}})^T = \mathrm{Ad}^T_{\mathcal{BP}}\mathrm{M}^T_{RBp} = \mathrm{Ad}^T_{\mathcal{BP}}\mathrm{M}_{RBp})$ Eq. (4-51) can now be rewritten as:

$$\begin{aligned}\frac{\partial \mathrm{M}_{ij}}{\partial \phi_k} &= \left\{ \begin{bmatrix} \mathrm{H}^T \\ \mathrm{I}_{6\times 6} \end{bmatrix} \left(\frac{\partial \mathrm{Ad}^T_{\mathcal{BP}}}{\partial \phi_k} \mathrm{M}_{RBp} \mathrm{Ad}_{\mathcal{BP}} \right)^{\mathrm{sym}} \begin{bmatrix} \mathrm{H} & \mathrm{I}_{6\times 6} \end{bmatrix} \right\}_{i,j} \\ &\triangleq \left\{ \frac{\partial \mathrm{M}}{\partial \phi_k} \right\}_{i,j} = \begin{bmatrix} \frac{\partial \mathrm{M}_{1,1}}{\partial \phi_k} & \frac{\partial \mathrm{M}_{1,2}}{\partial \phi_k} & \cdots & \frac{\partial \mathrm{M}_{1,N}}{\partial \phi_k} \\ \frac{\partial \mathrm{M}_{2,1}}{\partial \phi_k} & \frac{\partial \mathrm{M}_{2,2}}{\partial \phi_k} & \cdots & \frac{\partial \mathrm{M}_{2,N}}{\partial \phi_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathrm{M}_{N,1}}{\partial \phi_k} & \frac{\partial \mathrm{M}_{N,2}}{\partial \phi_k} & \cdots & \frac{\partial \mathrm{M}_{N,N}}{\partial \phi_k} \end{bmatrix}_{i,j}\end{aligned} \tag{4-53}$$

$\mathrm{Ad}_{\mathcal{BP}}|_{\phi=0}$ is equal to the adjugate matrix evaluated for $\bar{q}$ ($\mathrm{Ad}_{\mathcal{BP}}(\bar{q})$) as ϕ are pertubation coordinates. The partials of the adjugate matrix Ad, Eq. (4-38), are evaluated at $\phi = 0$ this gives the $\bar{q}$ dependent matrices:

$$\begin{aligned}\left.\frac{\partial \mathrm{Ad}_{\mathcal{BP}}}{\partial \phi_1}\right|_{\phi=0}(\bar{q}) &= \begin{bmatrix} 0 & 0 & 0 & 0 & c_\theta & 0 \\ 0 & 0 & 0 & -c_\phi & s_\phi s_\theta & 0 \\ 0 & 0 & 0 & s_\phi & c_\phi s_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \left.\frac{\partial \mathrm{Ad}_{\mathcal{BP}}}{\partial \phi_2}\right|_{\phi=0}(\bar{q}) &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ c_\phi s_\theta & -s_\phi & c_\phi c_\theta & z s_\phi & z c_\phi s_\theta & 0 \\ -s_\phi s_\theta & -c_\phi & -c_\theta s_\phi & z c_\phi & -z s_\phi s_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\phi s_\theta & -s_\phi & c_\phi c_\theta \\ 0 & 0 & 0 & -s_\phi s_\theta & -c_\phi & -c_\theta s_\phi \end{bmatrix} \\ \left.\frac{\partial \mathrm{Ad}_{\mathcal{BP}}}{\partial \phi_3}\right|_{\phi=0}(\bar{q}) &= \begin{bmatrix} -s_\theta & 0 & -c_\theta & 0 & -z s_\theta & 0 \\ c_\theta s_\phi & 0 & -s_\phi s_\theta & 0 & z c_\theta s_\phi & 0 \\ c_\phi c_\theta & 0 & -c_\phi s_\theta & 0 & z c_\phi c_\theta & 0 \\ 0 & 0 & 0 & -s_\theta & 0 & -c_\theta \\ 0 & 0 & 0 & c_\theta s_\phi & 0 & -s_\phi s_\theta \\ 0 & 0 & 0 & c_\phi c_\theta & 0 & -c_\phi s_\theta \end{bmatrix}\end{aligned} \tag{4-54}$$

The other partials of Ad are zero. The second set of partial derivatives of the inertia matrix is only nonzero for $i = 1..3$ an can be found using the partial derivative matrices of Eq. (4-53) and the velocity vector $\bar{v}$:

$$\begin{aligned}\sum_k \frac{\partial \mathrm{M}_{jk}}{\partial \phi_i} v_k &= \begin{bmatrix} \left\{\frac{\partial \mathrm{M}}{\partial \phi_1}\bar{v}\right\}_1 & \left\{\frac{\partial \mathrm{M}}{\partial \phi_1}\bar{v}\right\}_2 & \cdots & \left\{\frac{\partial \mathrm{M}}{\partial \phi_1}\bar{v}\right\}_N \\ \left\{\frac{\partial \mathrm{M}}{\partial \phi_2}\bar{v}\right\}_1 & \left\{\frac{\partial \mathrm{M}}{\partial \phi_2}\bar{v}\right\}_2 & \cdots & \left\{\frac{\partial \mathrm{M}}{\partial \phi_2}\bar{v}\right\}_N \\ \vdots & \vdots & \ddots & \vdots \\ \left\{\frac{\partial \mathrm{M}}{\partial \phi_N}\bar{v}\right\}_1 & \left\{\frac{\partial \mathrm{M}}{\partial \phi_N}\bar{v}\right\}_2 & \cdots & \left\{\frac{\partial \mathrm{M}}{\partial \phi_N}\bar{v}\right\}_N \end{bmatrix}_{i,j} \\ &= \begin{bmatrix} \left(\frac{\partial \mathrm{M}}{\partial \phi_1}\bar{v}\right) & \left(\frac{\partial \mathrm{M}}{\partial \phi_2}\bar{v}\right) & \left(\frac{\partial \mathrm{M}}{\partial \phi_3}\bar{v}\right) & 0_{9\times 6} \end{bmatrix}^T_{i,j}\end{aligned} \tag{4-55}$$

The partial derivatives of $\mathrm{Ad}_{\mathcal{BP}}$ (Eq. (4-38)) with $\phi = 0$ inserted can now be found. The left part of Eq. (4-50) $\sum_k \left(\frac{\partial \mathrm{M}_{ij}}{\partial \phi_k} - \frac{1}{2}\frac{\partial \mathrm{M}_{jk}}{\partial \phi_i}\right)\Big|_{\phi=0} v_k$ using Eq. (4-53) and Eq. (4-55) now becomes in matrix form:

$$\begin{bmatrix} \mathrm{C}_{cc1} & \mathrm{C}_{vc1} \\ \mathrm{C}_{cv1} & \mathrm{C}_{vv1} \end{bmatrix} = \sum_{k=1}^{3} \left[\frac{\partial \mathrm{M}}{\partial \phi_k}\right] v_k - \frac{1}{2}\begin{bmatrix} \left(\frac{\partial \mathrm{M}}{\partial \phi_1}\bar{v}\right) & \left(\frac{\partial \mathrm{M}}{\partial \phi_2}\bar{v}\right) & \left(\frac{\partial \mathrm{M}}{\partial \phi_3}\bar{v}\right) & 0_{9\times 6} \end{bmatrix}^T \tag{4-56}$$

Second part: partials of S For the second part of Eq. (4-50) (the partials of the local velocity transform) we define the reduced S_r anticipating on the differentiation by dropping out all constant terms and all higher order terms. Eq. (4-48) retains only $-\frac{1}{2}ad_\phi$ which looks for the boat as a Special Euclidean group (SE(3)) as given in Eq. (4-49) but with components $\phi_{4..9}$ instead of u, v, w, p, q, r, reserving the first 3 indices for the platform.

$$\mathrm{S}_{RS} = -\frac{1}{2}ad_\phi = -\frac{1}{2}\begin{bmatrix} [\phi_7, \phi_8, \phi_9]_\times & [\phi_4, \phi_5, \phi_6]_\times \\ 0_{3\times 3} & [\phi_7, \phi_8, \phi_9]_\times \end{bmatrix} \tag{4-57}$$

For the platform S_{rP} all directions except $\phi_i = \phi_{p3..p5}$ are zero (planar movements are constrained) therefore has entries:

$$\mathrm{S}_{RP} = -\frac{1}{2}\mathrm{H}^T ad_\phi \mathrm{H}\Big|_{\phi_{p1,p2,p6}=0} = -\frac{1}{2}\begin{bmatrix} 0 & -\phi_{2p} & \phi_{1p} \\ 0 & 0 & -\phi_{6p} \\ 0 & \phi_{6p} & 0 \end{bmatrix}\Bigg|_{\phi_{p1,p2,p6}=0} = 0_{3\times 3} \tag{4-58}$$

The reduced transformation has only entries in the bottom right corner:

$$\mathrm{S}_R(\phi) = \begin{bmatrix} 0_{3\times 3} & 0_{3\times 6} \\ 0_{6\times 3} & \mathrm{S}_{RS} \end{bmatrix} \tag{4-59}$$

The reason velocity transformations matrices can be diagonally stacked is that the velocity transformation between two consecutive rigid bodies in a serial chain is kinematically independent of the positions and velocities of the other transformations. The link velocity Jacobian is not, using J would result in a lower triangular matrix[71]. Using the mass matrix from Eq. (4-42), the right side of Eq. (4-50) now becomes the matrix given in Eq. (4-60), that is only nonzero for $i > 3$. Note the $\bar{\nu}$ instead of the $\bar{v}$ indicating ship velocities. (exact steps [71])

$$\begin{aligned} \begin{bmatrix} 0_{3\times 3} & 0_{3\times 6} \\ \mathrm{C}_{cv2} & \mathrm{C}_{vv2} \end{bmatrix} &= \sum_k \left(\frac{\partial \mathrm{S}_{ki}}{\partial \phi_j} - \frac{\partial \mathrm{S}_{kj}}{\partial \phi_i}\right)(\mathrm{M}\bar{v})_k\Big|_{\phi=0} \\ &= \sum_k \left(\frac{\partial \mathrm{S}_{R_{ki}}}{\partial \phi_j} - \frac{\partial \mathrm{S}_{R_{kj}}}{\partial \phi_i}\right)(\mathrm{M}\bar{v})_k \\ &= \sum_{k,m} \left(\frac{\partial \mathrm{S}_{R_{mi}}}{\partial \phi_k} - \frac{\partial \mathrm{S}_{R_{mk}}}{\partial \phi_i}\right) v_k \mathrm{M}_m = \begin{bmatrix} 0_{3\times 3} & 0_{3\times 6} \\ 0_{6\times 3} & 2(-\frac{1}{2}ad_{\bar{\nu}}^T) \end{bmatrix} \mathrm{M}(\bar{q}) \end{aligned} \tag{4-60}$$

Now the total Coriolis matrix can be formed combining Eq. (4-56) and Eq. (4-60).

$$\mathrm{C}(\bar{v}) = \begin{bmatrix} \mathrm{C}_{cc1} & \mathrm{C}_{vc1} \\ \mathrm{C}_{cv1} & \mathrm{C}_{vv1} \end{bmatrix} + \begin{bmatrix} 0_{3\times 3} & 0_{3\times 6} \\ \mathrm{C}_{cv2} & \mathrm{C}_{vv2} \end{bmatrix} \tag{4-61}$$

4-3-5 Total dynamics

The coupled dynamics are now found from the mass in equation Eq. (4-42), velocity state $\bar{v}^T = \begin{bmatrix}\dot{\bar{c}}^T & \bar{\nu}^T\end{bmatrix}$, platform pose $\bar{x}_\beta$ and ship location and orientation in $\bar{\eta}$. The Coriolis terms from Eq. (4-56) and Eq. (4-60) together in $\mathrm{C}(\bar{v})$ from Eq. (4-61) are used. So is the leg directional Jacobian $\mathrm{J}_{\bar{\ell},\bar{x}}$ Eq. (4-24), selection matrix H from Eq. (4-34), hydrodynamic retardation terms $\bar{\mu}$ from the parallel state space system Eq. (4-31), the hydrostatic matrix G, and the wave forces $\bar{\tau}_W$ discussed in Section 4-4:

$$\underbrace{\mathrm{M}(\bar{x}_\beta)\dot{\bar{v}}}_{\text{Inertia}} + \underbrace{\mathrm{C}(\bar{v})\bar{v}}_{\text{Coriolis}} + \underbrace{\mathrm{G}(\bar{\eta})\begin{bmatrix}0\\ \eta\end{bmatrix}}_{\text{Hydrostatic}} = \underbrace{f(\mathrm{J}^T_{\bar{\ell},\bar{x}})\tau_{leg}}_{\text{Leg loads}} + \underbrace{\begin{bmatrix}0\\ \bar{\mu}\end{bmatrix}}_{\text{Wave Radiation}} + \underbrace{\begin{bmatrix}0\\ \bar{\tau}_W\end{bmatrix}}_{\text{External Waves}} + \tau_{\text{gravity}} \tag{4-62}$$

Gravity As the dynamics are described in the body fixed frame and the gravity origins from the inertial frame, the forces need to be transformed from the inertial frame ($\mathcal{I}$) to the boat frame ($\mathcal{B}$). To accomplish this we use the body fixed vectors from the origins to the center of gravities of both bodies. For the ship this vector is written as $\bar{r}^{\mathcal{B}}_{o_b g_s}$ and for the platform as $\bar{r}^{\mathcal{P}}_{o_p g_p}$. Further the relative pose vector that defines the pose of the platform relative to the ship $\bar{x}_\beta$ is used, in particular the (heave only) translation in $\bar{c}$ and the roll and pitch Euler angles allowing the formation of rotation matrix $R_{\mathcal{B}\to\mathcal{P}}$ (Eq. (4-1) and the reduced Eq. (4-36)). We employ the notation m_s and m_p for ship and platform mass and g for the gravitational acceleration.

The main principle is projecting the inertial ship gravity on origin of the ship local coordinate system (not in CoG) and equivalently projecting the platform gravity in the platform base but rotated along with the boat base $\mathcal{B}$. Then the planar components are separated and translated to the ship's base as the planar movements of the platform are constrained. First the gravity of the ship mass:

$$\begin{aligned}
\bar{f}^{\mathcal{I}}_{cogm_s} &= \begin{pmatrix}0\\0\\m_s g\end{pmatrix} \\
\bar{f}^{\mathcal{B}}_{cogm_s} &= R_{\mathcal{I}\to\mathcal{B}}\begin{pmatrix}0\\0\\m_s g\end{pmatrix} \\
\bar{\tau}^{\mathcal{B}}_{o_b m_s} &= \begin{bmatrix}R_{\mathcal{I}\to\mathcal{B}}\\ [\bar{r}^{\mathcal{B}}_{o_b g_s}]_\times R_{\mathcal{I}\to\mathcal{B}}\end{bmatrix}\begin{pmatrix}0\\0\\m_s g\end{pmatrix}
\end{aligned} \tag{4-63}$$

Similar for the platform mass the rotated and translated gravity force (6×1) in the origin of

the platform frame is found as:

$$\begin{aligned} \bar{f}^{\mathcal{I}}_{cogm_p} &= \begin{pmatrix} 0 \\ 0 \\ m_p g \end{pmatrix} \\ \bar{\tau}^{\mathcal{B}}_{o_p m_p} &= \begin{bmatrix} R_{\mathcal{I}\to\mathcal{B}} \\ [R^T_{\mathcal{B}\to\mathcal{P}} \bar{r}^{\mathcal{P}}_{o_p g_p}]_\times R_{\mathcal{I}\to\mathcal{B}} \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ m_p g \end{pmatrix} \end{aligned} \tag{4-64}$$

Now this gravity component of the platform is splitted into a heave-roll-pitch part and a planar part. Where the second is transformed to the boat's origin as the platform is constrained in planar movements relative to the ship. The selection matrices in Eq. (4-34) are used.

$$\begin{aligned} &\dot{\bar{x}} \text{ aligned:} && \mathrm{H}^T \begin{bmatrix} R_{\mathcal{I}\to\mathcal{B}} \\ [R^T_{\mathcal{B}\to\mathcal{P}} \bar{r}^{\mathcal{P}}_{o_p g_p}]_\times R_{\mathcal{I}\to\mathcal{B}} \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ m_p g \end{pmatrix} \\ &\bar{\nu} \text{ alligned:} && \begin{bmatrix} \mathrm{I}_{3\times 3} \\ [\bar{c}]_\times \end{bmatrix} \tilde{\mathrm{H}}^T \begin{bmatrix} R_{\mathcal{I}\to\mathcal{B}} \\ [R^T_{\mathcal{B}\to\mathcal{P}} \bar{r}^{\mathcal{P}}_{o_p g_p}]_\times R_{\mathcal{I}\to\mathcal{B}} \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ m_p g \end{pmatrix} \end{aligned} \tag{4-65}$$

Integrated quasi velocity to pose The dynamics are integrated to find the solution in time. The pose of the ship and platform are not simply integrals of the velocity vector. To obtain the pose the transformation from Eq. (4-3) ("ground"-to-euler $[E(\bar{\beta})]^{-1}$) is reduced to 2×2 to exclude the constrained yaw. Rotation $R_{\mathcal{B}\to\mathcal{P}}(\bar{\beta})$ from Eq. (4-1) is not needed as the heave of the platform in $\bar{c}$ is already in the $\mathcal{B}$ frame. Rotation $R_{\mathcal{I}\to\mathcal{B}}$ from Eq. (2-13) and the ship's "body"-to-euler attitude transformation $[E'(\phi_s, \theta_s, \psi_s)]^{-1}$, Eq. (2-14), find the ship pose vector from the ship velocities. The total transformation matrix J_{qv} looks as:

$$\begin{aligned} \dot{\bar{q}}^T &= \begin{bmatrix} \dot{z}^{\mathcal{B}}_P & \dot{\phi}_P & \dot{\theta}_P & \dot{N}^{\mathcal{I}} & \dot{E}^{\mathcal{I}} & \dot{D}^{\mathcal{I}} & \dot{\phi}_S & \dot{\theta}_S & \dot{\psi}_S \end{bmatrix}^T \\ &= \begin{bmatrix} \dot{\bar{x}}_\beta \\ \dot{\bar{\eta}} \end{bmatrix} = \begin{bmatrix} 1 & 0_{1\times 2} & 0_{1\times 3} & 0_{1\times 3} \\ 0 & [E(\phi_P, \theta_P)]^{-1} & 0_{2\times 3} & 0_{2\times 3} \\ 0 & 0_{3\times 2} & R^T_{\mathcal{I}\to\mathcal{B}} & 0_{3\times 3} \\ 0 & 0_{3\times 2} & 0_{3\times 3} & [E'(\phi_S, \theta_S, \psi_S)]^{-1} \end{bmatrix} \begin{bmatrix} \dot{\bar{x}} \\ \bar{\nu} \end{bmatrix} \\ &= \mathrm{J}_{qv} \begin{bmatrix} \dot{z}^{\mathcal{B}}_P & \omega^{\mathcal{B}}_{xP} & \omega^{\mathcal{B}}_{yP} & u & v & w & p & q & r \end{bmatrix}^T = \mathrm{J}_{qv} \bar{v}^T \end{aligned} \tag{4-66}$$

Transforming the quasi velocities and integrating allows to track the vessel's position in the NED frame and generate a visualization.

4-4 Wave loads

In this section a the construction of the only disturbance term that is modeled is discussed: the external wave loads. Other disturbances such as wind and currents could be included. But as currents produce mainly forces in the planar part of the dynamics, which is assumed compensated by moorings or by DP, this factor is left out of the model. Wind loads are often

aligned with the principal wave load direction. As one random disturbance suffices to show the principle of the system therefore the wind loads are omitted too.

The waves are generated from summed harmonic components of a spectral density formula in combination with random phases. The wave loads are than found via the FTFs that were calculated by the frequency domain hydrodynamic package (WAMIT).

Wave spectrum The deep water assumption, depth h is more than half the wavelength $\frac{1}{2}\lambda$ [39], is used to derive the linear gravitational wave propagation relations. The phase speed is given by:

$$c_p = \frac{g}{\omega} \tag{4-67}$$

Where g is the gravitational constant[14] and ω is the wave's angular frequency ($\omega = 2\pi/T$). The phase speed can be used to find the wave number k as:

$$\begin{aligned} c_p &= \frac{\omega}{k} = \frac{\lambda}{T} \\ \rightarrow k &= \frac{\omega^2}{g} \end{aligned} \tag{4-68}$$

Now we can find the surface elevation ξ caused by a single frequency wave component of a parallel wave front traveling in the x direction as a function of time t and location:

$$\begin{aligned} \xi &= \zeta_\omega \cos\left(kx - \omega t\right) \\ &= \zeta_\omega \cos\left(\frac{\omega^2}{g}x - \omega t\right) \end{aligned} \tag{4-69}$$

The magnitude of the wave depends on the general condition of the sea called sea state. The sea state is generally measured in the significant wave height $H_{1/3}$, the mean height of the highest one third of the waves, and the mean wave period T_1. The sea state is captioned in a wave energy spectrum, the Pierson-Moskowitz Spectrum[15], whose formula is shown in Figure 4-9 and given by:

$$\frac{S(\omega)}{H_{1/3}^2 T_1} = \frac{0.11}{2\pi}\left(\frac{\omega T_1}{2\pi}\right)^{-5} \exp\left[-0.44\left(\frac{\omega T_1}{2\pi}\right)^{-4}\right] \tag{4-70}$$

The waves are directionally spread in three, discrete, 45° separated parallel wave fronts. The part of the wave energy flowing in direction α is found by:

$$S(\omega, \alpha) = \frac{2}{\pi}\cos^2(\alpha - \mu)S(\omega) \tag{4-71}$$

[14] $g = 9.81 m/s^2$

[15] Pierson-Moskowitz Spectrum is recommended for fully developed seas. Two other frequently used spectrum formulas are the JONSWAP Spectrum for limited fetch length: $S(\omega) = 155\frac{H_{1/3}^2}{T_1^4\omega^5}\exp\left(\frac{-944}{T_1^4\omega^4}\right)(3.3)^{\wedge}\exp\left[-\left(\frac{0.191\omega T_1 - 1}{\sqrt{2}\sigma}\right)^2\right]$ with $\sigma = \begin{cases} 0.07 \text{ if } \omega \leq 5.24/T_1 \\ 0.09 \text{ if } \omega > 5.24/T_1 \end{cases}$ and the Bretschneider Spectrum: $S(\omega) = \frac{173 H_{1/3}^2/T_1^4}{\omega^5}\exp\left[-\frac{691/T_1^4}{\omega^4}\right]$

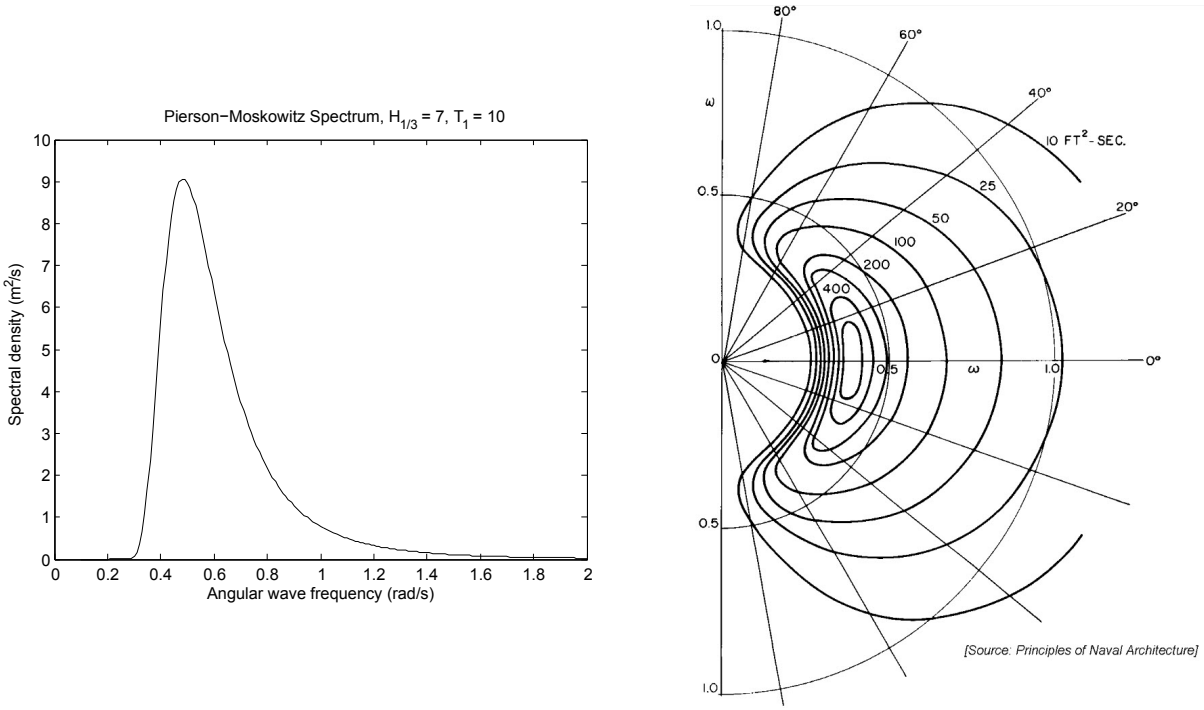

Figure 4-9: Wave generation via spectral densities. The exact waves are random but their energy distribution is known. Left: Pierson-Moskowitz Spectral density plot, Right: Cosine squared directional spreading [39].

Where μ is the predominant heading of the waves [37]. Now the wave amplitude from Eq. (4-69) is found by [39]:

$$\begin{aligned}\zeta(\omega_k, \alpha_l) &= \sqrt{2 \int_{\omega_k - d\omega/2}^{\omega_k + d\omega/2} S(\omega, \alpha_l) d\omega} \\ &\approx \sqrt{2 S(\omega_k, \alpha_l) \Delta\omega}\end{aligned} \tag{4-72}$$

With $\Delta\omega$ the wave frequency discretization step. The wave frequency discretization is conveniently set equal to the frequencies of the force response functions calculated by the hydrodynamic code that give the first order wave forces on the vessel, Section 2-2-1. As the waves are linear the irregular sea realization can be formed by superposition of the linear components in direction and in frequency Figure 4-10:

$$\xi(x_1, x_2, x_3, ...) = \sum^{l} \sum^{k} \zeta(\omega_k, \alpha_l) \cos\left(\frac{\omega_k^2}{g} x_l - \omega_k t + \gamma_{\text{rnd}}(k, l) \right) \tag{4-73}$$

In case of 3 directions 45° apart, $x_{1..3}$ are the x, $x = y$ and the y directions. Where x_2 is discretized with steps of $\sqrt{2}/2$ to match the grid of x_1 and x_2. And $\gamma_{\text{rnd}}(k, l)$ is a random fase shift given to each wave component to create a unique realization (e.g. in Figure 4-10).

Wave forces The wave forces are found from the FTFs calculated jointly with the hydrocoefficients in the hydrodynamic code (WAMIT). These transfer functions give a force or moment from a sinusoidal wave input with a relative phase and a frequency dependent magnitude, but

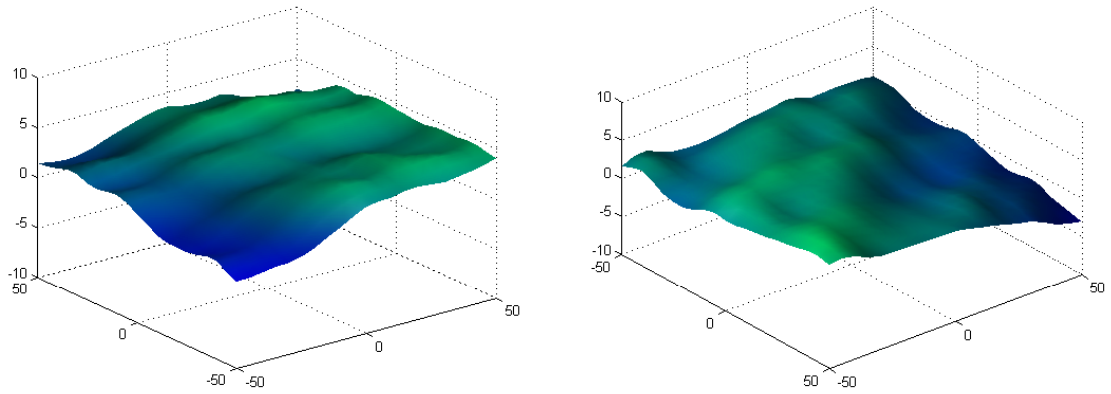

Figure 4-10: Two snapshots in time of a spatial wave field realization with the spectrum of Figure 4-9. Units are in meters and $\alpha = \{0°, 45°, 90°\}$. Note that the z axis is chosen smaller to visually accentuate the difference in surface elevation.

with the same sinusoidal shape and frequency as the wave component. In math a component (k, l) from Eq. (4-73) looks as:

$$\begin{aligned} &\zeta(\omega_k, \alpha_l) \cos\left(\frac{\omega_k^2}{g} x_l - \omega_k t + \gamma_{\text{rnd}}(k,l)\right) \overset{\text{FTF}}{\rightarrow} \left[\frac{\partial \tau}{\partial \zeta}, \gamma_W\right] \rightarrow \tau_W(\omega_k, \alpha_l, \{x, y, z, \phi, \theta, \psi\}) \\ &\tau_W(\omega_k, \alpha_l, \cdot) = \left(\frac{\partial \tau_W(\cdot)}{\partial \zeta_{\omega,\alpha}} \zeta(\omega_k, \alpha_l)\right) \cos\left(\frac{\omega_k^2}{g} x_l - \omega_k t + \gamma_{\text{rnd}}(k,l) + \gamma_W(\cdot)\right) \end{aligned} \tag{4-74}$$

An important value in a wave record is the highest wave that occurred. However as the wave generation is a stochastic principle and the modeled wave realization too, it is uncertain when and if this highest wave will occur. Therefore relative long simulation times are required. On the other hand a wave realization that is built by a finite sum of harmonic components has a finite periodicity. How long the wave simulation period will be depends on all the periods and the least common multiple of periods and the relative phase of the components. As the periods are non integer this is an undefined problem. Including infinite wave periods also results in infinite computation times so a trade off should be made. A wave record of wave 40 periods in 8 directions (major and minor wind directions) is generated. The periods are chosen with a non uniform grid to increase the least common multiple of periods of the record and therefore increase the period time of the wave field. [16]

4-5 MPC controller implementation

The dynamical system from Eq. (4-62) is implemented in MATLAB according to the simulation scheme shown in Figure 4-12, and the actual implementation in Figures B-1 to B-4. The coupled ship-platform dynamics form the central part of the scheme, drawn in the top left corner and consisting of the inertia and Coriolis terms. The different forcing components are calculated via the surrounding blocks. The gravity is transformed to the boat's reference

[16] With periods chosen for example as $\omega_{wave} = [0.1 : 0.05 : 2]$ the periodicity could be as fast as 2 minutes.

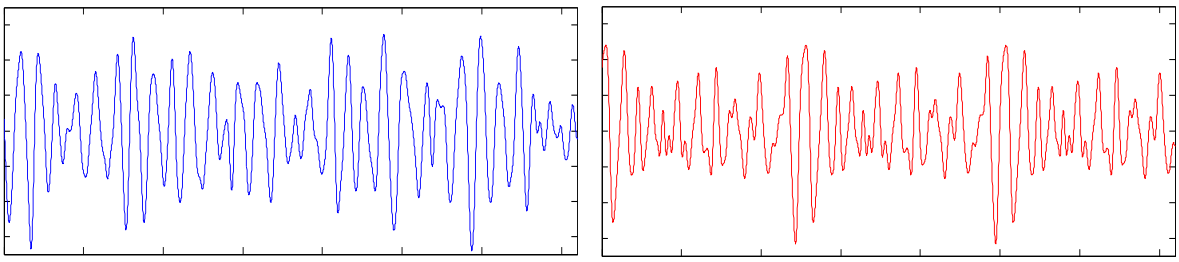

Figure 4-11: Wave realization of 40 wave components, 300 seconds. Identical first differences in wave period results in shorter wave record period than varying wave period differences. The right realization shows a period of about 120 seconds whereas the left realization that is constructed with an irregular spaced frequency grid does not show periodicity within the five minute range.

frame because the dynamics are described in a Lagrangian form. The wave realization from the spectral density with surface elevation Eq. (4-73) uses the FTFs to find the wave forces and the radiations terms are found from the state space approximation in radiation dummy state χ in Eq. (4-31). The forward kinematics find the platform position for given leg lengths which are generally simpler to measure.

The nonlinear dynamical model is used to generate smart control signals. The uncertain terms such as the wave loads are not considered for the control. And the radiation term is left out for execution speed at the cost of a plant-prediction model mismatch. We now have an approximated model that can be used to predict the future evolution of movements:

$$\begin{bmatrix} \dot{\bar{v}}^T & \dot{\bar{q}}^T \end{bmatrix}^T \approx F_{\text{pred}}(\bar{v}, \bar{q}, \bar{f}_{\text{legs}}) \tag{4-75}$$

Where $\dot{\bar{v}}$ is the boat fixed acceleration and $\dot{\bar{q}}$ the rate of change of pose coordinates. This prediction model F_{pred} is integrated by a fixed step Classical Runge-Kutta 4 scheme[17], with time steps of $\Delta t = .5s$ for $t = \Delta t ... i\Delta t ... N\Delta t$ where N is the prediction horizon. See Figure 4-13. From the predicted pose vector $\bar{q}_{pi}$ which is composed of the inertial ship coordinates predictions $\bar{\eta}_{pi}$ and the predicted relative coordinates of the platform $\bar{x}_{\beta pi}$, the inertial reference coordinate of the platform is calculated. This coordinate is denoted by $\overline{HRP}_{pi}$ indicating heave roll and pitch of the platform in the $\mathcal{I}$ frame.

$$\overline{HRP}_{pi} = \mathrm{H}^T \bar{\eta}_{pi} + \mathrm{H}^T \begin{bmatrix} R^T_{\mathcal{I}\to\mathcal{B}} & 0_{3\times3} \\ 0_{3\times3} & \mathrm{I}_{3\times3} \end{bmatrix} \mathrm{H}\bar{x}_{\beta pi} \tag{4-76}$$

Where selection matrix Eq. (4-34) is again applied. The error in inertial heave-roll-pitch, when a constant set point is chosen, is written as $\overline{HRP}_{\epsilon i} = \overline{HRP}_{pi} - \overline{HRP}_{set}$.

In order to handle constraints on input or predicted output signals we employ a linear penalty term on constraint violation, see Figure 4-14. The absolute violation is found by a violation function $\Phi(\bar{u}/\Delta\bar{u}/\bar{v}/\bar{q})$ and is multiplied by a factor to form a constraint violation term that can be added to the performance index we define next.

The forces that are applied to the legs intend to push the system towards an optimal trajectory with respect to some scalar performance criterion $\jmath$ (note: not matrix J). This scalar

[17] A four stage integration routine requiring four function evaluations per time step. Instability of a Forward Euler scheme with this time step showed the prediction required a higher order method.

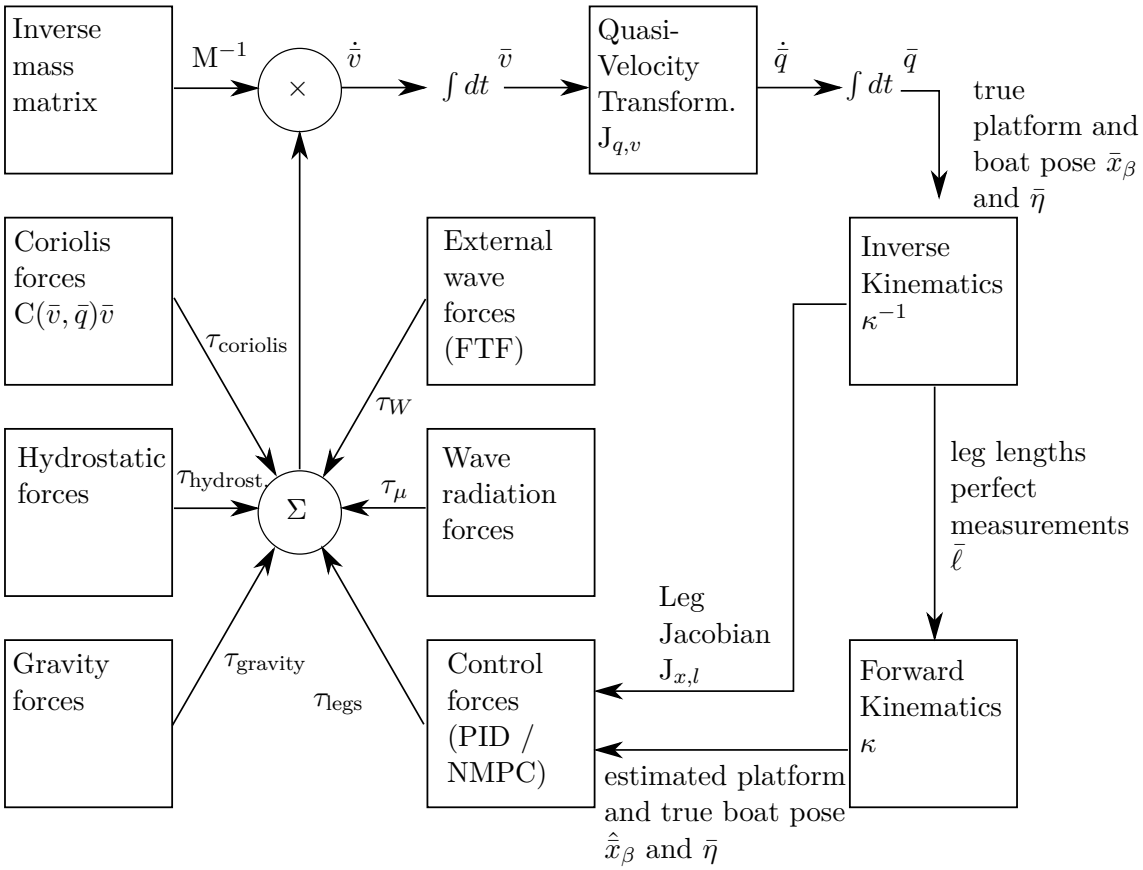

Figure 4-12: Global overview of the simulation components.

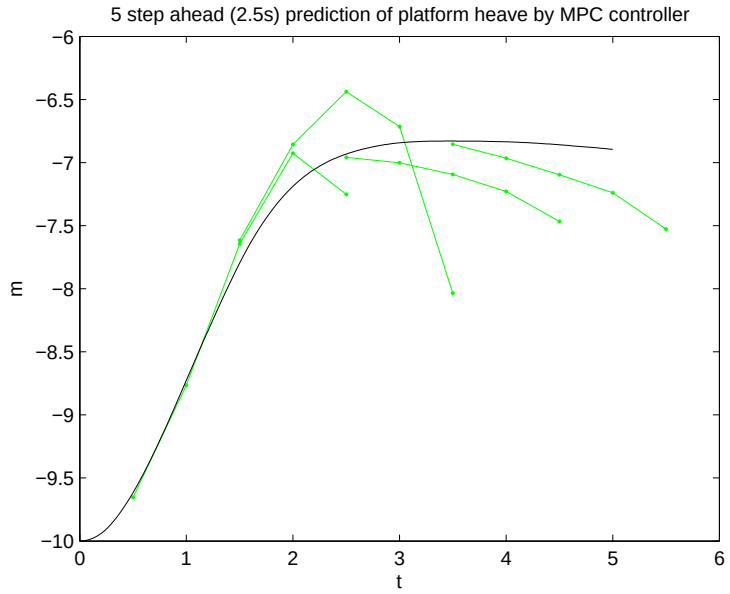

Figure 4-13: 2.5 second prediction of heave during the simulation. During the first part of the prediction the match is very usable. In the second part the motions are overestimated. This is probably due to the left out wave radiation effects that dampen the movements.

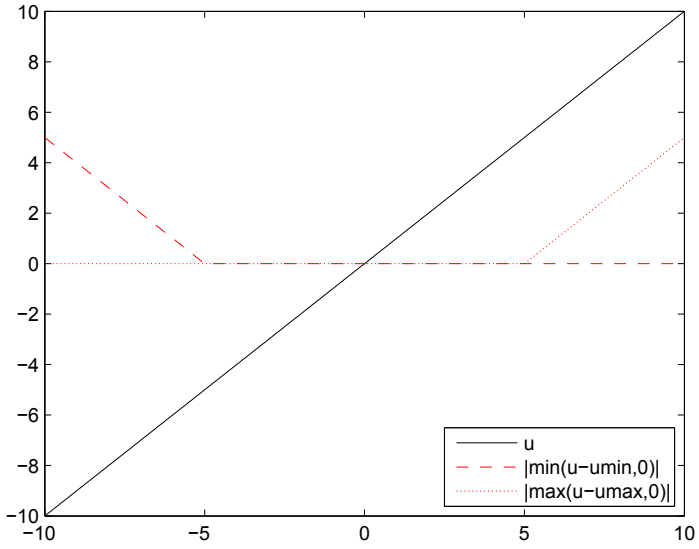

Figure 4-14: Input (and possibly output) constraint violations are implemented as soft constraints with a linear penalty term in the objective (performance index). This approach does not suffer from infeasibility of the optimization.

performance criterion can be chosen as in the examples in Section 2-1-1 and be calculated with knowledge about the present state or predicted state:

$$\jmath = \sum_{i=1}^{N} (\overline{HRP}_{pi})^T \mathrm{P} (\overline{HRP}_{pi}) + (\Delta \bar{f}_{\mathrm{legs}_{i-1}})^T \mathrm{Q} \Delta \bar{f}_{\mathrm{legs}_{i-1}} + \lambda_{\mathrm{constr.}} \Phi(u_i) \tag{4-77}$$

Where the control signal is the differenced leg force $\Delta u = \Delta \bar{f}_{\mathrm{legs}_i}$ as in the GPC (Eq. (2-1)) to allow a zero steady state signal.[18] The quadratic product matrices are $\mathrm{P} = \mathrm{diag}\begin{bmatrix} 1e2 & 4e4 & 4e4 \end{bmatrix}$ and $\mathrm{Q} = \mathrm{I}_{3\times 3}$ where the control signal is rescaled from Newton to MN. Now a nonlinear solver, a minimization algorithm, is employed to solve for the optimal control signals.

$$\min \jmath \rightarrow \texttt{minimizer}(@\texttt{Du J}(\texttt{v0}, \texttt{x0}, \texttt{q0}, \texttt{Du})) \tag{4-78}$$

The prediction horizon is a trade off between approaching the infinite horizon case with guaranteed stability and speed of execution of the scheme. The search space for the control signals grows exponentially with the number of steps and so does the complexity of the optimization. Several optimizations have been programmed and tested for the application: Nelder-Mead derivative free simplex search did not converge fast enough for real time capability. As the estimations are uncertain taking second derivatives to gain Hessian information seems to be risky. Therefore a Gauss-Newton search is chosen. This method approximates the gradient by the outer product of the gradients and requires only first order information. The algorithm converges from a bad guess in about 5 iterations, that each estimate a costly gradient, to a good sequence, Figure 4-15. Using the previous found solution as initial condition for

[18] Control signal $\bar{u}_i$ is found via initial $\bar{u}_0$ and the sequence of control differences $\bar{u}_k = \bar{u}_0 + \sum_{i=0}^{k} \Delta \bar{u}_i$

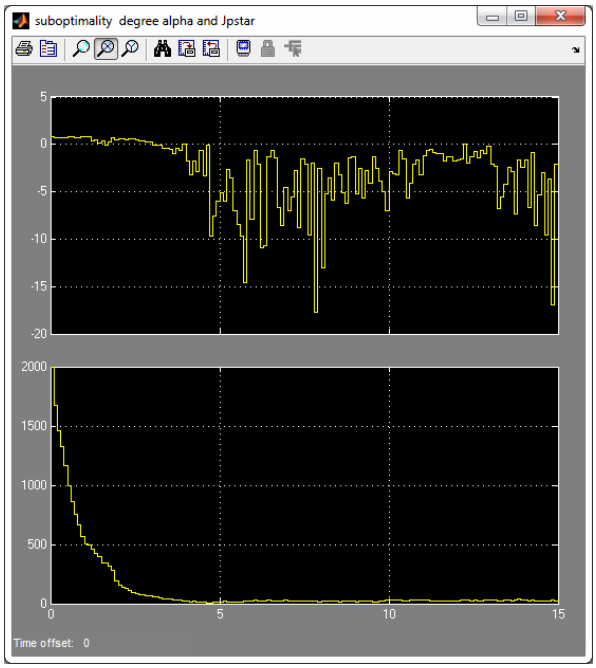

Figure 4-15: Scope showing suboptimality degree alpha (top) according to Eq. (2-7) and the value of the performance index (bottom) of the rolling optimization during a step response. The suboptimality value is positive in the initial disturbance rejection. When the reference signal is tracked the optimality estimate deteriorates. The guaranteed stability is lost, but it is observed that the optimality is regained after a small instance of time. A zone performance index or a local linear controller could help in this case.

the optimization (hot starting) reduces the iterations with a factor two. This, and efficient vectorized code generation, allows running the MPC controller at 50Hz.

4-6 Results

To put the performance of the NMPC controller in context a quasi-static PID controller is implemented. This controller calculates from the present pose of the platform and the desired pose, while assuming the ship remains steady, what the leg lengths should be. It then treats the difference of the present leg length and the desired leg length as error input for the controller. The error signal (a 3×1 vector) is split into a symmetric part and three differences from the mean. Both of these parts have separate quasi-static controllers to cope with the difference in force requirement for the heaving and for the rolling and pitching movements. The PID controller parameters are: $P_{\text{mean}} = 8e6$, $P_{\text{diff}} = 200e6$, $I_{\text{mean}} = 4e6$, $I_{\text{diff}} = 90e6$, $D_{\text{mean}} = 2e6$, $D_{\text{diff}} = 60e6$.

Figures 4-16 to 4-18 compare the performance of the two control strategies by starting a time domain simulation in the same[19] random sea with the high sea state $H_{1/3} = 4m$ and $T_1 = 6.5s$. Both in the same off initial reference position.

[19] exact same random seed.

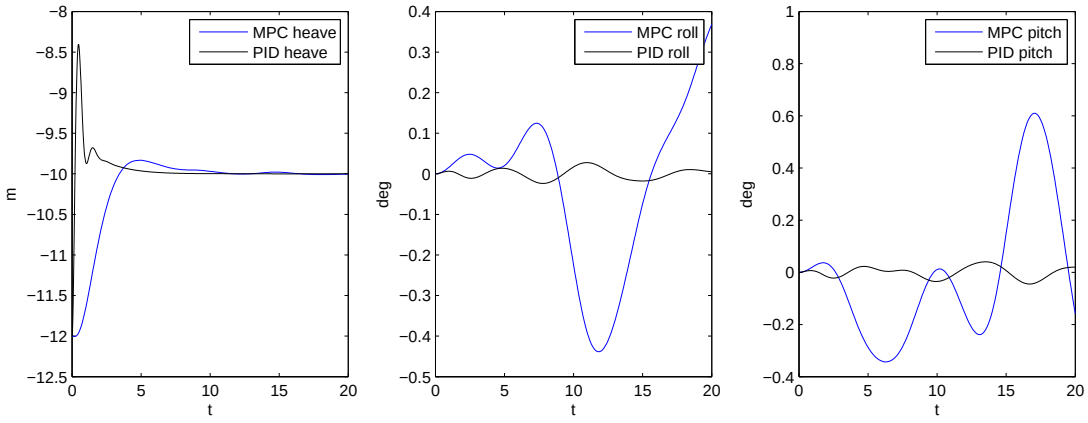

Figure 4-16: Comparison of the tracking capability of the MPC controller opposed to the PID controller.

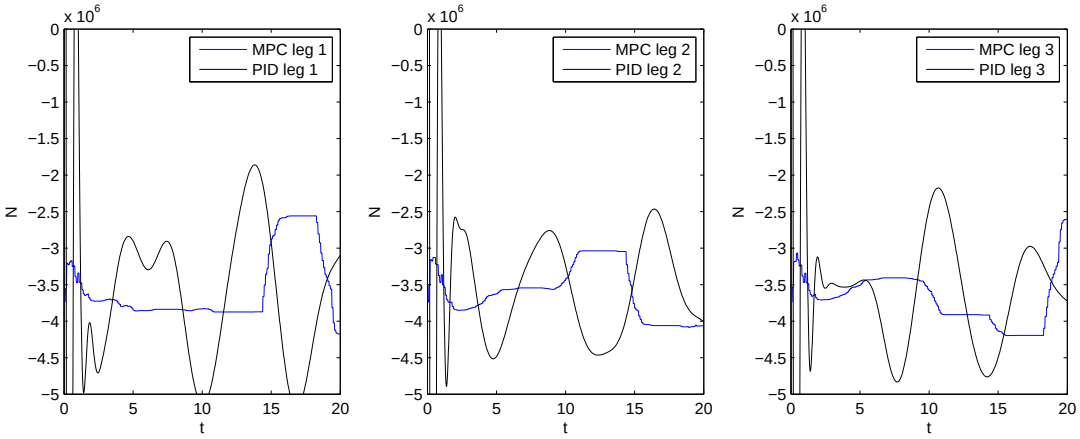

Figure 4-17: Comparison of the leg forces applied by the control to the platform.

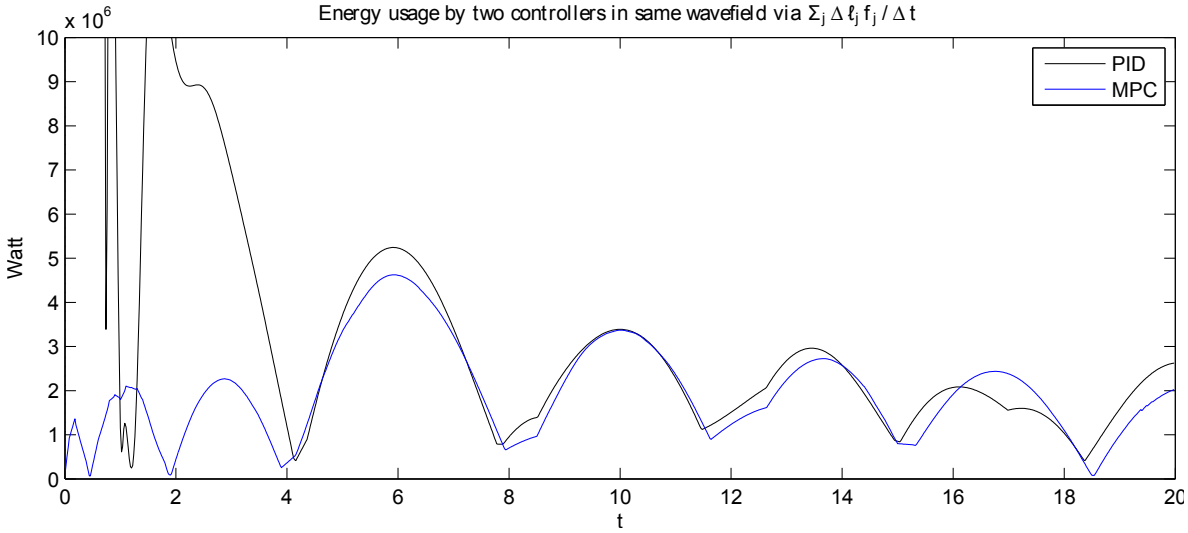

Figure 4-18: Comparison of the power required by the platform.

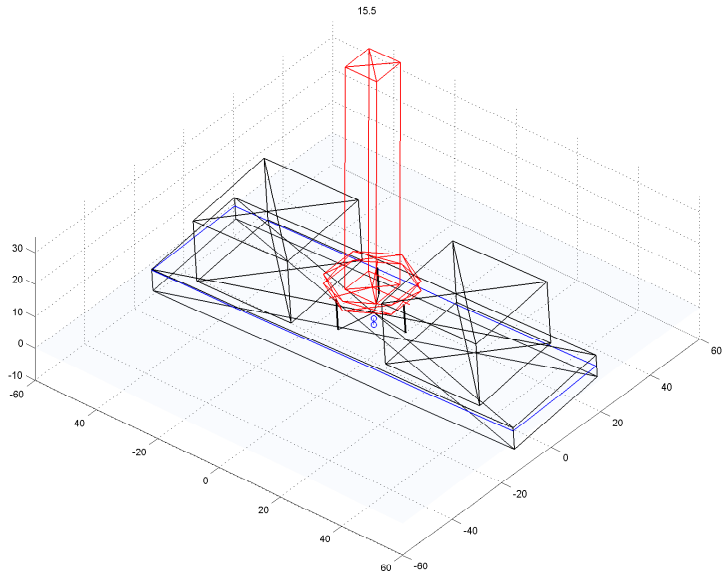

Figure 4-19: Visualisation with low computational demand. The boat is loaded with additional masses that increase the height of the center of gravity. The platform is loaded with a mass with a very high rotational inertia seen from the base. This case is very demanding for the platform and is therefore chosen for investigation.

Figure 4-16 shows that the PID controller is much stiffer as the rise times are faster and it shows oscillatory values. Keeping the tower straight at less than .1 degree is most likely not necessary and impossible. Figure 4-17 shows that the NMPC controller neatly complies to its constraints on maximum force, visible by the flat tops on the force lines and does not show of the chart extremes as the the PID controller does in the first few seconds. In Figure 4-18 the power usages are calculated via the leg force, the time and the leg extension rates. The MPC requires about half the power the PID controller demands in the disturbance rejection phase. Figures 4-19 to 4-21 show several visualizations of the 3D simulation with the platform in operation.

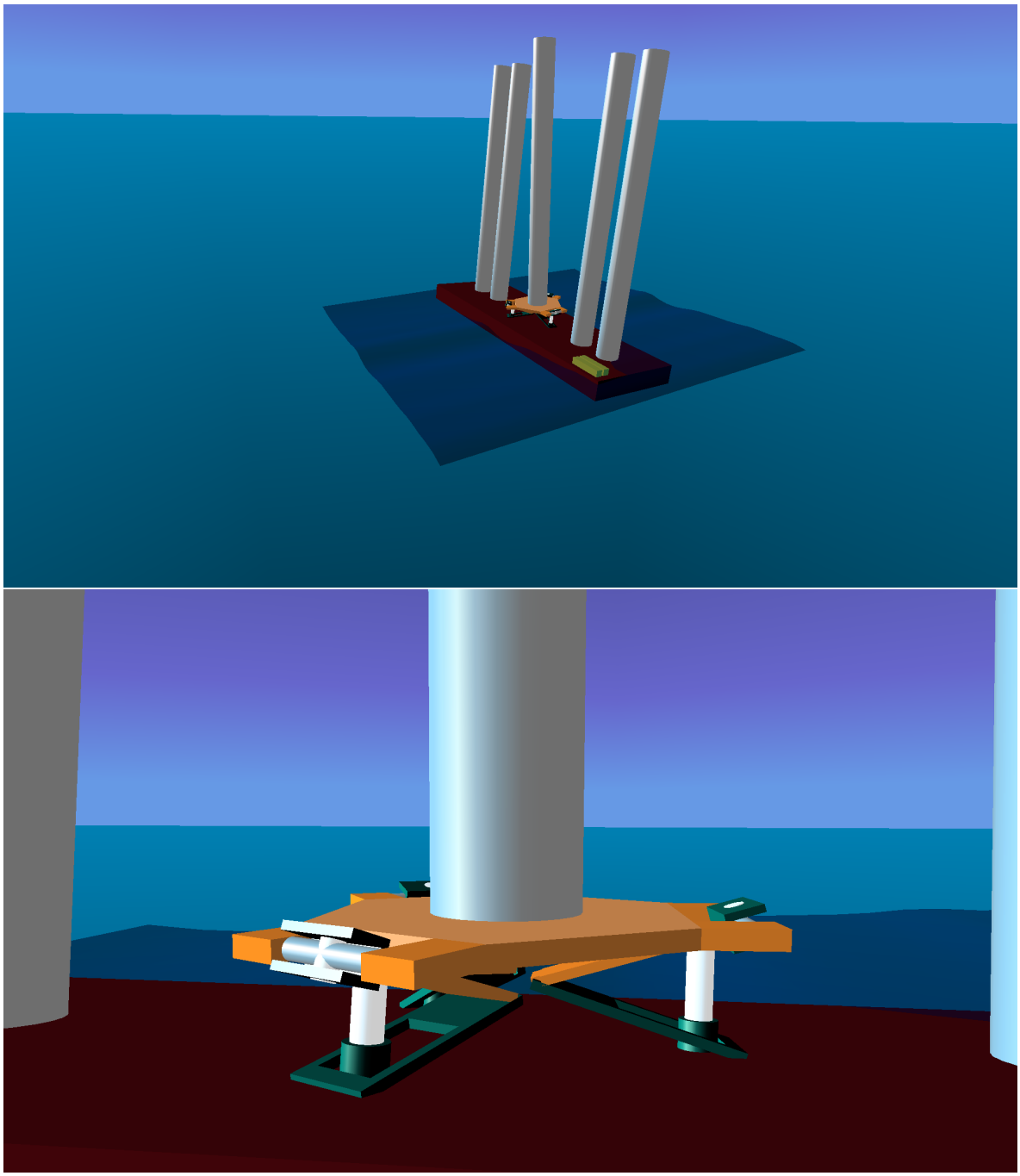

Figure 4-20: Visualization of the simulation 1. In the top rendering the load is kept at it's reference while the entire barge has a roll angle.

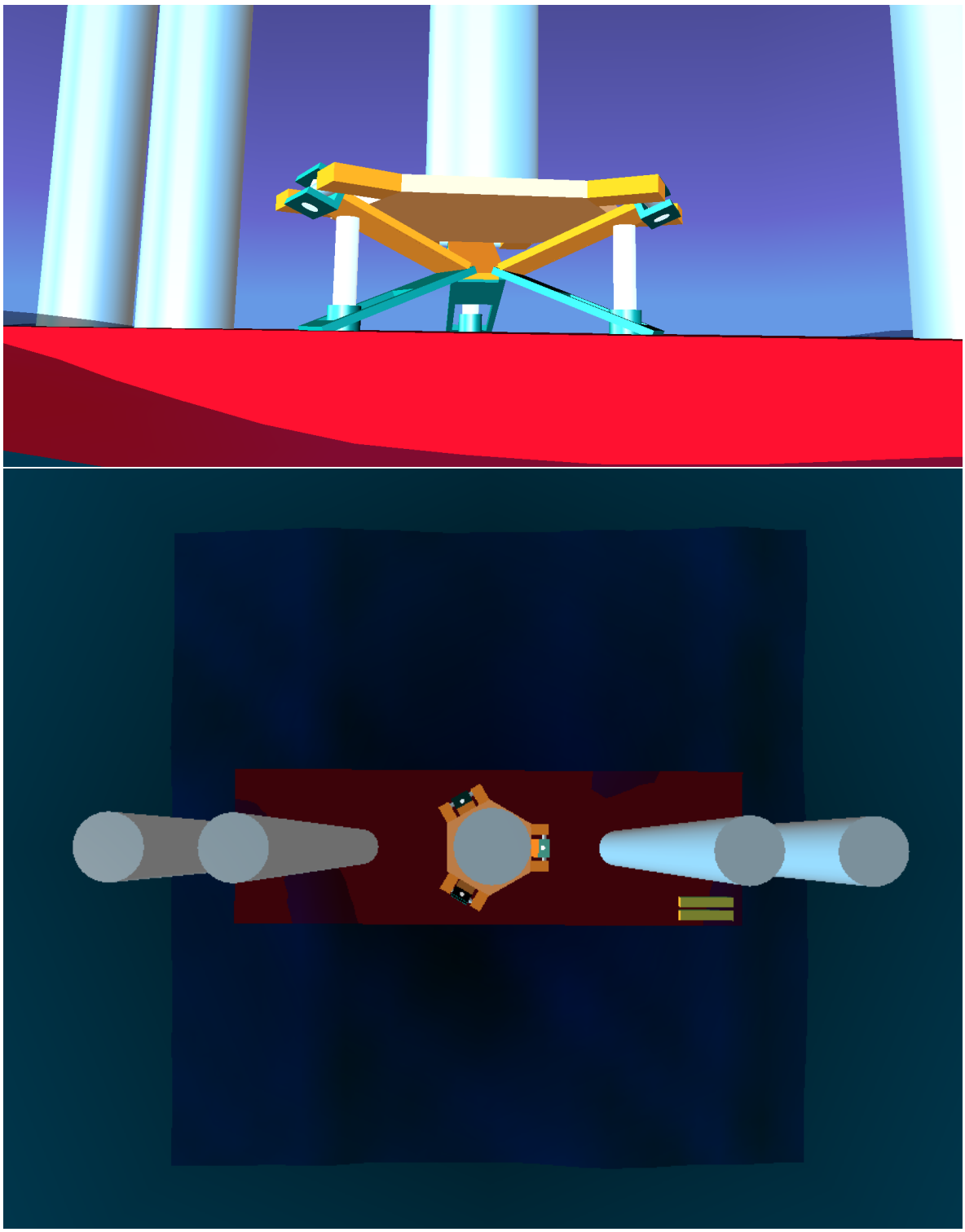

Figure 4-21: Visualization of the simulation 2. The top figure shows the connection of the hydraulics placed 20% downwards the top linkage. The yellow bars at the front stern side of the barge are container sized (12m) for visual reference.

Chapter 5

Conclusions

In this thesis the dynamic analysis and control of a ship motion compensation platform for high payloads is discussed. Investigation of data from a scale model experiment with a relatively heavy hexapod placed on its deck is performed. The roll instability visible in these experiments is identified by an estimated parametric linear model. The parameters of the linear model, e.g. the masses, hydrostatic stiffnesses and control stiffnesses, are varied one by one after their estimation to give quantitative statements about the estimated regions of instability. What is shown is that there always possible instable sets of parameters. The linear system can also always be stabilized via the linear controller if the proportional term may be set at high enough values. The risk of a double pole marginal stable pole, one from the controller, and one from the hydrostatics, remains and can destabilize the system.

To ensure that this roll instability does not take place a more sophisticated control system is proposed: Non Linear Model Predictive Control. To check feasibility and examine the capabilities of such a solution a simulation is performed. First a new kind of 3 degree of freedom parallel robotic platform is proposed. Then for the first time in literature the full 3D time domain dynamics of a parallel platform on a ship are derived and implemented in MATLAB. Finally two controllers are designed, one naive quasi-static and one true online nonlinear optimizing model predictive controller. This is the first time such a controller with real time capabilities is implemented on such a system. The model based controller outperforms the quasi-static in versatility, stability and energy efficiency.

Appendix A

The big list of assumptions

Here a list of major assumptions used to model the system in sufficient, but not exhaustive detail is provided. Probably this list is incomplete, but it acts as an indication for the boundaries of the applicability.

1. The platform links are rigid, massless bodies. [28]
2. There is no friction in the in the actuators and the platform can be force controlled.
3. The platform is placed in on the symmetry plane of the ship.
4. The two bodies are rigid.
5. The exact rigid body inertia matrices, dimensions etc. are known.
6. The barge is in "Seekeeping" state (non traversing, $< 1.5m/s$) [35]
7. The hull shape is symmetric, so are the hydrocoefficients ([39] Chapter 7 pp.8-9)
8. The water is assumed ideal, thus potential theory can be applied. ([49])
9. The hydrodynamics are linear (nonlinear terms can enter the equations on the right hand side, [39] and [44]).
10. The hydrostatics are decoupled in the 6 DoFs / or are linear in the position vector.
11. The waves are simple, a harmonics based sea is assumed. Irregular sea is generated by superposition with random phase.
12. The wave force is calculated from an undisturbed wave (Froude-Krylov) with a Hydrodynamic code
13. The unit operates in deep water so there is no bottom effect. ([39])
14. There is no wind load or wind and waves can be superimposed ([35])

15. In plane movements (surge/sway/yaw) are handled externally by mooring or DP.
16. The hydrocoefficients from the hydrodynamic frequency domain code take care of the effects from the mooring and the DP [73].
17. The earth is flat with respect to the limited operating range.
18. The NED frame is inertial.

Appendix B

Simulink block schemes

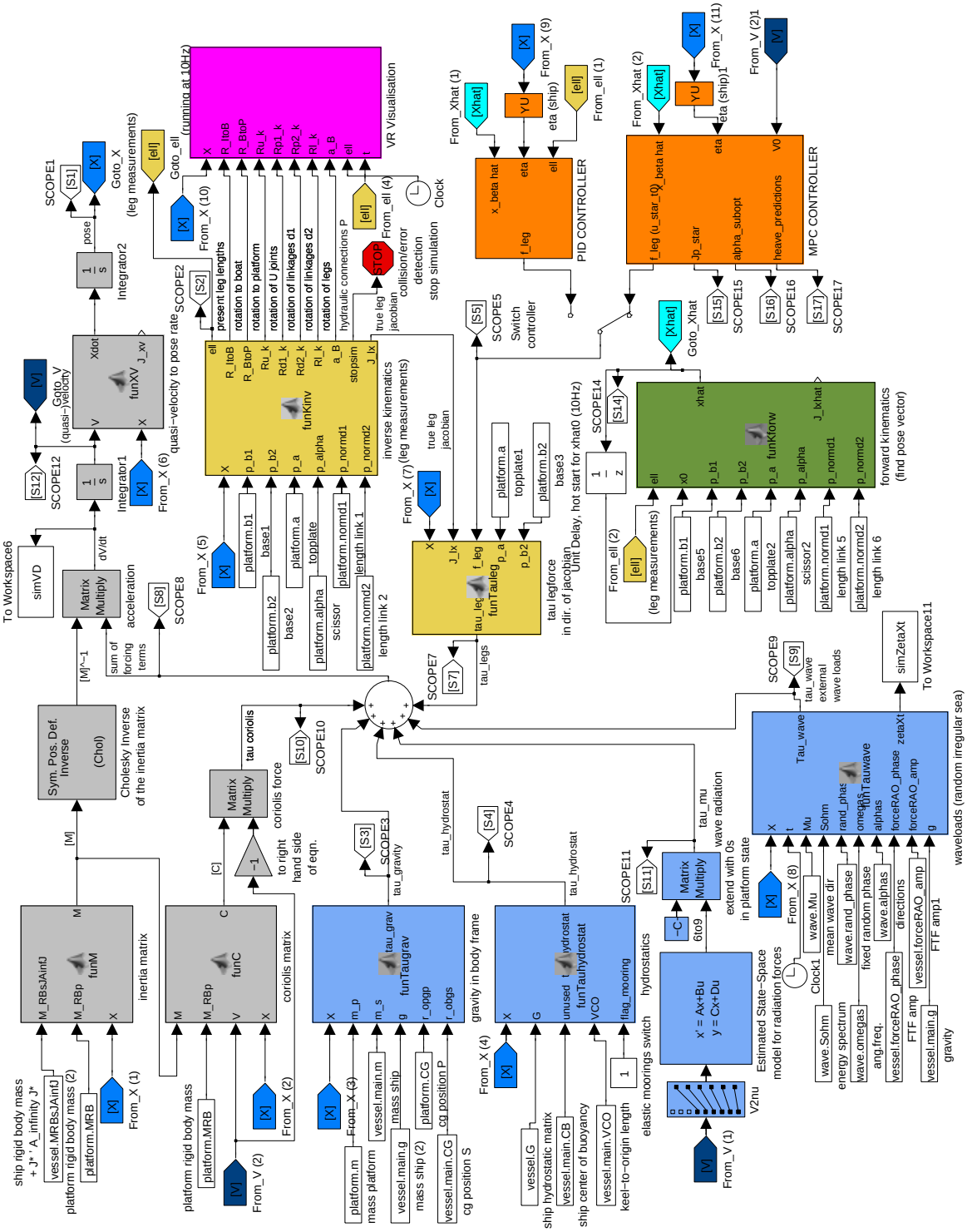

Figure B-1: The simulation implementation block scheme.

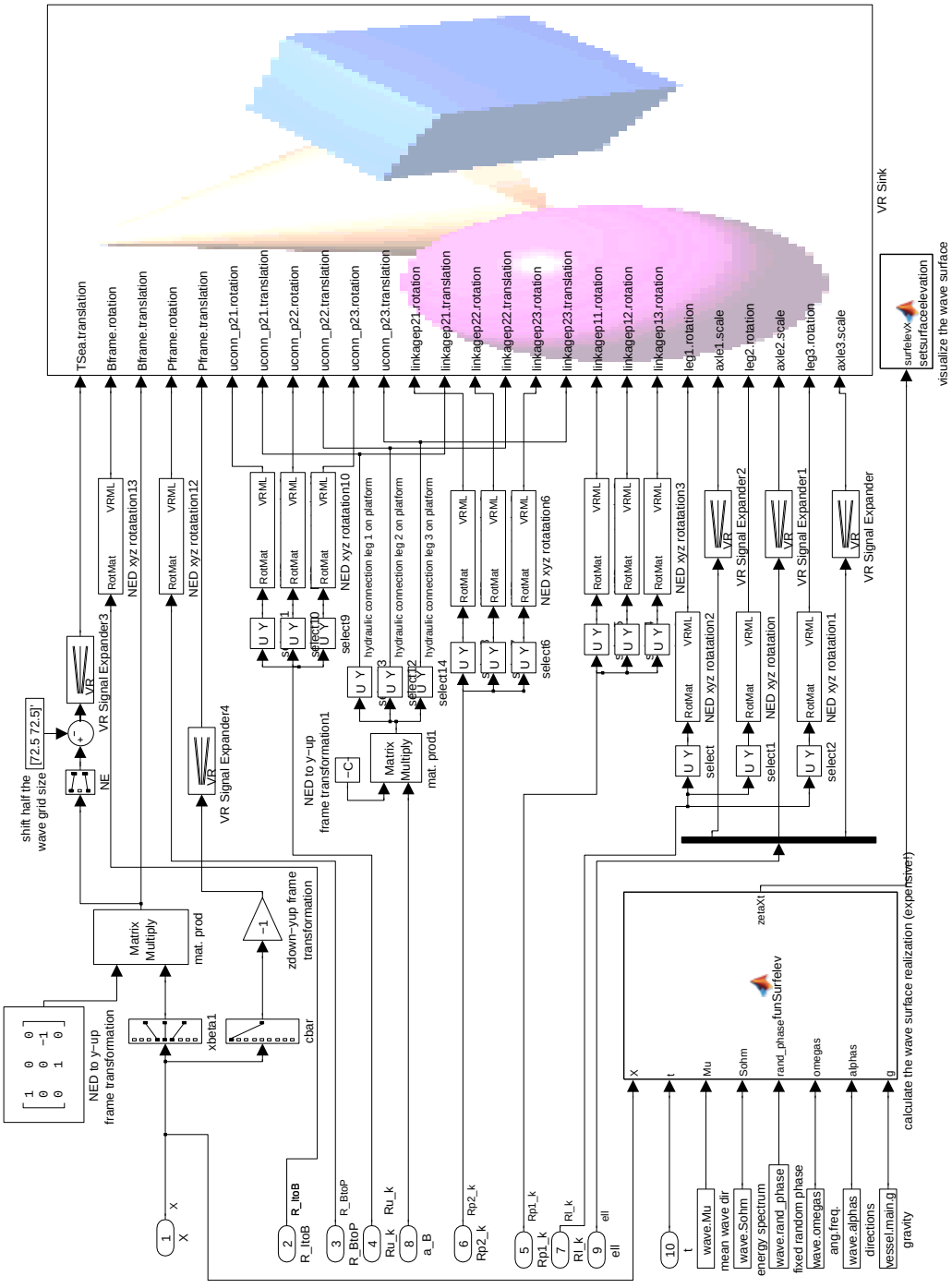

Figure B-2: The simulation implementation block scheme. Virtual Reality part.

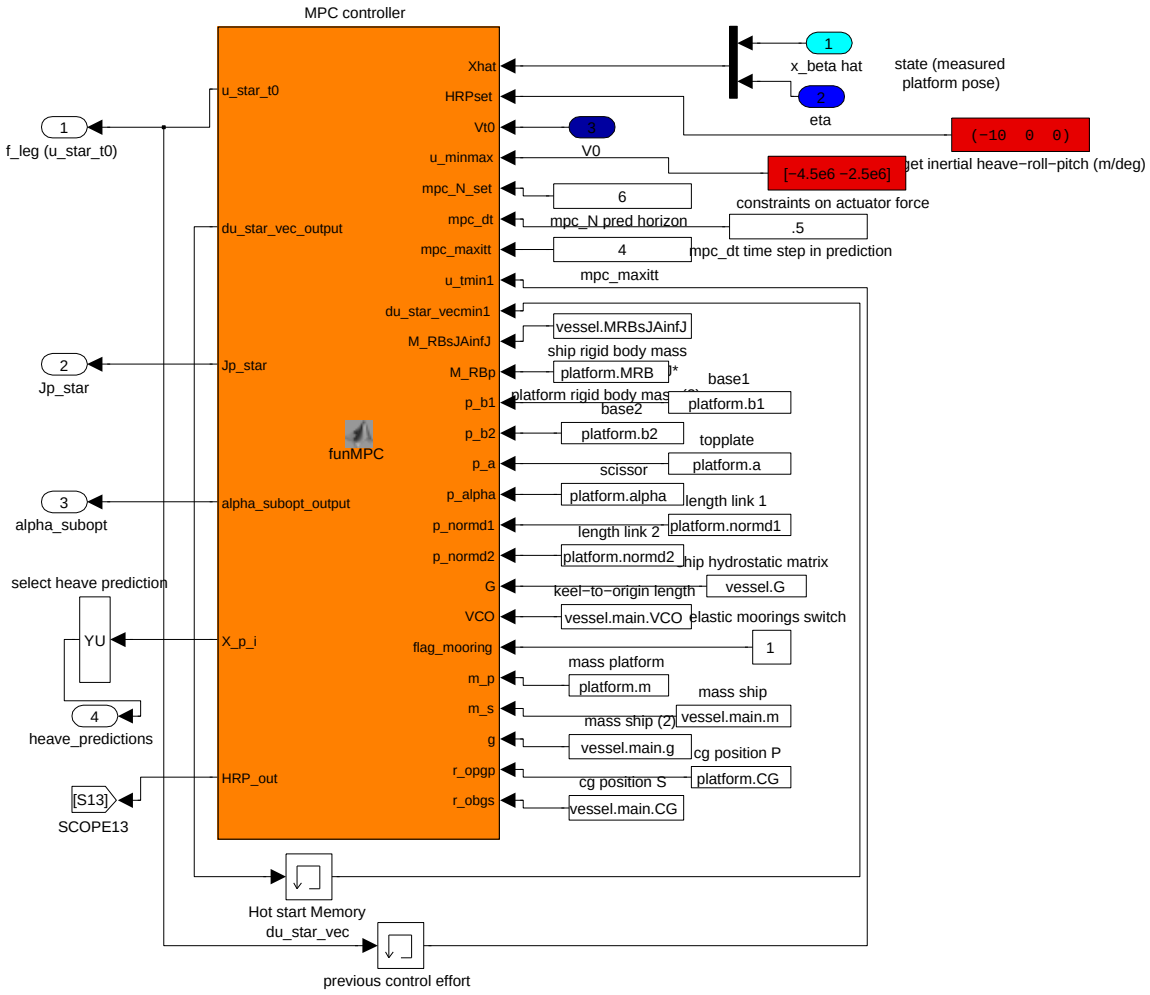

Figure B-3: The simulation implementation block scheme. MPC part.

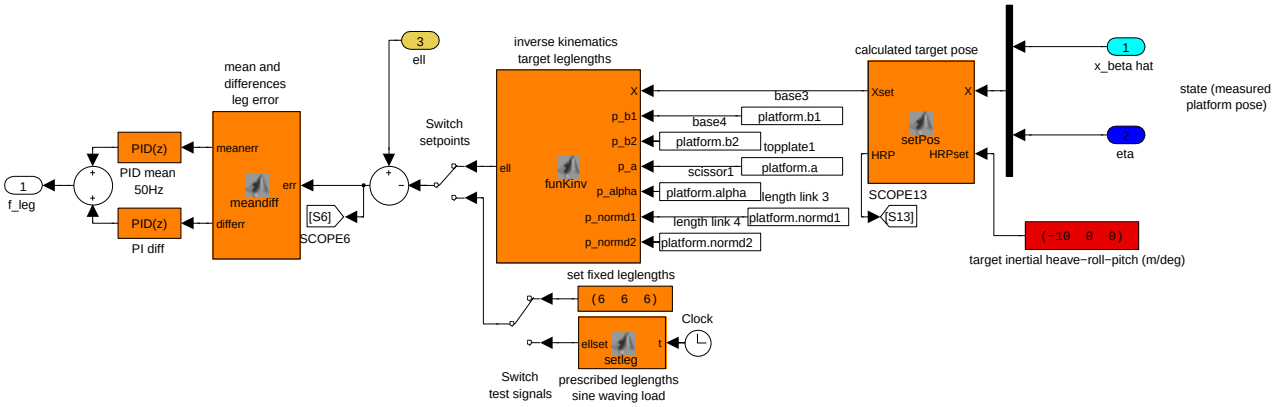

Figure B-4: The simulation implementation block scheme. PID part.

Appendix C

Matlab Listings

C-1 maxlike.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%              function  maxlike.m                  %%%%%%
%%%%       part of MSc thesis WA de Zeeuw 2011-2012    %%%%%%
%%%  estimates the linear system by maximum likelihood   %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function X=maxlike
% load track
% Y = [phi_p, phi_b];
load track2
Y = [phi_p_centered, phi_b_centered];
tset = 0:0.1:30.7;
options = optimset('PlotFcns',{@optimplotx,@optimplotfval},'MaxFunEvals',20000,'MaxIter',10000);
X = QN_BFGS(@(X) -l(X,tset,Y),X0');
end

function plotfit_phys(X,tset,Y,pp)
h_fig = figure(2);
X = X * 5.*[1 1 1 1 1 25 25 25 1 1 1 1/5]';
X(5:12) = abs(X(5:12));
x01 = X(1);x02 = X(2);x03 = X(3);x04 = X(4);
m1 = X(5);m2 = X(6);kP = X(7);kS = X(8);
kH = X(9);cD = X(10);cS = X(11);cH = X(12);
sigm2 = X(13)^2
disp(['m1 = ' num2str(m1) ', m2 = ' num2str(m2)])
disp(['kP = ' num2str(kP) ', kS = ' num2str(kS) ', kH = ' num2str(kH)])
disp(['cD = ' num2str(cD) ', cS = ' num2str(cS) ', cH = ' num2str(cH)])

% As = [[ c11,c12;c21,c22]*25 [k11,k12;k21,k22]*25;eye(2) zeros(2)];
As = [[ -(cD+cS)/m1, cS/m1;(cD+cS)/m2 -(cH+cS)/m2 ] [-(kP+kS)/m1, kS/m1; (kP+kS)/m2 -(kS+kH)/m2 ]; eye(2) zeros(2)]
x0 = [x01,x02,x03,x04]'
for i=1:numel(tset)
    t=tset(i);
    t=tset(i)-tset(1);
    YH(i,:) = (expm(As*t)*x0)';
end
plot(tset,Y(:,1),'r.',tset,Y(:,2),'b.',...
    tset,YH(:,3),'r-',tset,YH(:,4),'b-')%,...
% %      [tset(end/2) tset(end/2)],[-10 10],'k')
% plot(tset,YH(:,3),'r-',tset,YH(:,4),'b-')
K = round([-(kP+kS)/m1, kS/m1, (kP+kS)/m2 -(kS+kH)/m2 ]*100)/100;
C = round([ -(cD+cS)/m1, cS/m1,(cD+cS)/m2 -(cH+cS)/m2 ]*100)/100;
X00 = round([x01,x02,x03,x04]*100)/100;
h_title = title({['Linear state space fit: log-likelihood = ' num2str(pp,'%3.2f')],...
```

```
44      ['$[k_{11}$ $k_{12}$ $k_{21}$ $k_{22}] = $[' num2str(K,'% 2.2f ') '] $[c_{11}$ $c_{12}$ $c_{21}$
            $c_{22}] = $[' num2str(C,'% 02.2f ') ']']});
45  set(h_title,'interpreter','latex')
46  h_l1 = xlabel(['time (s), initial condition: $[\dot \phi_p$ $\dot \phi_b$ $\phi_p$ $\phi_b] = $['
        num2str(X00,'% 2.2f ') ']']);
47  set(h_l1,'interpreter','latex')
48  h_l2 = ylabel('rollangle (degree)');
49  set(h_l2,'interpreter','latex')
50  ylim([-10 10])
51  h_leg = legend('boat','platform','location','NW');
52  set(h_leg,'interpreter','latex')
53
54  figure(3)
55  % plot(tset,Y(:,1)-YH(:,3),'b',tset,Y(:,2)-YH(:,4),'r')
56  % a=50000;x = randn(a,1);n=25;hist(x,n);rangex = max(x(:)) - min(x(:));binwidth = rangex/n;xn =
        [-3:.01:3];sig1 = std(x); mu1 = mean(x);yn1 = (a*binwidth)*1/sqrt(2*pi*sig1^2)*exp(-.5*(xn-mu1)
        .^2/sig1^2); hold on; plot(xn,yn1,'b');hold off
57  xn = [-2:.01:2];
58  res_1 = Y(:,1)-YH(:,3);
59  res_2 = Y(:,2)-YH(:,4);
60  nbins = 25;
61  range1 = max(res_1) - min(res_1);binwidth1 = range1/nbins;
62  range2 = max(res_2) - min(res_2);binwidth2 = range2/nbins;
63  sig1 = std(res_1); mu1 = mean(res_1);
64  sig2 = std(res_2); mu2 = mean(res_2);
65  N1 = length(res_1);
66  kurt1 = 1/N1*sum((res_1-mu1).^4)/(1/N1*sum((res_1-mu1).^2))^2;
67  skew1 = 1/N1*sum((res_1-mu1).^3)/(sqrt(1/N1*sum((res_1-mu1).^2)))^3;
68  JB1 = N1/6*(skew1^2+(kurt1-3)^2/4);
69  N2 = length(res_2);
70  kurt2 = 1/N2*sum((res_2-mu2).^4)/(1/N2*sum((res_2-mu2).^2))^2;
71  skew2 = 1/N2*sum((res_2-mu2).^3)/(sqrt(1/N2*sum((res_2-mu2).^2)))^3;
72  JB2 = N2/6*(skew2^2+(kurt2-3)^2/4);
73  truncate=1;
74  if truncate
75      for j = 0:5
76          res_1_trunc = sort(res_1);res_1_trunc = res_1_trunc(1+j:end-j);
77          res_2_trunc =sort(res_2);res_2_trunc = res_2_trunc(1+j:end-j);
78          N1 = length(res_1_trunc);
79          kurt1 = 1/N1*sum((res_1_trunc-mu1).^4)/(1/N1*sum((res_1_trunc-mu1).^2))^2;
80          skew1 = 1/N1*sum((res_1_trunc-mu1).^3)/(sqrt(1/N1*sum((res_1_trunc-mu1).^2)))^3;
81          JB1_trunc(1+j) = N1/6*(skew1^2+(kurt1-3)^2/4);
82          N2 = length(res_2_trunc);
83          kurt2 = 1/N2*sum((res_2_trunc-mu2).^4)/(1/N2*sum((res_2_trunc-mu2).^2))^2;
84          skew2 = 1/N2*sum((res_2_trunc-mu2).^3)/(sqrt(1/N2*sum((res_2_trunc-mu2).^2)))^3;
85          JB2_trunc(1+j) = N2/6*(skew2^2+(kurt2-3)^2/4);
86      end
87      [JB1_trunc,ntrunc1] = min(JB1_trunc);
88      ntrunc1 = ntrunc1-1;
89      [JB2_trunc,ntrunc2] = min(JB2_trunc);
90      ntrunc2 = ntrunc2-1;
91  end
92  yn1 = (binwidth1*N1)/sqrt(2*pi*sig1^2)*exp(-.5*(xn-mu1).^2/sig1^2);
93  yn2 = (binwidth2*N2)/sqrt(2*pi*sig2^2)*exp(-.5*(xn-mu2).^2/sig2^2);
94  subplot(2,1,1)
95  hold off
96  hist(res_1,nbins)
97  hold on
98  plot(xn,yn1,'r')
99  title({['Residuals of platform, and Gaussian fit, sigma = ' num2str(sig1,'%2.3f') ', mu = ' num2str(
        mu1,'%2.3f')],['K = ' num2str(kurt1) ', S = ' num2str(skew1) ', JB = ' num2str(JB1) ', JB
        truncated ends:-' num2str(ntrunc1) ' = ' num2str(JB1_trunc)]})
100 subplot(2,1,2)
101 hold off
102 hist(res_2,nbins)
103 hold on
104 plot(xn,yn2,'b')
105 title({['Residuals of boat, and Gaussian fit, sigma = ' num2str(sig2,'%2.3f') ', mu = ' num2str(mu2,
        '%2.3f')],['K = ' num2str(kurt2) ', S = ' num2str(skew2) ', JB = ' num2str(JB2) ', JB truncated
        ends:-' num2str(ntrunc2) ' = ' num2str(JB2_trunc)]})
106 % set(h_fig,'paperunits','centimeters') %pdf whitespace removal
107 % set(h_fig,'papersize',[10,7])
108 % set(h_fig,'paperposition',[0,0,10,7])
109 % filename = ['fit_unconstrained.pdf'];
110 % print('-dpdf',filename) %export
111 figure(5)
112 plot(tset,res_1,'r',tset,res_2,'b')
```

```
113 end
114
115
116 function [X,objFuncValue,H]=QN_BFGS(fun,X0,noline)
117 h=12;
118 x = X0;
119 df = numdiff(fun,x,(1*10^-h));
120 objFuncValue = fun(x);
121 H=eye(length(X0)); % initial inverse hessian approximation
122 iter = 1;
123 conv = 0;
124 alpha_o=1;
125 % hold on
126 % plot(iter,objFuncValue,'db');
127 while conv~=1 && iter<600
128     % obtain search direction
129
130     dir = -(H*df); % newton step
131     dir = dir/norm(dir);
132     % linesearch to to find acceptable stepsize alpha
133     if (noline-iter)<0
134         alpha_o = fminsearch(@(alpha) fun(x+alpha*dir),1);
135         alpha = max(0.001,min(alpha_o,1.1));
136         disp(num2str([iter objFuncValue alpha_o]))
137     else
138         alpha = 0.000001;
139         disp(num2str([iter objFuncValue alpha]))
140     end
141
142     % update xk -> xk+1
143     dx = alpha*dir;
144     x_new = x + dx;
145
146     % calculate new gradient
147     df_new = numdiff(fun,x_new,(1*10^-h));
148     yk = df_new - df; %gradient change
149     while (sum(isnan(yk))>0) ||  (sum(yk==0)>0);
150         h=h-1;
151         df_new = numdiff(fun,x_new,(1*10^-h));
152         yk = df_new - df; %gradient change
153     end
154     %update inverse hessian
155     %     H_new = (eye(13) - yk*dx'/(yk'*dx))' ...
156     %          * H * (eye(13) - yk*dx'/(yk'*dx)) + ...
157     %          dx*dx'/(yk'*dx); %BFGS update
158     H_new = H + (dx'*yk + yk'*H*yk)*(dx*dx')/(dx'*yk)^2 - ...
159         (H*yk*dx'+dx*yk'*H)/(dx'*yk);%BFGS update
160     %  HESSIANNEW = H + (q*q')/(q'*p) - ((H*p)*(H*p)')/(p'*H*p);
161
162     % overwrite old values
163     df = df_new;
164     x = x_new;
165     prev_objFuncValue = objFuncValue;
166     objFuncValue = fun(x);
167     if isnan(objFuncValue); disp('objective is NaN'); break; end
168     iter = iter+1;
169     % plot(iter,objFuncValue,'db');
170     % drawnow
171     if  ((prev_objFuncValue-objFuncValue)<1e-8) && (norm(df)<1)
172         conv=1
173         normgradient = norm(df)
174         figure(4)
175         imagesc(H); colorbar
176     end
177     H = H_new;
178     if (iter/100 - round(iter/100))==0.0
179         x=x
180         format long
181         H=H
182         format short
183     end
184 end
185 X=x; %output
186 H
187 end
188
```

```
function df = numdiff(fun,X,h)
if nargin < 3
    h  = 1e-8;
end
df = NaN*X;
oldObjFuncValue = fun(X);
for i = 1:numel(X)
    X_new    = X;
    X_new(i) = X_new(i) + h;
    newObjFuncValue = fun(X_new);
    df(i) = (newObjFuncValue-oldObjFuncValue)/h;
end
end
```

C-2 coordinates.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% 19 apr 2012 WA de Zeeuw - graduation at GustoMSC
%%% analysis of roll resonance movie ampelmann scale model
%%% via tracking of four points with a specific colour in 308 frames
%%% of the movie spaced 0.1 seconds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all
colorimg = imread('foo-001color.png');
color = squeeze(colorimg(1,1,:));
fig1=figure(1);clf
set(1,'outerposition',[ 516   571   835   605])
i = 1;
for i=1:308
%plot found corners
name = ['foo-' num2str(i,'%03u') '.png'];
foo = imread(name);
F = find(foo(:,:,1)==color(1) & foo(:,:,2)==color(2) & foo(:,:,3)==color(3)); %automatically sorted
    on x
SZ = size(foo(:,:,1));
[row col]=ind2sub(SZ,F);
% figure(1);
subplot(1,2,1)
image(foo)
hold on;plot(col,row,'oc');axis square
title(['Roll resonance movie Ampelmann, t = ' num2str(i*0.1,'%02.1f')])
xlabel('(property of D. Cerda Salzmann)')

%save track
Xcoordinates(i,1:4) = col';
Ycoordinates(i,1:4) = row';
% figure(2);clf
subplot(1,2,2) %plot of angles
% plot(Xcoordinates(:,1),Ycoordinates(:,1),'r--',...
%      Xcoordinates(:,2),Ycoordinates(:,2),'b--',...
%      Xcoordinates(:,3),Ycoordinates(:,3),'b',...
%      Xcoordinates(:,4),Ycoordinates(:,4),'r')
% plot([1:i].*0.1,Ycoordinates(:,1)-200,'r--',...
%      [1:i].*0.1,Ycoordinates(:,4)-200,'r',...
%      [1:i].*0.1,Ycoordinates(:,3)-50,'b',...
%      [1:i].*0.1,Ycoordinates(:,2)-50,'b--')
ratio_b =  diff(Ycoordinates(:,[1 4])')'./diff(Xcoordinates(:,[1 4])')';
ratio_p =  diff(Ycoordinates(:,[2 3])')'./diff(Xcoordinates(:,[2 3])')';
phi_b = 180/pi*atan(ratio_b)-6;
phi_p = 180/pi*atan(ratio_p)-6;
plot([1:i].*0.1,phi_b,'r.-',...
     [1:i].*0.1,phi_p,'b.-')
xmax = max(10,i*0.1);
xmin = max(0,xmax-10);
 xlim([xmin xmax])
 ylim([-8 8])
xlabel('time (s) (analysis: W.A. de Zeeuw)');ylabel('roll angle (deg)');
legend('boat','platform','location','NW')
title('Approx. roll angle via atan(dy/dx)')
set(fig1, 'color', 'white')

%movie record
winsize = get(fig1,'Position');
```

```
winsize(1:2) = [0  0];
numframes = 308;
if i==1 %first frame
A=moviein(numframes,fig1,winsize);
set(fig1,'NextPlot','replacechildren')
end
A(:,i)=getframe(fig1,winsize);

% pause(0.3)
end
mpgwrite(A,jet,'Resonance_ampelmann_roll_angle.mpg');

figure(2)
plot([1:i].*0.1,phi_b,'r.-',...
     [1:i].*0.1,phi_p,'b.-')
xlabel('time (s) (experiment: D. Cerda Salzmann, analysis: W.A. de Zeeuw)');ylabel('roll angle (deg)
    ');
legend('boat','platform','location','NW')
title('Approx. roll angle via atan(dy/dx)')
```

C-3 start3dsim.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion start3dsim.m
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start up script for 3d simulation of coupled boat and platform
% Boat hydrodynamics are based on WAMIT data with retardation in cummins
% equation approximated by state space estimations. A rectangular barge is
% used for hydrocoefficient generation.
% The platform is a 3DOF sarrus leg type heave-roll-pitch platform with a
% force-displacement hinge mechanism
clc; clear all
disp('...,,,---{{{ 3D simulation of coupled boat and platform }}}---,,,...')
disp('...              Part of MSc Thesis Wouter de Zeeuw ')
disp('...                 TU Delft - GustoMSC 2011-2012')
disp(' ')


disp('Starting simulink model structure')
shipplatform %.mdl file
disp('Rerrunning loaddata.m to load simulink workspace data to matlab workspace')
wave.loadseadflag = 1; %load random seed data from .mat file (fix for the second run)
loaddata %rerun the loaddata script
```

C-4 RBinertias.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% m-file RBinertias.m
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculates the rigid body inertia inertia matrices for the vessel and
% platform around the platform.local body axi. The platform inertia is composed out
% of a base and several building bplatform.locks (cilinders and boxes)
%
% assumes presence of vessel and platform structures with mass values:
% platform.m ande vessel.main.m
% rigid body (CoG) inertia matrix vessel.MRBcg and cg platform.loc. vessel.main.CG
%
%

%--- Vessel inertia around \rf{B} axis
S = skew(vessel.main.CG);% skew symmetric matrix of r_obog vector in B frame
vessel.MRB = [vessel.MRBcg(1:3,1:3) -vessel.main.m*S;
              vessel.main.m*S     vessel.MRBcg(4:6,4:6)-vessel.main.m*S^2];
% Infinity added mass in CO added to MRBship
% force infinity added mass to be symmetric!!
vessel.FDI.Ainf_hatJstar_sym=.5*(vessel.FDI.Ainf_hatJstar+vessel.FDI.Ainf_hatJstar');
vessel.MRBsJAinfJ= vessel.MRB + vessel.FDI.Ainf_hatJstar_sym;
clear S
```

```
25
26  %--- Platform inertia matrix
27  m_p = platform.m;
28  % #1: box with weightfraction f_box at platform.location r_box from platform base
29  platform.dim_box = [5 4 4]; %platform.dimensions x y z
30  platform.loc_box = [0 0 -100]; %platform.location of the CoG of the unit
31  f_box = 0.35;%mass fraction
32  m_box = f_box*m_p; %mass
33  rho_box = m_box/prod(platform.dim_box);
34  Ixx_box = 1/12*m_box*(platform.dim_box(2)^2+platform.dim_box(3)^2);
35  Iyy_box = 1/12*m_box*(platform.dim_box(1)^2+platform.dim_box(3)^2);
36  Izz_box = 1/12*m_box*(platform.dim_box(1)^2+platform.dim_box(2)^2);
37  MRB_boxcg = diag([m_box m_box m_box Ixx_box Iyy_box Izz_box]);%cog inertia
38  S = skew(platform.loc_box); %platform P coordinate inertia
39  MRB_box = [MRB_boxcg(1:3,1:3) -m_box*S;
40         m_box*S     MRB_boxcg(4:6,4:6)-m_box*S^2];
41
42  % #2: standing cylinder
43  platform.dim_cyl_up = [100 5 5.05]; %platform.dimensions h r1 r2
44  platform.loc_cyl_up = [0 0 -50]; %platform.location of the CoG of the unit
45  f_cyl_up = 0.35;%mass fraction
46  m_cyl_up = f_cyl_up*m_p; %mass
47  rho_cyl_up = m_cyl_up/(platform.dim_cyl_up(1)*pi*(platform.dim_cyl_up(3)^2 ...
48                          -platform.dim_cyl_up(2)^2));
49  Ixx_cyl_up = 1/12*m_cyl_up*(3*(platform.dim_cyl_up(2)^2 ...
50                      +platform.dim_cyl_up(3)^2)+platform.dim_cyl_up(1)^2);
51  Iyy_cyl_up = Ixx_cyl_up;
52  Izz_cyl_up = 1/2*m_cyl_up*(platform.dim_cyl_up(2)^2+platform.dim_cyl_up(3)^2);
53  MRB_cyl_upcg = diag([m_cyl_up m_cyl_up m_cyl_up ...
54                   Ixx_cyl_up Iyy_cyl_up Izz_cyl_up]);%cog inertia
55  S = skew(platform.loc_cyl_up); %platform P coordinate inertia
56  MRB_cyl_up = [MRB_cyl_upcg(1:3,1:3) -m_cyl_up*S;
57         m_cyl_up*S     MRB_cyl_upcg(4:6,4:6)-m_cyl_up*S^2];
58
59  % #3: lying cylinder
60  platform.dim_cyl_ly = [100 5 5.05]; %platform.dimensions l r1 r2
61  platform.loc_cyl_ly = [0 0 -100]; %platform.location of the CoG of the unit
62  f_cyl_ly = 0.1;%mass fraction
63  m_cyl_ly = f_cyl_ly*m_p; %mass
64  rho_cyl_ly = m_cyl_ly/(platform.dim_cyl_ly(1)*pi*(platform.dim_cyl_ly(3)^2 ...
65               -platform.dim_cyl_ly(2)^2));
66  Ixx_cyl_ly = 1/2*m_cyl_ly*(platform.dim_cyl_ly(2)^2+platform.dim_cyl_ly(3)^2);
67  Iyy_cyl_ly = 1/12*m_cyl_ly*(3*(platform.dim_cyl_ly(2)^2 ...
68                  +platform.dim_cyl_ly(3)^2)+platform.dim_cyl_ly(1)^2);
69  Izz_cyl_ly = Iyy_cyl_ly;
70  MRB_cyl_lycg = diag([m_cyl_ly m_cyl_ly m_cyl_ly ...
71                   Ixx_cyl_ly Iyy_cyl_ly Izz_cyl_ly]);%cog inertia
72  S = skew(platform.loc_cyl_ly); %platform P coordinate inertia
73  MRB_cyl_ly = [MRB_cyl_lycg(1:3,1:3) -m_cyl_ly*S;
74         m_cyl_ly*S     MRB_cyl_lycg(4:6,4:6)-m_cyl_ly*S^2];
75
76  % #4: hexagonal base %circumradius = chordlenght a, thickness t
77  a = 7; t=0.4;
78  platform.dim_hex = [a t];
79  platform.loc_hex = [0 0 -t]; %platform.location of the CoG of the unit
80  f_hex = 0.2;%mass fraction
81  m_hex = f_hex*m_p; %mass
82  A = 3/2*sqrt(3)*a^2;
83  rho_hex = m_hex/A;
84  Ixx_hex = 5/16*sqrt(3)*a^4*t*rho_hex;
85  Iyy_hex = Ixx_hex;
86  Izz_hex = m_hex*a^2/2;
87  MRB_hexcg = diag([m_hex m_hex m_hex ...
88                   Ixx_hex Iyy_hex Izz_hex]);%cog inertia
89  S = skew(platform.loc_hex); %platform P coordinate inertia
90  MRB_hex = [MRB_hexcg(1:3,1:3) -m_hex*S;
91         m_hex*S     MRB_hexcg(4:6,4:6)-m_hex*S^2];
92
93  % output
94  platform.MRB = MRB_box+MRB_cyl_up+MRB_cyl_ly+MRB_hex;
95  platform.CG = (platform.loc_box*m_box+platform.loc_cyl_up*m_cyl_up+...
96                      platform.loc_cyl_ly*m_cyl_ly+platform.loc_hex*m_hex)/...
97                (m_box+m_cyl_up+m_cyl_ly+m_hex);
98  disp('RBinertias.m - done: Rigid body inertias calculated in body fixed axi')
99  disp(' Platform (hexagonal) loaded with: box, hollow cylinder up, hollow cylinder lying')
100 disp([' Density components platform are: ' ...
```

```
    num2str(round([rho_hex rho_box rho_cyl_up rho_cyl_ly])) ' kg/m3'])
disp(['Mass fractions platform (tot. ' num2str(platform.m) ') are: '...
     num2str(round([m_hex m_box m_cyl_up m_cyl_ly])) ' kg'])

clear a t A m_p f_box m_box rho_box Ixx_box Iyy_box Izz_box MRB_boxcg S MRB_box
clear f_cyl_up m_cyl_up rho_cyl_up Ixx_cyl_up Iyy_cyl_up Izz_cyl_up MRB_cyl_upcg MRB_cyl_up
clear f_cyl_ly m_cyl_ly rho_cyl_ly Ixx_cyl_ly Iyy_cyl_ly Izz_cyl_ly MRB_cyl_lycg MRB_cyl_ly
clear f_hex m_hex rho_hex Ixx_hex Iyy_hex Izz_hex MRB_hexcg MRB_hex
```

C-5 loaddata.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% m-file loaddata.m
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%load the data to run the simulation

%--- Platform and Ship data from datafiles
load platformdata %load the platform parameters
load vesseldata %load the ship parameters
disp('Platform and vessel data loaded in workspace')
RBinertias %calculate rigid body inertia matrices (platform loading)
% platform.alpha=0; %for simpler calculations, ln is direction of forces!
% platform.b2=platform.b1; %for simpler calculations, ln is direction of forces!
%restructure FTF components (delete cell structures)
vessel.forceRAO_amp = [vessel.forceRAO.amp{1},vessel.forceRAO.amp{2},...
    vessel.forceRAO.amp{3},vessel.forceRAO.amp{4},...
    vessel.forceRAO.amp{5},vessel.forceRAO.amp{6}];
vessel.forceRAO_phase = [vessel.forceRAO.phase{1},vessel.forceRAO.phase{2},...
    vessel.forceRAO.phase{3},vessel.forceRAO.phase{4},...
    vessel.forceRAO.phase{5},vessel.forceRAO.phase{6}];
% roll static stability
% plot([0:pi/64:pi/2]*180/pi,...
%     -((vessel.main.CG(3)*vessel.main.g*vessel.main.m+...
%     platform.CG(3)*vessel.main.g*platform.m)*sin(0:pi/64:pi/2)),'g',..
% [0:pi/64:pi/2]*180/pi,vessel.G(4,4)*[0:pi/64:pi/2],'b',...
%     [0:pi/64:pi/2]*180/pi,vessel.G(4,4)*[0:pi/64:pi/2]+...
%     ((vessel.main.CG(3)*vessel.main.g*vessel.main.m+...
%     platform.CG(3)*vessel.main.g*platform.m)*sin(0:pi/64:pi/2)),'k')
% % 30deg roll->550 MNm 60deg 1500MNm netto! rightening moment
disp(['Platform linkage parameter alpha = ' num2str(platform.alpha)])

%-- initial conditions
X0 = [-8 0 0  0 0 -4  0*pi/180 0*pi/180 0]';%pose state
V0 = [0 0 0  0 0 0   0*12*pi/180 0*5*pi/180 0*10*pi/180/60]';%velocity state
% V0 = zeros(9,1);%velocity state

%-- wave field generation
disp('Generating wave field...')
wave.Mu = 0; %mean wave direction in NED inertial frame (North dir = 0)
wave.H13 = 4; %significant wave height (m)
wave.T1 = 6.5; %mean period (s)
disp([' Significant wave height = ' num2str(wave.H13) ...
      ', mean period T1 = ' num2str(wave.T1)])
%%% Random phases
if ~exist('wave','var')
    wave.loadseadflag = 1; %load random seed data from .mat file
else
%     wave.loadseadflag = 0; %create new random seed and store to .mat file
    wave.loadseadflag = 1; %overrule, force seed loading
end
if wave.loadseadflag == 1 %if set to 1, load rnd seed from .mat file
    load('randomseed.mat');
    rng(seed); %set random seed to loaded seed
else
    seed = rng; %random number generator
    save('randomseed.mat','seed'); %save seed data to .mat file
end
wave.seed = seed;
wave.alphas = vessel.headings;%headings (wave directions from ship frame)
wave.Nalpha = length(wave.alphas);%number of headings
wave.omegas = vessel.forceRAO.w; %wave angular frequencies
wave.Nomega = length(wave.omegas);%number of angular frequencies
```

```
wave.rand_phase = 2*pi*rand(wave.Nalpha,wave.Nomega); %random phase for realization
%%% Pierson-Moskowitz frequency spectrum
wave.Sohm = (wave.H13^2*wave.T1) * ( (0.11/(2*pi)) * ...
    ((wave.omegas.*wave.T1)./(2*pi)).^-5 ...
     .* exp(-0.44*((wave.omegas.*wave.T1)./(2*pi))).^-4) );%S(omega)
disp(' done')

%---MPC settings
mpc.N = 20; %horizon
mpc.dt = .2; %second per euler step
mpc.maxitt = 30; %maximum iterations per function call to funMPC
```

C-6 skew.m

```
function W=skew(w)
%#codegen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion W = skew(w)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forms a skew symmetric matrix such that w x a = Wa
W = [0   -w(3) w(2);
     w(3) 0   -w(1);
    -w(2) w(1) 0];
```

C-7 adjoint.m

```
function ad=adjoint(x)
%#codegen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion ad=adjoint(x)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% forms a adjoint matrix used in velocity transformations, dim(x)=6x1
ad = [0   -x(6) x(5)   0   -x(3) x(2);
      x(6)  0   -x(4)   x(3) 0   -x(1);
     -x(5)  x(4) 0     -x(2) x(1) 0;
      0     0    0      0   -x(5) x(6);
      0     0    0      x(5) 0   -x(4);
      0     0    0     -x(6) x(4) 0;];
```

C-8 coriolis symbolic.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Coriolis matrix robot on ship - symbolic computation
%%% Check brute force method of computation vs partitioned
%%% Part of MSc Thesis Wouter de Zeeuw 2011-2012
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% m-file that computes the coriolis matrix of a body fixed lagragrian
% system with a non trivial velocity place dependancy
% two methods of computation are compared to investigate equality because
% one method is much more efficient in implementation for dynamics
% symulation
%
%some key variables:
%phi  phi1-3 local platform coordinates phi4-9 local ship coordinates
%M mass matrix
%V quasi-velocities in the same directions as phi (& phidot)
%S velocity transformation matrix based on adjoint ad, due to the reduction
%  from 6dof to heave-roll-pitch the adjoint of the platform is empty
%  therefore ad = [0 0;0 ad_ship]
%Ad velocity transformation matrix from Body to Platform frame
%
%the brute force computations are based on From2012a and From2012b
% Cij = SUM_k (dMij/dphik-dMjk/dphii)|_{phi=0} v(k) + {partials of mass}
%       SUM_k (dSki/dphij-dSkj/dphii)*([M]V)_k|_{phi=0} {partials of S}
```

```
% the first part is calculated from partial derivatives of the orientation
% dependent mass matrix M
% the second part is based on the partial derivatives of the quasi-velocity
% transformation S
% the sectioned method is according to the method discussed in the thesis
% and assumes a symetric mass matrix

clear all
clc

%% First part: Checking partials of S
M = sym('M',[9 9]);
sym('M','real');
syms phi1 phi2 phi3 phi4 phi5 phi6 phi7 phi8 phi9 real
phi = [phi1 phi2 phi3 phi4 phi5 phi6 phi7 phi8 phi9]';
ad_phi = [0 -phi9 phi8 0 -phi6 phi5;
    phi9 0 -phi7 phi6 0 -phi4;
    -phi8 phi7 0 -phi5 phi4 0;
    0     0    0 0 -phi9 phi8;
    0     0    0 phi9 0 -phi7 ;
    0     0    0 -phi8 phi7 0];
S = [zeros(3,3) zeros(3,6);
    zeros(6,3) -1/2*ad_phi;];
syms Cdummy
syms mass  wp pp pq u v w p q r real;
V = [ wp pp pq u v w p q r]';
syms C1 C2
C1(1,1)=0;C1(9,9)=0;
C2(1,1)=0;C2(9,9)=0;
MV = (M*V);
for i = 1:9
    for m = 1:9
        for k = 1:9
            dSphii =  reshape(jacobian(S,phi(i)),[9 9]);
            dSphim =  reshape(jacobian(S,phi(m)),[9 9]);
            dSphik =  reshape(jacobian(S,phi(k)),[9 9]);
            Cdummy1 = (dSphik(m,i)-dSphii(m,k))*V(k);  %de Zeeuw
            Cdummy2 = (dSphim(k,i)-dSphii(k,m))*MV(k); %From2012
            C1(i,m) =  C1(i,m)+Cdummy1;
            C2(i,m) =  C2(i,m)+Cdummy2;
        end
    end
end
F1 = C1*M*V %EQUIVALENT!!
F3 = 2*subs(S,phi(4:9),[ u v w p q r]')'*M*V%EQUIVALENT!!
F2 = C2*V %EQUIVALENT!!
simplify(F1-F2) %==0
simplify(F3-F1) %==0

%% Second part: Checking partials of M
syms x1 x2 x3 real
x = sym('x',[3 1]) %z phi theta
Ad = [[cos(x(3)) 0 -sin(x(3)); ...
    sin(x(2))*sin(x(3)) cos(x(2)) sin(x(2))*cos(x(3));...
    cos(x(2))*sin(x(3)) -sin(x(2)) cos(x(2))*cos(x(3))] ...
    [ 0 x(1)*cos(x(3)) 0;...
    -x(1)*cos(x(2)) x(1)*sin(x(2))*sin(x(3)) 0;...
    x(1)*sin(x(2)) x(1)*cos(x(2))*sin(x(3)) 0];
    zeros(3) [cos(x(3)) 0 -sin(x(3)); ...
    sin(x(2))*sin(x(3)) cos(x(2)) sin(x(2))*cos(x(3)); ...
    cos(x(2))*sin(x(3)) -sin(x(2)) cos(x(2))*cos(x(3))]];
dAddx1=reshape(jacobian(Ad,x(1)),[6 6]); %partials of Adjoint
dAddx2=reshape(jacobian(Ad,x(2)),[6 6]);
dAddx3=reshape(jacobian(Ad,x(3)),[6 6]);
dAddx1_0=   subs(dAddx1,x(1:3),zeros(3,1)) %evaluate at phi=0 for \bar{x}=0
dAddx2_0=   subs(dAddx2,x(1:3),zeros(3,1))
dAddx3_0=   subs(dAddx3,x(1:3),zeros(3,1))
H = [0 0 0; 0 0 0; eye(3); 0 0 0]; %reduction matrix H
syms M_RBp1_1 M_RBp1_2 M_RBp1_3 M_RBp1_4 M_RBp1_5 M_RBp1_6 ...
    M_RBp2_1 M_RBp2_2 M_RBp2_3 M_RBp2_4 M_RBp2_5 M_RBp2_6 ...
    M_RBp3_1 M_RBp3_2 M_RBp3_3 M_RBp3_4 M_RBp3_5 M_RBp3_6 ...
    M_RBp4_1 M_RBp4_2 M_RBp4_3 M_RBp4_4 M_RBp4_5 M_RBp4_6 ...
    M_RBp5_1 M_RBp5_2 M_RBp5_3 M_RBp5_4 M_RBp5_5 M_RBp5_6 ...
    M_RBp6_1 M_RBp6_2 M_RBp6_3 M_RBp6_4 M_RBp6_5 M_RBp6_6 real
M_RBp = sym('M_RBp',[6 6]) %rigid body mass matrix of platform 6x6
M_vv = Ad'*M_RBp*Ad %ship part of platform mass matrix
```

```
M = [H';eye(6)]*M_vv*[H eye(6)]; %expand to all dof using H
syms Cdummy
syms mass  wp pp pq u v w p q r real;
V = [ wp pp pq u v w p q r]'; %velocity vector
syms C1
C1(1,1)=0;C1(9,9)=0;
dMdx1 =  reshape(jacobian(M,x(1)),[9 9]); %partials of full mass matrix
dMdx2 =  reshape(jacobian(M,x(2)),[9 9]);
dMdx3 =  reshape(jacobian(M,x(3)),[9 9]);
for i = 1:9 %brute force computation of partials of M
    for j = 1:9
        for k = 1:9;
            switch k
                case 1
                    dMdxk = dMdx1;
                case 2
                    dMdxk = dMdx2;
                case 3
                    dMdxk = dMdx3;
                otherwise
                    dMdxk = zeros(9);
            end
            switch i
                case 1
                    dMdxi = dMdx1;
                case 2
                    dMdxi = dMdx2;
                case 3
                    dMdxi = dMdx3;
                otherwise
                    dMdxi = zeros(9);
            end
            Cdummy = dMdxk(i,j)-1/2*dMdxi(j,k);
            C1(i,j) =  C1(i,j)+Cdummy*V(k);
        end
    end
end
C1 %brute force coriolis term of mass partials
%Sectioned method
dMvv_dx1 = (dAddx1)'*M_RBp*Ad;    %partials of Mvv x1
dMvv_dx1sym = dMvv_dx1+dMvv_dx1'; %symmetric part
dMMdx1 = [H';eye(6)]*(dMvv_dx1sym)*[H eye(6)]; %matrix 9x9 expansion
dMvv_dx2 = (dAddx2)'*M_RBp*Ad;    %partials of Mvv x2
dMvv_dx2sym = dMvv_dx2+dMvv_dx2'; %symmetric part
dMMdx2 = [H';eye(6)]*(dMvv_dx2sym)*[H eye(6)]; %matrix 9x9 expansion
dMvv_dx3 = (dAddx3)'*M_RBp*Ad;    %partials of Mvv x3
dMvv_dx3sym = dMvv_dx3+dMvv_dx3'; %symmetric part
dMMdx3 = [H';eye(6)]*(dMvv_dx3sym)*[H eye(6)]; %matrix 9x9 expansion
C2 = dMMdx1*V(1)+dMMdx2*V(2)+dMMdx3*V(3)...
    -1/2*[dMMdx1*V dMMdx2*V dMMdx3*V zeros(9,6)]' %coriolis matrix M part
%comparison
A = 2*randn(6); A = .5*(A+A')+4*eye(6);%random symmetric(!!) mass matrix
X = [A(:) ;rand(12,1)]; %random velocities
% (WARNING TAKES SEVERAL MINUTES ON 2.8GHz dual core)
replacestr = ['[M_RBp1_1 M_RBp1_2 M_RBp1_3 M_RBp1_4 M_RBp1_5 M_RBp1_6 '...
    'M_RBp2_1 M_RBp2_2 M_RBp2_3 M_RBp2_4 M_RBp2_5 M_RBp2_6 '...
    'M_RBp3_1 M_RBp3_2 M_RBp3_3 M_RBp3_4 M_RBp3_5 M_RBp3_6 '...
    'M_RBp4_1 M_RBp4_2 M_RBp4_3 M_RBp4_4 M_RBp4_5 M_RBp4_6 '...
    'M_RBp5_1 M_RBp5_2 M_RBp5_3 M_RBp5_4 M_RBp5_5 M_RBp5_6 '...
    'M_RBp6_1 M_RBp6_2 M_RBp6_3 M_RBp6_4 M_RBp6_5 M_RBp6_6 '...
    'x1 x2 x3 wp pp pq u v w p q r]'];
C1subs = eval(subs(C,replacestr,X))
C2subs = eval(subs(C2,replacestr,X))
C1subs-C2subs %EQUAL UP TO NUMERICAL PRECISION (1e-14)


%%
%%% TOTAL CORIOLIS MATRIX
% via partitioned method in M and ad_V in S
clear all
clc
%-- partials of S
syms nu1 nu2 nu3 nu4 nu5 nu6 real
syms xd1 xd2 xd3 real
nu = sym('nu',[6 1]);
xd = sym('xd',[3 1]);
```

```
177 V = [xd;nu]; %velocity vector
178 syms x1 x2 x3 real %platform orientation
179 x = sym('x',[3 1]); %z phi theta
180 %mass matrix including J*^T Ainfinity J*
181 syms M1_1 M1_2 M1_3 M1_4 M1_5 M1_6 M1_7 M1_8 M1_9 M2_1 M2_2 M2_3 M2_4 ...
182     M2_5 M2_6 M2_7 M2_8 M2_9 M3_1 M3_2 M3_3 M3_4 M3_5 M3_6 M3_7 M3_8 ...
183     M3_9 M4_1 M4_2 M4_3 M4_4 M4_5 M4_6 M4_7 M4_8 M4_9 M5_1 M5_2 M5_3 ...
184     M5_4 M5_5 M5_6 M5_7 M5_8 M5_9 M6_1 M6_2 M6_3 M6_4 M6_5 M6_6 M6_7 ...
185     M6_8 M6_9 M7_1 M7_2 M7_3 M7_4 M7_5 M7_6 M7_7 M7_8 M7_9 M8_1 M8_2 ...
186     M8_3 M8_4 M8_5 M8_6 M8_7 M8_8 M8_9 M9_1 M9_2 M9_3 M9_4 M9_5 M9_6 ...
187     M9_7 M9_8 M9_9 real
188 M = sym('M',[9 9]);%
189 for i=1:9;for j=i+1:9; M(i,j)=M(j,i);end;end; %force symmetry M
190 % M = [M(1:3,1:3) zeros(3,6);zeros(6,3) M(4:9,4:9)] %block diagonal mass
191 % M = diag(diag(M)) %diagonal mass
192 syms M_RBp1_1 M_RBp1_2 M_RBp1_3 M_RBp1_4 M_RBp1_5 M_RBp1_6 ...
193     M_RBp2_1 M_RBp2_2 M_RBp2_3 M_RBp2_4 M_RBp2_5 M_RBp2_6 ...
194     M_RBp3_1 M_RBp3_2 M_RBp3_3 M_RBp3_4 M_RBp3_5 M_RBp3_6 ...
195     M_RBp4_1 M_RBp4_2 M_RBp4_3 M_RBp4_4 M_RBp4_5 M_RBp4_6 ...
196     M_RBp5_1 M_RBp5_2 M_RBp5_3 M_RBp5_4 M_RBp5_5 M_RBp5_6 ...
197     M_RBp6_1 M_RBp6_2 M_RBp6_3 M_RBp6_4 M_RBp6_5 M_RBp6_6 real
198 M_RBp = sym('M_RBp',[6 6]); %rigid body mass matrix of platform 6x6
199 for i=1:6;for j=i+1:6; M_RBp(i,j)=M_RBp(j,i);end;end;%force symmetry M_RBp
200 H = [0 0 0; 0 0 0; eye(3); 0 0 0]; %reduction matrix H
201
202 %S-part of coriolis matrix
203 Cim = [zeros(3) zeros(3,6);
204         zeros(6,3) -[skew([nu(4) nu(5) nu(6)]) skew([nu(1) nu(2) nu(3)]);
205         zeros(3) skew([nu(4) nu(5) nu(6)])]';
206 Cij_S = Cim*M;
207
208 %-- partials of M
209 % rigid body mass of platform only as Mship is not orientation dependent in
210 % ship B frame description of dynamics
211 Ad = [[cos(x(3)) 0 -sin(x(3)); ...
212     sin(x(2))*sin(x(3)) cos(x(2)) sin(x(2))*cos(x(3));...
213     cos(x(2))*sin(x(3)) -sin(x(2)) cos(x(2))*cos(x(3))] ...
214     [ 0 x(1)*cos(x(3)) 0;...
215     -x(1)*cos(x(2)) x(1)*sin(x(2))*sin(x(3)) 0;...
216     x(1)*sin(x(2)) x(1)*cos(x(2))*sin(x(3)) 0];
217     zeros(3) [cos(x(3)) 0 -sin(x(3)); ...
218     sin(x(2))*sin(x(3)) cos(x(2)) sin(x(2))*cos(x(3)); ...
219     cos(x(2))*sin(x(3)) -sin(x(2)) cos(x(2))*cos(x(3))]];
220
221 dAddx1 =[ 0 0 0  0   cos(x3) 0;...
222         0 0 0 -cos(x2) sin(x2)*sin(x3) 0;...
223         0 0 0  sin(x2) cos(x2)*sin(x3) 0;...
224         0 0 0  0  0 0;...
225         0 0 0  0  0 0;...
226         0 0 0  0  0 0];
227 dAddx2 =[  0   0   0   0      0   0;...
228         cos(x2)*sin(x3) -sin(x2)  cos(x2)*cos(x3)   x1*sin(x2)  ...
229         x1*cos(x2)*sin(x3)  0;...
230         -sin(x2)*sin(x3) -cos(x2) -cos(x3)*sin(x2)  x1*cos(x2) ...
231         -x1*sin(x2)*sin(x3)  0;...
232         0   0   0   0      0   0;...
233         0   0   0  cos(x2)*sin(x3)  -sin(x2)  cos(x2)*cos(x3);...
234         0   0   0 -sin(x2)*sin(x3)   -cos(x2) -cos(x3)*sin(x2)];
235 dAddx3 =[ -sin(x3) 0   -cos(x3) 0  -x1*sin(x3)  0;...
236         cos(x3)*sin(x2) 0 -sin(x2)*sin(x3) 0 x1*cos(x3)*sin(x2)  0;...
237         cos(x2)*cos(x3) 0 -cos(x2)*sin(x3) 0 x1*cos(x2)*cos(x3)  0;...
238         0 0   0  -sin(x3)    0   -cos(x3);...
239         0 0   0 cos(x3)*sin(x2)  0 -sin(x2)*sin(x3);...
240         0 0   0 cos(x2)*cos(x3)  0 -cos(x2)*sin(x3)];
241
242 dMvv_dx1 = (dAddx1)'*M_RBp*Ad;    %partials of Mvv x1
243 dMvv_dx1sym = dMvv_dx1+dMvv_dx1'; %symmetric part
244 dMMdx1 = [H';eye(6)]*(dMvv_dx1sym)*[H eye(6)]; %matrix 9x9 expansion
245 dMvv_dx2 = (dAddx2)'*M_RBp*Ad;    %partials of Mvv x2
246 dMvv_dx2sym = dMvv_dx2+dMvv_dx2'; %symmetric part
247 dMMdx2 = [H';eye(6)]*(dMvv_dx2sym)*[H eye(6)]; %matrix 9x9 expansion
248 dMvv_dx3 = (dAddx3)'*M_RBp*Ad;    %partials of Mvv x3
249 dMvv_dx3sym = dMvv_dx3+dMvv_dx3'; %symmetric part
250 dMMdx3 = [H';eye(6)]*(dMvv_dx3sym)*[H eye(6)]; %matrix 9x9 expansion
251
252 Cij_M = dMMdx1*V(1)+dMMdx2*V(2)+dMMdx3*V(3)...
```

```
253     -1/2*[dMMdx1*V dMMdx2*V dMMdx3*V zeros(9,6)]'; %coriolis matrix M part
254
255 %-- Force
256 Cij = Cij_S+Cij_M;
257 F = Cij*V %force (same side of equation as mass)
258 %% symbolic mass matrix
259 clear all
260 syms M11v M22v M33v M44v M55v M66v M15v M24v real
261 syms M11p M22p M33p M44p M55p M66p M15p M24p real
262 MRBv = diag([M11v M22v M33v M44v M55v M66v]);MRBv(1,5)=M15v;MRBv(5,1)=M15v;MRBv(2,4)=M24v;MRBv(4,2)=
        M24v;
263 MRBp = diag([M11p M22p M33p M44p M55p M66p]);MRBp(1,5)=M15p;MRBp(5,1)=M15p;MRBp(2,4)=M24p;MRBp(4,2)=
        M24p;
264 syms x1 x2 x3 real;x = sym('x',[3 1]);
265 R_theta = [cos(x(3)) 0 -sin(x(3));0 1 0; sin(x(3)) 0 cos(x(3))];
266 R_phi = [1 0 0; 0 cos(x(2)) sin(x(2));0 -sin(x(2)) cos(x(2))];
267 Ad_theta=[R_theta R_theta*skew([0 0 0])';zeros(3) R_theta];
268 Ad_phi = [R_phi R_phi*skew([0 0 0])';zeros(3) R_phi];
269 Ad_c = [eye(3) eye(3)*skew([0 0 x(1)])'; zeros(3) eye(3)];
270 Ad = Ad_phi*Ad_theta*Ad_c %Adjoint matrix
271 H = [0 0 0; 0 0 0; eye(3); 0 0 0];
272 M_vv = Ad'*MRBp*Ad;
273 M_p = [H';eye(6)]*M_vv*[H eye(6)]
274 M_s = [zeros(3) zeros(3,6);zeros(6,3) MRBv]
275 M = M_p + M_s
276 syms xd1 xd2 xd3 nu1 nu2 nu3 nu4 nu5 nu6 real
277 xd = sym('xd',[3,1]);
278 V = [xd1 xd2 xd3 nu1 nu2 nu3 nu4 nu5 nu6]';
279 C1 =  reshape(jacobian(M,x(1)),[9 9])*V(1)+...
280       reshape(jacobian(M,x(2)),[9 9])*V(2)+...
281       reshape(jacobian(M,x(3)),[9 9])*V(3)
282
283 C2 = -1/2*[ jacobian(M*V,x(1))';...
284     jacobian(M*V,x(2))';...
285     jacobian(M*V,x(3))';...
286     zeros(6,3) 2*adjoint(M(4:9,:)*V)']
287 C = C1+C2
```

C-9 funC.m

```
1  function C = funC(M,M_RBp,V,X)
2  %#codegen
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %%% funtion C=funC(M,M_RBp,V,X)
5  %%% part of MSc Thesis Wouter de Zeeuw
6  %%% TU Delft - GustoMSC 2011-2012
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  % Calculated the coriolis matrix for the coupled ship-platform 2body 9dof
9  % system. Inputs are:
10 % M     9x9 mass matrix of coupled system (dep. on platform pose)
11 % M_RBp 6x6 rigid body mass matrix of platform
12 % V     9x1 velocity vector with entries [xd;nu] stacked platform,ship
13 %         where xd is 3x1 heave-roll-pitch (B)rate and nu 6x1 linear-angular
14 % X     9x1 pose vector
15 %    [3x1 platform pose heave-roll-pitch (x), 6x1 zeros (eta) (irrelevant)]
16 %
17 % Assumed are symetry for the platform M_RBp matrix and the total M matrix
18 % this is not checked, all error handling is omitted for sake of excecution
19 % speed
20 % Methods are discussed in Chapter 3 of W.A. de Zeeuw's MSc Thesis
21 %
22 % Examples:
23 % % #1: 10000 random coriolis matrices execution time
24 % M = 2*randn(9); M = .5*(M+M')+5*(rand+2)*eye(9); %random symetric mass matrix
25 % M_RBp = 2*randn(6); M_RBp = .5*(M_RBp+M_RBp')+5*(rand+2)*eye(6);
26 % V = 5*rand(9,1);
27 % X = [randn(2,1);5+3*randn;zeros(6,1)];
28 % C = funC(M,M_RBp,V,X)
29 % tic;for count=1:10000; C = funC(M,M_RBp,V,X);end;
30 % disp([num2str(count) ' Coriolis matrices calculated, ']);toc;
31 % disp([num2str(round(count/toc)) ' excecutions per second']) %~6k
32 %
33 % % #2: only ship movements, (symbolic toolbox), diagonal inertia
34 % syms m Ix Iy Iz u v w p q r real;
```

```
% M = [m m m Ix Iy Iz];
% V = [0 0 0 u v w p q r]';
% funC(diag([zeros(1,3) M]),diag(M),V,zeros(9,1))*V
% %equal to eqn. 3.55 Handbook of Marine Craft Hydrodynamics and Motion Control Thor I. Fossen
%
% % #3: rigid body, off diagonal inertia
% syms m Ix Iy Iz u v w p q r real;
% M = [m m m Ix Iy Iz];
% V = [0 0 0 u v w p q r]';
% syms r1 r2 r3 real;rcg = [ r1 r2 r3]';
% % Fossen RB coriolis equations Handbook of Marine Craft Hydrodynamics
% I_b = diag(M(4:6))-m*skew(rcg)^2;
% tau_coriolis1a=[zeros(3) -m*skew([u v w])-m*skew(skew([p q r])*rcg); -m*skew([u v w])-m*skew(skew([p q r])*rcg) m*skew(skew([u v w])*rcg)-skew(I_b*[p q r])]*V(4:end)
% tau_coriolis1b=[m*skew([p q r]) -m*skew([p q r])*skew(rcg);m*skew(rcg)*skew([p q r]) -skew(I_b*[p q r])]*V(4:end)
% %Rigid body mass matrix
% M_O = [zeros(3) zeros(3,6); zeros(6,3) [diag(M(1:3)) -m*skew(rcg);m*skew(rcg) I_b]];
% tau_coriolis2=funC(M_O,zeros(6),V,zeros(9,1))*V
% simplify(tau_coriolis1a-tau_coriolis1b)
% simplify(tau_coriolis1a-tau_coriolis2(4:9))

%-- input handling: reduce X to x and expand V in xd and nu
x = X(1:3);
% eta = X(4:9); %eta is not used in funC()
xd = V(1:3);
nu = V(4:9);
H = [0 0 0; 0 0 0; eye(3); 0 0 0]; %reduction matrix H

%-- partials of S
Mv = M(4:9,:)*V;
adtilde_Mv = -[zeros(3)  skew([Mv(1) Mv(2) Mv(3)]);
            skew([Mv(1) Mv(2) Mv(3)])     skew([Mv(4) Mv(5) Mv(6)])];
C_S = [zeros(3) zeros(3,6);zeros(6,3) adtilde_Mv];

%-- partials of M
%%% rigid body mass of platform only as Mship is not orientation dependent in
%%% ship B frame description of dynamics, Ad transforms velocity from B->P
%%% partials of adjoint
Ad =[
           cos(x(3)),          0,          -sin(x(3)),                    0,          x(1)*cos(x(3)),
                                                                                                  0
 sin(x(2))*sin(x(3)),  cos(x(2)), cos(x(3))*sin(x(2)),     -x(1)*cos(x(2)), x(1)*sin(x(2))*sin(x(3))
      ,                                                                                           0
 cos(x(2))*sin(x(3)), -sin(x(2)), cos(x(2))*cos(x(3)),      x(1)*sin(x(2)), x(1)*cos(x(2))*sin(x(3))
      ,                                                                                           0
                   0,          0,                    0,           cos(x(3)),
                                                                                              0,          -sin(x(3))
                   0,          0,                    0, sin(x(2))*sin(x(3)),                cos(x(2))
                                                                                                   , cos(x(3))*sin(x(2))
                   0,          0,                    0, cos(x(2))*sin(x(3)),               -sin(x(2))
                                                                                                   , cos(x(2))*cos(x(3))];

%partials of adjoint
dAddx1 =[ 0 0 0  0   cos(x(3)) 0;...
          0 0 0 -cos(x(2)) sin(x(2))*sin(x(3)) 0;...
          0 0 0  sin(x(2)) cos(x(2))*sin(x(3)) 0;...
          0 0 0  0  0 0;...
          0 0 0  0  0 0;...
          0 0 0  0  0 0];
dAddx2 =[ 0   0   0   0   0   0;...
    cos(x(2))*sin(x(3)) -sin(x(2))  cos(x(2))*cos(x(3)) x(1)*sin(x(2))  ...
    x(1)*cos(x(2))*sin(x(3))  0;...
    -sin(x(2))*sin(x(3)) -cos(x(2)) -cos(x(3))*sin(x(2)) x(1)*cos(x(2)) ...
    -x(1)*sin(x(2))*sin(x(3))  0;...
    0   0   0   0   0   0;...
    0   0   0  cos(x(2))*sin(x(3))  -sin(x(2))  cos(x(2))*cos(x(3));...
    0   0   0 -sin(x(2))*sin(x(3))   -cos(x(2)) -cos(x(3))*sin(x(2))];
dAddx3 =[ -sin(x(3)) 0   -cos(x(3)) 0  -x(1)*sin(x(3))  0;...
 cos(x(3))*sin(x(2)) 0 -sin(x(2))*sin(x(3)) 0 x(1)*cos(x(3))*sin(x(2)) 0;...
 cos(x(2))*cos(x(3)) 0 -cos(x(2))*sin(x(3)) 0 x(1)*cos(x(2))*cos(x(3)) 0;...
 0   0   0  -sin(x(3))    0   -cos(x(3));...
 0   0   0 cos(x(3))*sin(x(2))  0 -sin(x(2))*sin(x(3));...
 0   0   0 cos(x(2))*cos(x(3))  0 -cos(x(2))*sin(x(3))];

dMvv_dx1 = (dAddx1)'*M_RBp*Ad;    %partials of Mvv x(1)
dMvv_dx1sym = dMvv_dx1+dMvv_dx1'; %symmetric part
```

```
dMMdx1 = [H';eye(6)]*(dMvv_dx1sym)*[H eye(6)]; %matrix 9x9 expansion
dMvv_dx2 = (dAddx2)'*M_RBp*Ad;     %partials of Mvv x(2)
dMvv_dx2sym = dMvv_dx2+dMvv_dx2'; %symmetric part
dMMdx2 = [H';eye(6)]*(dMvv_dx2sym)*[H eye(6)]; %matrix 9x9 expansion
dMvv_dx3 = (dAddx3)'*M_RBp*Ad;     %partials of Mvv x(3)
dMvv_dx3sym = dMvv_dx3+dMvv_dx3'; %symmetric part
dMMdx3 = [H';eye(6)]*(dMvv_dx3sym)*[H eye(6)]; %matrix 9x9 expansion

C_M = dMMdx1*V(1)+dMMdx2*V(2)+dMMdx3*V(3)...
    -1/2*[dMMdx1*V dMMdx2*V dMMdx3*V zeros(9,6)]'; %coriolis matrix M part

%-- coriolis matrix
C = C_S+C_M; %output
f = -C*V(:); %coriolis forcing terms (RHS)
f = f(:);
end
```

C-10 funM.m

```
function M = funM(M_RBsJAinfJ,M_RBp,X)
%#codegen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion M = funM(M_RBs,M_RBp,X)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculates the pose dependent inertia matrix
% Inputs are:
% M_RBsJAinfJ   6x6 rigid body inertia matrix of the ship plus the
%               inifinity added mass Ainf transposed from the hydrodynamic
%              equilibrium frame H to the boat fixed frame H via J* (cnst.)
% M_RBp          6x6 rigid body inertia matrix of the platform
% X              9x1 pose vector
%    [3x1 platform pose heave-roll-pitch (x), 6x1 zeros (eta) (irrelevant)]
%
%Example with random symetric RB mass matrices and random pose:
% M_RBsJAinfJ = 3*randn(6);
% M_RBsJAinfJ = .5*(M_RBsJAinfJ+M_RBsJAinfJ')+15*(rand+2)*eye(6);
% M_RBp = 2*randn(6); M_RBp = .5*(M_RBp+M_RBp')+5*(rand+2)*eye(6);
% X = [randn(2,1);5+3*randn;zeros(6,1)];
% M = funM(M_RBsJAinfJ,M_RBp,X)
% tic;for count=1:10000; M = funM(M_RBsJAinfJ,M_RBp,X);end;
% disp([num2str(count) ' Mass matrices calculated, ']);toc;
% disp([num2str(round(count/toc)) ' excecutions per second']) %~20k

%-- input handling: reduce X to x and expand V in xd and nu
x = X(1:3);
% eta = X(4:9); %eta is not used in funM()
H = [0 0 0; 0 0 0; eye(3); 0 0 0]; %reduction matrix H

%-- mass matrix components
% rigid body mass of platform only as Mship is not orientation dependent in
% ship B frame description of dynamics, Ad transforms velocity from B->P
% entries are [R R[x]_x; 0 R] where R is a rotation matrix and [x]_x a skew
% symmetric matrix in x
Ad = [[cos(x(3)) 0 -sin(x(3)); ...
    sin(x(2))*sin(x(3)) cos(x(2)) sin(x(2))*cos(x(3));...
    cos(x(2))*sin(x(3)) -sin(x(2)) cos(x(2))*cos(x(3))] ...
    [ 0 x(1)*cos(x(3)) 0;...
    -x(1)*cos(x(2)) x(1)*sin(x(2))*sin(x(3)) 0;...
    x(1)*sin(x(2)) x(1)*cos(x(2))*sin(x(3)) 0];
    zeros(3) [cos(x(3)) 0 -sin(x(3)); ...
    sin(x(2))*sin(x(3)) cos(x(2)) sin(x(2))*cos(x(3)); ...
    cos(x(2))*sin(x(3)) -sin(x(2)) cos(x(2))*cos(x(3))]];
%ship observed part of platform mass (as x is a releative coordinate)
M_vv = Ad'*M_RBp*Ad;
%total platform origining inertia matrix (9x9)
M_p = [H';eye(6)]*M_vv*[H eye(6)];
%ship inertia component (only acting on ship, only rigid body component)
M_s = [zeros(3) zeros(3,6);
       zeros(6,3) M_RBsJAinfJ];

%-- total orientation dependent mass matrix
M = M_p + M_s;
```

```
56 end
```

C-11 funXV.m

```
1  function [Xdot,J_xv] = funXV(V,X)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %%% funtion [Xdot,J_xv] = funXV(V,X)
4  %%% part of MSc Thesis Wouter de Zeeuw
5  %%% TU Delft - GustoMSC 2011-2012
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  % quasi velocity transform of platform and ship
8  % transforms body fixed velocities V (9x1) = [xdot nu] to the rate of
9  % change of the pose vector Xdot (9x1) = d/dt[x_beta eta]
10 %
11 % inputs:
12 % V = heave-roll-pitch rate platform (B!) ...
13 %                          surge-sway-heave-roll-pitch-yaw boat (B)
14 % X = [x_beta eta] pose vector
15 % output:
16 % Xdot = d/dt[x_beta eta]
17 %      = [dz_p/dt phidot_p thetadot_p dx_s/dt dy_s/dt dz_s/dt ...
18 %                                          phidot_s thetadot_s psidot_s]
19
20 %-- transformation matrix J
21 % platform attitude transformation E:("ground"-to euler)
22 % Eaccent_inv = [1/cos(X(3)) 0 0;
23 %                0           1 0;
24 %                tan(x(3))   0 1];
25 % Eaccent_inv_P = [1/cos(X(3)) 0 ;
26 %                 0           1 ];%only required part (psi platform =0)
27 E_inv_P  =  [1    sin(X(2))*tan(X(3)); % 2x2 only required part (psi platform =0)
28              0    cos(X(2))];
29 % boat rotation matrix, dependent on angle part in eta
30 R_ItoB =  [ cos(X(9))*cos(X(8)), cos(X(8))*sin(X(9)),-sin(X(8));
31             cos(X(9))*sin(X(7))*sin(X(8)) - cos(X(7))*sin(X(9)), ...
32             cos(X(7))*cos(X(9)) + sin(X(7))*sin(X(9))*sin(X(8)), ...
33             cos(X(8))*sin(X(7));
34             sin(X(7))*sin(X(9)) + cos(X(7))*cos(X(9))*sin(X(8)) ,...
35             cos(X(7))*sin(X(9))*sin(X(8)) - cos(X(9))*sin(X(7)), ...
36             cos(X(7))*cos(X(8))];
37 % boat rotation attitude transformation E':("body"-to euler)
38 % E_inv = 1/cos(X(8))*[cos(X(9)) sin(X(9)) 0;%slower 270k/s
39 %         -cos(X(8))*sin(X(9)) cos(X(8))*cos(X(9)) 0;
40 %         cos(X(9))*sin(X(8)) sin(X(9))*sin(X(8)) cos(X(8))]
41 % E_inv = [cos(X(9))/cos(X(8)) sin(X(9))/cos(X(8)) 0;
42 %           -sin(X(9)) cos(X(9)) 0;
43 %           cos(X(9))*tan(X(8)) sin(X(9))*tan(X(8)) 1];%300k/s (fastest)
44 E_accentinv_S = [1    sin(X(7))*tan(X(8))   cos(X(7))*tan(X(8));
45                  0    cos(X(7))            -sin(X(7));
46                  0    sin(X(7))/cos(X(8))   cos(X(7))/cos(X(8))];
47
48 J_xv =  [[1 zeros(1,2); zeros(2,1) E_inv_P ] zeros(3,6);
49       zeros(6,3) [R_ItoB' zeros(3);zeros(3) E_accentinv_S]];
50
51 %-- quasi velocities transformed to pose change rate
52 Xdot = J_xv*V;
53 end
```

C-12 funTaugrav.m

```
1  function tau_grav = funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs)
2  %#codegen
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %%% funtion tau_grav = funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs)
5  %%% part of MSc Thesis Wouter de Zeeuw
6  %%% TU Delft - GustoMSC 2011-2012
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  % Function that calculates the gravity forces and torques acting on the
9  % 3 dof of the platform and the 6 dof of the ship. The gravity is
10 % transformed to the body fixed frame to use in the Langranian description.
11 % The forces are acting on a body fixed vector r away from the CoG's of the
```

```
% platform(P) and the boat(B)
%
% Inputs:
% X      pose state vector [heave roll pitch (platform relative to boat)
%        surge sway heave roll pitch yaw]
% m_p    mass platform
% m_s    mass ship
% g      gravity
% r_opgp vector from origin platform to cog described in P (const)
% r_obgb vector from origin boat to cog described in B (const)
%
% Example:
% m_p = 1234321; m_s = 12344321; g=9.81;
% r_opgp = [0 0 -50]'; r_obgs = [0 0 -12]';
% X = [-rand; rand(8,1) ];  funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs)
% tic;for count=1:10000;  funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs);end;
% disp([num2str(count) ' gravity transformations calculated, ']);toc;
% disp([num2str(round(count/toc)) ' excecutions per second']) %~8.2k

%-- rotation matrices and bar{c} vector (heave)
R_BtoP = [  cos(X(3)),   0,    -sin(X(3));
            sin(X(2))*sin(X(3)),  cos(X(2)), cos(X(3))*sin(X(2));
            cos(X(2))*sin(X(3)), -sin(X(2)), cos(X(2))*cos(X(3))];
c = [0 0 X(1)];
R_ItoB =  [ cos(X(9))*cos(X(8)), cos(X(8))*sin(X(9)),-sin(X(8));
            cos(X(9))*sin(X(7))*sin(X(8)) - cos(X(7))*sin(X(9)), ...
            cos(X(7))*cos(X(9)) + sin(X(7))*sin(X(9))*sin(X(8)), ...
            cos(X(8))*sin(X(7));
            sin(X(7))*sin(X(9)) + cos(X(7))*cos(X(9))*sin(X(8)) ,...
            cos(X(7))*sin(X(9))*sin(X(8)) - cos(X(9))*sin(X(7)), ...
            cos(X(7))*cos(X(8))];

%-- gravity vectors inertial I frame
f_cog_s_I = [0;0;m_s*g];
f_cog_p_I = [0;0;m_p*g];

%-- gravity in B frame
H = [0 0 0; 0 0 0; eye(3); 0 0 0]; %reduction matrix H
Htilde = [1 0 0; 0 1 0; zeros(3); 0 0 1]; %reduction matrix Htilde

tau_grav_s = [zeros(3);...
        R_ItoB;...
        skew(r_obgs)*R_ItoB]*f_cog_s_I; %gravity ship in ship CO and B-frame
    %rotation to B frame and translation from CG platform to CO platform
R_SkewRrR = [R_ItoB;skew(R_BtoP'*r_opgp(:))*R_ItoB];
% selection of off planar components (untouched) and planar translated to
% CO B via c
tau_grav_p = [H'*R_SkewRrR;
         [eye(3);skew(c)]*Htilde'*R_SkewRrR]...
         *f_cog_p_I;
tau_grav = tau_grav_s+tau_grav_p;
% [zeros(3) R_ItoB' zeros(3)]*(tau_grav+[zeros(5,1); tau_grav(1) ;zeros(3,1)]) %when
% rotated back only in z

end
```

C-13 funTauhydrostat.m

```
function tau_hydrostat = funTauhydrostat(X,G,~,VCO,flag_mooring)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% function tau_hydrostat = funTauhydrostat(X,G,CB,VCO)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%#codegen
% function to calculate the hydrostatic load
% X is the pose vector (9x1)
% G is the hydrostatic matrix acting in CB
% CB is the vector from the origin of the body coordinates to the center of
%    buoyancy
% VCO is the vertical distance from the keel to the origin (B-frame)
% flag_mooring switches on the elastic mooring lines

%-- input handling: reduce X to eta
```

```
17 eta = X(4:9); eta = eta(:);
18 %%% adapt eta to have draught in the heave spot
19 eta(3) = eta(3)+VCO;
20
21 %-- elastic mooring cable
22 if flag_mooring
23     k_mooring = G(3,3)/10^2;
24     G(1,1) = k_mooring;
25     G(2,2) = k_mooring;
26 end
27
28 %-- force due to stiffness matrix
29 tau_hydrostat_eta_I = -G*eta;
30 %rotate forces, assume roll and pitch axis parralel to inertial?
31 R_ItoB =  [ cos(X(9))*cos(X(8)), cos(X(8))*sin(X(9)),-sin(X(8));
32             cos(X(9))*sin(X(7))*sin(X(8)) - cos(X(7))*sin(X(9)), ...
33             cos(X(7))*cos(X(9)) + sin(X(7))*sin(X(9))*sin(X(8)), ...
34             cos(X(8))*sin(X(7));
35             sin(X(7))*sin(X(9)) + cos(X(7))*cos(X(9))*sin(X(8)),...
36             cos(X(7))*sin(X(9))*sin(X(8)) - cos(X(9))*sin(X(7)), ...
37             cos(X(7))*cos(X(8))];
38 % tau_hydrostat_eta_B = [R_ItoB zeros(3);zeros(3) eye(3)]*tau_hydrostat_eta_I;
39 % tau_hydrostat_eta_B = [R_ItoB zeros(3);zeros(3) R_ItoB]*tau_hydrostat_eta_I;
40 tau_hydrostat_eta_B = [R_ItoB zeros(3);zeros(3) eye(3)]*tau_hydrostat_eta_I;
41 tau_hydrostat = [zeros(3,6);eye(6)]*tau_hydrostat_eta_B; %resize output to 9x1
42
43 %%% NO ROTATION, assume B is alligned with the inertial only on B frame coordinates
44 % tau_hydrostat = [zeros(3,6);eye(6)]*tau_hydrostat_eta_I; %
45 % slides sideways when ship rolls because gravity is decomposed, rotation of
46 % only forces (not moments) gives realistic results. Note that the constant
47 % stiffness matrix in allready a raw linear approximation only valid to
48 % about 20 deg roll and pitch heave in the straight walled part of the hull
49 end
```

C-14 funKinv.m

```
1  function [ell,R_ItoB,R_BtoP,Ru_k,Rd1_k,Rd2_k,Rl_k,a_B,stopsim,J_lx] = funKinv(X,p_b1,p_b2,p_a,
       p_alpha,p_normd1,p_normd2)
2  %#codegen
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %%% funtion [ell,R_ItoB,R_BtoP,Ru_k,Rd1_k,Rd2_k,Rl_k,a_B,stopsim,J_lx] = ...
5  %%% funKinv(X,p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2)
6  %%% part of MSc Thesis Wouter de Zeeuw
7  %%% TU Delft - GustoMSC 2011-2012
8  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9  % Inverse kinematics solution of sarrus legs spread 120 degree apart. Finds
10 % the leg lengths from a known pose
11 %
12 % Inputs:
13 % X      pose state vector [heave roll pitch (platform relative to boat)
14 %        surge sway heave roll pitch yaw]
15 % platform is a structure with two base vectors (b1/b2) one platform vector
16 % (a) 2 lenghts of the linkages (normd1) and (normd2) and the fraction along d1
17 % where the hydraulics connect to the linkage
18 %         b1: [3x1 double]
19 %         b2: [3x1 double]
20 %          a: [3x1 double]
21 %      alpha: 0...1
22 %      normd1: scalar
23 %      normd2: scalar
24 %
25 % Outputs
26 % ell    lenghts of legs
27 % R_ItoB    boat rotation matrices(3x3)
28 % R_BtoP    platform  rotation matrix(3x3)
29 % Ru_k    U joint rotation matrices(3times 3x3)
30 % Rd1_k   d1 linkage rotation matrices(3times 3x3)
31 % Rd2_k   d2 linkage rotation matrices(3times 3x3)
32 % Rl_k    leg linkage rotation matrices(3times 3x3)
33 % a_B     hydraulic connection translation vectors 3 times 3x1 + c_z!
34 % J_lx   jacobian velocity transformation from leg to joint
35 %
36 % Example:
37 % p_b1 = platform.b1;p_b2 = platform.b2; p_a = platform.a;
```

```
% p_alpha = platform.alpha; p_normd1 = platform.normd1;
% p_normd2 =  platform.normd2
% X = [-rand; rand(8,1) ]; funKinv(X,platform)
% tic;for count=1:10000; funKinv(X,platform);end;
% disp([num2str(count) ' inverse kinematic solutions, ']);toc;
% disp([num2str(round(count/toc)) ' excecutions per second'])
% %~7.2k not vectorized, ~9.2k vectorized

% X =  [-8 0 0 20 20 -10 30*pi/180 10*pi/180 45*pi/180]';
% X = [-4 20*pi/180 -0*pi/180 20 20 -10 0*pi/180 -0*pi/180 0*pi/180];
%--- input handling
stopsim = 0;
flag_errhand = 1; %flag error handling on or off
flag_calcJlx = 1; %flag jacobian calcualtion on or off
flag_calcR = 1; %flag rotation matrix calculation handling on or off
errstr = '';
%%% decompose pose vector (only dependent on x_beta=X(1:3))
R_BtoP = [  cos(X(3)),   0,    -sin(X(3));
    sin(X(2))*sin(X(3)),  cos(X(2)), cos(X(3))*sin(X(2));
    cos(X(2))*sin(X(3)), -sin(X(2)), cos(X(2))*cos(X(3))];
c = [0 0 X(1)]';
J_lx = zeros(3,6); %preassign jacobian

%--- caculation of leg coordinates
notvecotorized = 0; %switch vectorized computation off
if notvecotorized
%%% Z rotation for 3 legs
g = zeros(3);gn = zeros(3);normg = zeros(3,1);
d1 = zeros(3);d2 = zeros(3);
l = zeros(3);ell = zeros(3,1);ln = zeros(3);
a = zeros(3);b2 = zeros(3);
for N=1:3 %%%<><> not vectorized, looping
    psis = -[0,2/3*pi,4/3*pi];
    psi_N = psis(N);
    Rz = [cos(psi_N) sin(psi_N) 0;
        -sin(psi_N) cos(psi_N) 0 ;
        0 0 1];
    %%% auxilariy vector from b1 to a
    g(:,N) = c + R_BtoP'*Rz*p_a-Rz*p_b1; %auxiliary vector
    normg(N) = norm(g(:,N));
    gn(:,N) = g(:,N)/normg(N);
    if sum(normg>(p_normd1+p_alpha*p_normd2))>0 && flag_errhand
        errstr='Error: Planar linkage overstretched';%error check over stretch link
        stopsim = 1;
    end
    %%% angle of lower linkage with vertical
    negz_B = [0;0;-1];
    ang_zb_g = acos(real(gn(:,N)'*negz_B));
    ang_g_d1 = acos((normg(N)^2+p_normd1^2-p_normd2^2)...
        /(2*normg(N)*p_normd1));
    ang_zb_d1 = ang_zb_g+ang_g_d1;
    %ang_zb_d1*180/pi% check: -z=normd1+normd2 ->0 V -z=normd1=normd2 ->60 V
                    %        z=-0.01->~90 V z=0 NAN
    %-- linkage vectors
    d1(:,N) = p_normd1*(negz_B*cos(ang_zb_d1)+...
        (-Rz*p_b1/norm(Rz*p_b1))*sin(ang_zb_d1));
    d2(:,N) = Rz*p_b1+d1(:,N)-c-R_BtoP'*Rz*p_a;

    %check: alpha=0 z=-d1-d2 b1=b2 ->l=-z V, alpha=1 l=d1 V
    l(:,N) = p_alpha*d2(:,N)+R_BtoP'*Rz*p_a+c-Rz*p_b2;
    ell(N) = norm(l(:,N));
    ln(:,N) = l(:,N)/ell(N); %unit vector in leg direction
    a(:,N) = Rz*p_a;
    b2(:,N) = Rz*p_b2;
end %for 3 legs
J_lx = nan(6,3); %the jacobian is not calculated in the non vectorized loop

else %%%<><> vectorized computation of leg lenghts AND JACOBIAN
%%% vectorized computation of leg lengths and jacobian
Rz1 =  [cos(2*pi/3) sin(2*pi/3) 0;
    -sin(2*pi/3) cos(2*pi/3) 0 ;
    0 0 1]';
Rz2 = [cos(4*pi/3) sin(4*pi/3) 0;
    -sin(4*pi/3) cos(4*pi/3) 0 ;
    0 0 1]';
a = [p_a  Rz1*p_a  Rz2*p_a];
```

```
114 b1 = [p_b1  Rz1*p_b1  Rz2*p_b1];
115 normb1 = sqrt(sum(b1.^2,1));
116 b1n = b1.*repmat(normb1(:)'.^-1,[3 1]);
117 b2 = [p_b2  Rz1*p_b2  Rz2*p_b2];
118 a_B = R_BtoP'*a;
119 z_B = [0 0 1]'; %Down direction of B coordinate
120 %%% out of plane direction for cross products
121 i = skew(z_B)*b1n; %cross product of construction base vectors b1 and z_B
122 %%% auxilariy vector from b1 to a
123 g = [c c c]+a_B-b1;
124 normg = sqrt(sum(g.^2,1));
125 gn = g.*repmat(normg(:)'.^-1,[3 1]);%unit dir g
126 if sum(normg>(p_normd1+p_normd2))>0 && flag_errhand
127         errstr='Error: Planar linkage overstretched';%error check over stretch link
128         stopsim=1;
129 end  %end if error stretch
130 %%% angle of lower linkage with vertical
131 negz_B = [0;0;-1];
132 ang_zb_g = acos(-gn(3,:));
133 cosrule = (normg.^2+ones(1,3)*p_normd1^2-ones(1,3)*p_normd2^2)...
134     ./(2*normg.*ones(1,3)*p_normd1); %cosine rule argument
135 cosrule = (cosrule<1).*cosrule+(cosrule>1);
136 ang_g_d1 = acos(cosrule);%cosine rule argument
137 ang_zb_d1 = ang_zb_g+ang_g_d1;
138 if sum(cosrule>1)>0 && flag_errhand
139     errstr='Error: Planar linkage overstretched';%error check over stretch link
140     stopsim=1;
141 end %end if error stretch
142 %%% linkage vectors
143 d1 = p_normd1*(negz_B*cos(ang_zb_d1)+(-b1/norm(p_b1))...
144      .*[sin(ang_zb_d1);sin(ang_zb_d1);sin(ang_zb_d1)]);
145 d1n = d1/p_normd1; %unit vector in botom linkage direction
146 d2 = b1+d1-[c c c]-a_B;
147 d2n = d2/p_normd2; %unit vector in dir top linkage
148 %%% legs
149 l = p_alpha*d2+a_B+[c c c]- b2;
150 ell = sqrt(sum(l.^2,1))';ell = ell(:); %output leg lengths
151 ln = l.*repmat(ell(:)'.^-1,[3 1]); %unit vector in leg direction
152 end %end if vecotrized
153 %%% auxilariy vectors from b1 to D (g' gaccent)
154 gaccent = g-p_alpha*d2;
155 normgaccent = sqrt(sum(gaccent.^2,1));
156 gaccentn = gaccent.*repmat(normgaccent(:)'.^-1,[3 1]);%unit dir g'
157
158 %-- Construction of jacobian that relates leg rate to body velocity
159 if flag_calcJlx
160 %%% J_DA (from point A to D) = J_VDvA*J_vaVA
161 %%% 1. J_vaVA (inverse decomposition of instantanious velocities)
162 %%% 2. J_VDvA (translation along rotation radii of inst.veloc.)
163 for k=1:3
164 %%% 1. J_vaVA
165 % % > slow routine with inverse()
166 % gn_x = cross(gn,i); %orthogonal to gn (in plane)
167 % d1n_x = cross(d1n,i); %orthogonal to d1n (in plane)
168 % J_vaVA = [gn_x(:,k) d1n_x(:,k) i(:,k)]^-1; %8.6kHz
169 % > fast routine with hard coded inverse of the 3x3 matrix with orthogonals
170  %leg number
171 Delta = 1/( (gn(3,k)*d1n(2,k) - gn(2,k)*d1n(3,k))*i(1,k) + ...
172             (gn(1,k)*d1n(3,k) - gn(3,k)*d1n(1,k))*i(2,k) +...
173             (gn(2,k)*d1n(1,k) - gn(1,k)*d1n(2,k))*i(3,k) );
174 J_vaVA = Delta*[...
175     d1n(1,k)*i(2,k)^2-d1n(2,k)*i(1,k)*i(2,k)+d1n(1,k)*i(3,k)^2-d1n(3,k)*i(1,k)*i(3,k) ...
176     d1n(2,k)*i(1,k)^2-d1n(1,k)*i(2,k)*i(1,k)+d1n(2,k)*i(3,k)^2-d1n(3,k)*i(2,k)*i(3,k) ...
177     d1n(3,k)*i(1,k)^2-d1n(1,k)*i(3,k)*i(1,k)+d1n(3,k)*i(2,k)^2-d1n(2,k)*i(2,k)*i(3,k);
178     -gn(1,k)*i(2,k)^2+gn(2,k)*i(1,k)*i(2,k)-gn(1,k)*i(3,k)^2+gn(3,k)*i(1,k)*i(3,k) ...
179     -gn(2,k)*i(1,k)^2+gn(1,k)*i(2,k)*i(1,k)-gn(2,k)*i(3,k)^2+gn(3,k)*i(2,k)*i(3,k) ...
180     -gn(3,k)*i(1,k)^2+gn(1,k)*i(3,k)*i(1,k)-gn(3,k)*i(2,k)^2+gn(2,k)*i(2,k)*i(3,k);
181     i(1,k)/Delta i(2,k)/Delta i(3,k)/Delta];
182 %%% 2. J_VDvA
183 J_VDvA = [normgaccent(k)/normg(k)*(skew(gaccentn(:,k))*i(:,k)) ...
184     (1-p_alpha)*(skew(d2n(:,k))*i(:,k)) zeros(3,1)];
185 %%% 3. combine J_DA
186 J_DA = J_VDvA*J_vaVA;
187 %%% constructing J_lx from J_DA (row by row)
188 J_lx(k,:) = [ln(:,k)'*J_DA (skew(a_B(:,k))*J_DA'*ln(:,k))'];
189 end%end for all legs J_lx loop
```

```
190 end% flag_calcJlx
191
192 %-- Rotation matrices output of
193 R_ItoB =  eye(3); %preassign rotation matrix to boat frame
194 Ru_k  = [eye(3) eye(3) eye(3)];%preassign U joint rotation matrices(3times 3x3)
195 Rd1_k = [eye(3) eye(3) eye(3)]; %preassign d1 linkage rotation matrices(3times 3x3)
196 Rd2_k = [eye(3) eye(3) eye(3)]; %preassign d2 linkage rotation matrices(3times 3x3)
197 Rl_k  = [eye(3) eye(3) eye(3)]; %preassign leg rotation matrices(3times 3x3)
198 if flag_calcR
199 R_ItoB =  [ cos(X(9))*cos(X(8)), cos(X(8))*sin(X(9)),-sin(X(8));
200         cos(X(9))*sin(X(7))*sin(X(8)) - cos(X(7))*sin(X(9)), ...
201         cos(X(7))*cos(X(9)) + sin(X(7))*sin(X(9))*sin(X(8)), ...
202         cos(X(8))*sin(X(7));
203         sin(X(7))*sin(X(9)) + cos(X(7))*cos(X(9))*sin(X(8)) ,...
204         cos(X(7))*sin(X(9))*sin(X(8)) - cos(X(9))*sin(X(7)), ...
205         cos(X(7))*cos(X(8))]; %rotation to boat frame
206 Rd2_k =[ [ cross(i(:,1),d2n(:,1)) i(:,1)  d2n(:,1)   ]' ...
207          [cross(i(:,2),d2n(:,2)) i(:,2)  d2n(:,2)   ]' ...
208          [ cross(i(:,3),d2n(:,3)) i(:,3)    d2n(:,3)]']; %YES
209 Rd1_k = [ [cross(d1n(:,1),i(:,1)) i(:,1)  -d1n(:,1)]' ...
210          [cross(d1n(:,2),i(:,2)) i(:,2)  -d1n(:,2)]'...
211          [cross(d1n(:,3),i(:,3)) i(:,3)  -d1n(:,3)]']; %OK
212 Rl_k = -[[-cross(ln(:,1),i(:,1))  -i(:,1) ln(:,1)]...
213          [-cross(ln(:,2),i(:,2)) -i(:,2) ln(:,2)] ...
214          [-cross(ln(:,3),i(:,3)) -i(:,3) ln(:,3)]];
215 i_P = R_BtoP'*i;
216 Ru_k = [[cross(i_P(:,1),d2n(:,1)) i_P(:,1) d2n(:,1) ]'...
217          [cross(i_P(:,2),d2n(:,2)) i_P(:,2) d2n(:,2) ]'...
218          [cross(i_P(:,3),d2n(:,3)) i_P(:,3) d2n(:,3)   ]' ];%OK
219 a_B = a_B + [c c c];
220 end%flag_calcR
221
222 %-- error handling
223 if  ( (X(2)>=pi/2) || (X(2)<=-pi/2) ) && flag_errhand
224     errstr='Error: Platform roll not in {-pi/2..pi/2}';
225     stopsim = 1;
226 end
227 if  ( (X(3)>=pi/2) || (X(3)<=-pi/2) ) && flag_errhand
228     errstr='Error: Platform pitch not in {-pi/2..pi/2}';
229     stopsim = 1;
230 end
231 if  ( (X(7)>=pi/2) || (X(7)<=-pi/2) ) && flag_errhand
232     errstr='Error: Ship roll not in {-pi/2..pi/2}';
233     stopsim = 1;
234 end
235 if  ( (X(8)>=pi/2) || (X(8)<=-pi/2) ) && flag_errhand
236     errstr='Error: Ship pitch not in {-pi/2..pi/2}';
237     stopsim = 1;
238 end
239 if X(1)>0 && flag_errhand%(positive (down) heave: platform crashed into main deck
240     errstr='Error: Zero or negative platform heave detected, platform crashed into maindeck';
241     stopsim = 1;
242 end
243 coder.extrinsic('disp');
244 if stopsim == 1
245     disp(errstr)
246 end
247 end
```

C-15 funKforw.m

```
1  function [xhat,J_lxhat] = funKforw(ell,x0,p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2)
2  %#codegen
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %%% funtion [xhat,J_lxhat] = funKforw(ell,x0,p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2)
5  %%% part of MSc Thesis Wouter de Zeeuw
6  %%% TU Delft - GustoMSC 2011-2012
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  % function that finds the forward kinematical solution of the 3dof platform
9  % with 3 sarrus linkages. The solution is found via an iterative scheme
10 % dependent on the approximated pose that converges towards the accual pose
11 %
12 % inputs:
13 % ell                          current leg lengths
```

```
% x0                              initial guess pose state (x_beta=heave roll
%                                 pitch)
% p_b1,p_b2,p_a                   platform linkage base, hydraulic base,
%                                 topplate hydraulic connection poitns
% p_alpha,p_normd1,p_normd2       linkage lengths and fractional length of
%                                 connection to hydraulics in upper (d2)
%
% outputs:
% xhat                            found pose from iterations
% J_lxhat                         jacobian velocity tranformation from xdot
%                                 to ldot
%
% % Example #1: inverse and forward similarity
% load platformdata
% Xactual = [-0.31 3*pi/180 1*pi/180]
% ellmeasured = funKinv([Xactual 0 0 0 0 0 0],platform.b1,platform.b2,platform.a,...
%     platform.alpha,platform.normd1,platform.normd2)'
% badguess = [-1 0 0]
% Xapproximated = funKforw(ellmeasured,badguess,platform.b1,platform.b2,platform.a,...
%     platform.alpha,platform.normd1,platform.normd2)
% Xapproximated - Xactual'
%
% % Example #2: speed of excecution
% badguess = [-5 -.1 0]';
% goodguess = Xactual(:).*(ones(3,1)+1/5*randn(3,1)); %about 10-20% off
% tic; for cnt = 1:10000; %only 800kHz
% Xapproximated = funKforw(ellmeasured,badguess,platform.b1,platform.b2,platform.a,...
%     platform.alpha,platform.normd1,platform.normd2);end;
% toc; disp(['Bad guess gives excecution speed ' num2str(cnt/toc) 'Hz']);
% tic; for cnt = 1:10000; %about 1.2kHz, perfect guess gives 2.2kHz
% Xapproximated = funKforw(ellmeasured,goodguess,platform.b1,platform.b2,platform.a,...
%     platform.alpha,platform.normd1,platform.normd2);end;
% toc; disp(['Good guess gives excecution speed ' num2str(cnt/toc) 'Hz']);


%--- input handling
% coder.extrinsic('exist'); %does not work in simulink
% if ~exist('x0','var') %if no guess, set guess to zero
%     x0 = zeros(3,1);
%     x0(1) = -1;% x0(2) = 0; x0(3) = 0;
% end
% if isempty(x0)
%     x0 = [-1 ; zeros(2,1)];
% end
xj = x0(:); %set current estimate to intial guess
ell = ell(:); %current leg lenghts
%%% base and top plate vectors (constant in their own frames)
Rz1 =  [cos(2*pi/3) sin(2*pi/3) 0;
       -sin(2*pi/3) cos(2*pi/3) 0 ;
        0 0 1]';
Rz2 = [cos(4*pi/3) sin(4*pi/3) 0;
      -sin(4*pi/3) cos(4*pi/3) 0 ;
       0 0 1]';
a = [p_a  Rz1*p_a  Rz2*p_a];%topplate hydraulic connections
b1 = [p_b1  Rz1*p_b1  Rz2*p_b1];%coordinates of linkage base
normb1 = sqrt(sum(b1.^2,1));
b1n = b1.*repmat(normb1(:)'.^-1,[3 1]);
b2 = [p_b2  Rz1*p_b2  Rz2*p_b2];%coordinates of hydraulic base
z_B = [0 0 1]'; %Down direction of B coordinate
%%% out of plane direction for cross products
i = skew(z_B)*b1n; %cross product of construction base vectors b1 and z_B
J_lxhat = zeros(3,6); %preassign jacobian

%<><> start of NR iteration loop
j=1; %iteration nr 1,
norm_delta_ell = 1; %reset norm step change
while (j<5)&&norm_delta_ell>1e-2
%-- inverse kinematics with current estimate xj
%%% current rotation matirces
R_BtoP = [ cos(xj(3)),                      0,          -sin(xj(3));
           sin(xj(2))*sin(xj(3)),  cos(xj(2)), cos(xj(3))*sin(xj(2));
           cos(xj(2))*sin(xj(3)), -sin(xj(2)), cos(xj(2))*cos(xj(3))];
c = [0 0 xj(1)]';
%%% auxilariy vectors from b1 to A (g)
a_B = R_BtoP'*a;
g = [c c c]+a_B-b1;%from base to cyl joint at platform
```

```
90  normg = sqrt(sum(g.^2,1));%length
91  gn = g.*repmat(normg(:)'.^-1,[3 1]);%unit dir g
92  %%% angle of lower linkage with vertical
93  negz_B = [0;0;-1];
94  ang_zb_g = acos(-gn(3,:));
95  ang_g_d1 = acos((normg.^2+ones(1,3)*p_normd1^2-ones(1,3)*p_normd2^2)...
96            ./(2*normg.*ones(1,3)*p_normd1));
97  ang_zb_d1 = ang_zb_g+ang_g_d1;
98  %%% linkage vectors
99  d1 = p_normd1*(negz_B*cos(ang_zb_d1)+(-b1/norm(p_b1))...
100       .*[sin(ang_zb_d1);sin(ang_zb_d1);sin(ang_zb_d1)]);
101 d1n = d1/p_normd1; %unit vector in botom linkage direction
102 d2 = b1+d1-[c c c]-a_B;
103 d2n = d2/p_normd2; %unit vector in dir top linkage
104 lj = p_alpha*d2+a_B+[c c c]- b2; %hydraulic legs of iteration j
105 ellj = sqrt(sum(lj.^2,1))'; %lenghts of hydraulic legs
106 ellj = ellj(:);
107 lnj = lj.*repmat(ellj(:)'.^-1,[3 1]); %unit vector in leg direction
108 %%% auxilariy vectors from b1 to D (g' gaccent)
109 gaccent = g-p_alpha*d2;
110 normgaccent = sqrt(sum(gaccent.^2,1));
111 gaccentn = gaccent.*repmat(normgaccent(:)'.^-1,[3 1]);%unit dir g'
112
113 %-- Construction of jacobian that relates leg rate to body velocity
114 %%% J_DA (from point A to D) = J_VDvA*J_vaVA
115 %%% 1. J_vaVA (inverse decomposition of instantanious velocities)
116 %%% 2. J_VDvA (translation along rotation radii of inst.veloc.)
117 for k=1:3
118 %%% 1. J_vaVA
119 % % > slow routine with inverse()
120 % gn_x = cross(gn,i); %orthogonal to gn (in plane)
121 % d1n_x = cross(d1n,i); %orthogonal to d1n (in plane)
122 % J_vaVA = [gn_x(:,k) d1n_x(:,k) i(:,k)]^-1; %8.6kHz
123 % > fast routine with hard coded inverse of the 3x3 matrix with orthogonals
124  %leg number
125 Delta = 1/( (gn(3,k)*d1n(2,k) - gn(2,k)*d1n(3,k))*i(1,k) + ...
126             (gn(1,k)*d1n(3,k) - gn(3,k)*d1n(1,k))*i(2,k) +...
127             (gn(2,k)*d1n(1,k) - gn(1,k)*d1n(2,k))*i(3,k) );
128 J_vaVA = Delta*[...
129     d1n(1,k)*i(2,k)^2-d1n(2,k)*i(1,k)*i(2,k)+d1n(1,k)*i(3,k)^2-d1n(3,k)*i(1,k)*i(3,k) ...
130     d1n(2,k)*i(1,k)^2-d1n(1,k)*i(2,k)*i(1,k)+d1n(2,k)*i(3,k)^2-d1n(3,k)*i(2,k)*i(3,k) ...
131     d1n(3,k)*i(1,k)^2-d1n(1,k)*i(3,k)*i(1,k)+d1n(3,k)*i(2,k)^2-d1n(2,k)*i(2,k)*i(3,k);
132     -gn(1,k)*i(2,k)^2+gn(2,k)*i(1,k)*i(2,k)-gn(1,k)*i(3,k)^2+gn(3,k)*i(1,k)*i(3,k) ...
133     -gn(2,k)*i(1,k)^2+gn(1,k)*i(2,k)*i(1,k)-gn(2,k)*i(3,k)^2+gn(3,k)*i(2,k)*i(3,k) ...
134     -gn(3,k)*i(1,k)^2+gn(1,k)*i(3,k)*i(1,k)-gn(3,k)*i(2,k)^2+gn(2,k)*i(2,k)*i(3,k);
135     i(1,k)/Delta i(2,k)/Delta i(3,k)/Delta]; %16.3kHz(!) 2x faster
136 %%% 2. J_VDvA
137 J_VDvA = [normgaccent(k)/normg(k)*(skew(gaccentn(:,k))*i(:,k)) ...
138     (1-p_alpha)*(skew(d2n(:,k))*i(:,k)) zeros(3,1)];
139 %%% 3. combine J_DA
140 J_DA = J_VDvA*J_vaVA;
141 %%% constructing J_lxhat from J_DA (row by row)
142 J_lxhat(k,:) = [lnj(:,k)'*J_DA (skew(a_B(:,k))*J_DA'*lnj(:,k))'];
143 end
144
145 %-- combine velocity jacobian with pose euler attitude transformation
146 %%% retain only 3,4,5 (heave roll pitch)
147 H = [zeros(2,3);eye(3);zeros(1,3)];
148 % the only nonzero velocities (planar constraints)
149 J_lxhat_red = J_lxhat*H;
150 %%% [J_l,x_beta]^-1 = J_x_beta,x *[J_lxhat]^-1
151 J_xbetax = [1   0     0;%eye(1)
152             0   1     sin(xj(2))*tan(xj(3)); %E'^-1
153             0   0     cos(xj(2))];
154
155 %--- Newton-Rhapson iteration step
156 J_lxhatbeta_red_inv = J_xbetax*J_lxhat_red^-1;
157 xj_plus1 = xj + J_lxhatbeta_red_inv*(ell-ellj);
158 xj = xj_plus1; j = j+1;
159 norm_delta_ell = norm(ell-ellj);
160 %J_lxhat2 = [lnj;skew(a_B(:,1))*lnj(:,1) skew(a_B(:,2))*lnj(:,2) skew(a_B(:,3))*lnj(:,3)]';
161 end % <><> end NR iteration loop
162 xhat = xj(:);
163 %  coder.extrinsic('disp'); disp(j);
164
165 %%%%% funKforw appendix
```

```
166 %%% symbolic derivation of inverse term:
167 % syms d1 d2 d3 g1 g2 g3 i1 i2 i3 real;
168 % d = sym('d',[3 1]);i = sym('i',[3 1]);g = sym('g',[3 1]);
169 % JJ = simplify(((i1^2 + i2^2 + i3^2)*(d1*g2*i3 - d1*g3*i2 - d2*g1*i3 ...
170 % + d2*g3*i1 + d3*g1*i2 - d3*g2*i1))*[cross(g,i) cross(d,i) i]^(-1))
171 % J_vaVA = Delta*[...
172 %      d1n(1,k)*i(2,k)^2-d1n(2,k)*i(1,k)*i(2,k)+d1n(1,k)*i(3,k)^2-d1n(3,k)*i(1,k)*i(3,k)...
173 %      d1n(2,k)*i(1,k)^2-d1n(1,k)*i(2,k)*i(1,k)+d1n(2,k)*i(3,k)^2-d1n(3,k)*i(2,k)*i(3,k)...
174 %      d1n(3,k)*i(1,k)^2-d1n(1,k)*i(3,k)*i(1,k)+d1n(3,k)*i(2,k)^2-d1n(2,k)*i(3,k)*i(2,k);
175 % -gn(1,k)*i(2,k)^2+gn(2,k)*i(1,k)*i(2,k)-gn(1,k)*i(3,k)^2+gn(3,k)*i(1,k)*i(3,k)...
176 %      -gn(2,k)*i(1,k)^2+gn(1,k)*i(2,k)*i(1,k)-gn(2,k)*i(3,k)^2+gn(3,k)*i(2,k)*i(3,k)...
177 %      -gn(3,k)*i(1,k)^2+gn(1,k)*i(3,k)*i(1,k)-gn(3,k)*i(2,k)^2+gn(2,k)*i(3,k)*i(2,k);
178 %      i(1,k)/Delta i(2,k)/Delta i(3,k)/Delta];
```

C-16 waverealization.m

```
1  function waverealization
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %%% Wave Field Realization
4  %%% 3D simulation ship with platform
5  %%% Wouter de Zeeuw 2012 - Gusto - MSc Thesis
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  % Generate wave field from spectral density function (Pierson-Moskowitz)
8  % 3D spread is calculated with cosine^2 function (3 components: 0 45 90)
9  % wave field realization with random fase (possible from seed)
10
11 avirecord_flag=0;%switch on avi movie capture
12 plot_flag = 0;%switch on 3D surface plot of wavefield
13 if avirecord_flag
14     vidObj = VideoWriter(['waverealization.avi']);
15     vidObj.FrameRate = 10;
16     open(vidObj);
17     scrsz = get(0,'ScreenSize');%make the frame size fixed
18     fig1=figure('Position',[1 1 scrsz(3)/2 scrsz(4)/2]);%quarter scr
19 end
20
21 %%% Inputs
22 % t = 0; %time
23 H13 = 7; %significant wave height (m)
24 T1 = 10; %mean period (s)
25 omega1 = 0.1; %first anglular wave frequency (rad/s)
26 omegaN = 2; %last angular wave frequency
27 deltaomega = 0.05; %discretization step
28 alphas = [0:45:90]*pi/180; %wave directions
29 mu = 45*pi/180; %mean wave direction (used in directional spread)
30 g = 9.81; %gravity m/s2
31 loadseadflag = 1; %load random seed data from .mat file
32
33 %%% Process
34 omegas = omega1:deltaomega:omegaN; %angular frequencies
35 omegas = [0.1000    0.1200    0.1400    0.1500    0.1600    0.1700    0.1800    0.2000    0.2200
       0.2400    0.2600    0.2800    0.3000    0.3100    0.3200    0.3300    0.3400    0.3500    0.3600
       0.3900    0.4100    0.4300    0.4600    0.4900    0.5200    0.5500    0.5800    0.6100
       0.6500    0.7500    0.8500    0.9500    1.0500    1.1500    1.2500    1.3500    1.4500    1.6000
       1.8000    2.0000];
36 % %random periods, MUCH MUCH HIGHER(inf)GREATEST COMMON DEVISOR, LESS REPETITION RISK
37 % omegas = sort(2*rand(1,39));
38 Nomega = length(omegas); %number of angular frequencies
39 Nalpha = length(alphas); %number of wave directions
40 idir0 = find(alphas==0*pi/180); %direction index
41 idir45 = find(alphas==45*pi/180);
42 idir90 = find(alphas==90*pi/180);
43
44 %%% Random phases
45 if loadseadflag == 1 %if set to 1, load rnd seed from .mat file
46     load('randomseed.mat');
47     rng(seed); %set random seed to loaded seed
48 else
49     seed = rng; %random number generator
50     save('randomseed.mat','seed'); %save seed data to .mat file
51 end
52 phase = 2*pi*rand(Nalpha,Nomega); %random phase for realization
53
54
```

```
%%% Pierson-Moskowitz frequency spectrum
Sohm = (H13^2*T1) * ( (0.11/(2*pi)) * ((omegas.*T1)./(2*pi)).^-5 ...
    .* exp(-0.44*((omegas.*T1)./(2*pi)).^-4) );%S(omega)
Sohmalpha = repmat([2/pi*cos(alphas-mu).^2]',[1 Nomega]).*repmat(Sohm,[Nalpha 1]);
zeta = (2*Sohmalpha*deltaomega).^(1/2); %wave amplitude (surface elevation)
figure(2)
plot(omegas,Sohm,'k')
title(['Pierson-Moskowitz Spectrum, H_{1/3} = ' num2str(H13) ', T_1 = ' num2str(T1)])
xlabel('Angular wave frequency (rad/s)')
ylabel('Spectral density (m^2/s)')

pause
h=.1;
T = 0:h:60*6    ;%seconds-minutes-hours
zeta_t0 = zeros(length(T),1);
%%% 3d plot of wave surface
w = 100; %length of the wave field
w = 0; %only (0,0)
dx = 5; %meter
dy = 5; %meter (should be equal to dx)
[X,Y]=meshgrid(-w/2:dx:w/2,-w/2:dy:w/2); %create spatial grid
[nx,ny]=size(X);
dirxy = [1/sqrt(2),1/sqrt(2)]; %unit vector in 45 degree direction
for t = T;
    tic
    Z = zeros(nx,ny);
    for i=1:nx
        for j=1:ny
            Z(i,j) = sum(...
                zeta(idir0,:).*cos(omegas.^2./g*X(i,j) ...
                - omegas*t + phase(idir0,:))...
                +  zeta(idir45,:).*cos(omegas.^2./g*(dirxy*[X(i,j),Y(i,j)]') ...
                - omegas*t + phase(idir45,:))...
                +  zeta(idir90,:).*cos(omegas.^2./g*Y(i,j) ...
                - omegas*t + phase(idir90,:))...
                );
        end
    end
    if plot_flag
        if avirecord_flag
            currFrame = struct('cdata', [],'colormap', []);
        end
        Z = Z+0; %elevate the surface
        colormap winter
        surf(X,Y,Z,'linestyle','none','FaceColor','interp','FaceLighting','phong')
        camlight right
        %  axis equal
        zlim([-10 10])
        drawnow
        dt = h-(toc-round(toc));
        pause(dt)
        if avirecord_flag
            set(fig1, 'color', 'white')
            currFrame=getframe(fig1);
            if isempty(currFrame.cdata)
                pause(0.3)
                currFrame=getframe(fig1);
            end %frame check
            writeVideo(vidObj,currFrame);
        end%record if
    end%plot if
    zeta_t0(round((t+h)/h)) = Z(round(end/2),round(end/2));
end %for loop
pause
plot(T,zeta_t0,'b')
title('Surface elevation timeseries coordinate (0,0)')
if avirecord_flag
    close(vidObj)
end
end
```

C-17 funSurfelev.m

```
function zetaXt = funSurfelev(X,t,Mu,Sohm,rand_phase,omegas,alphas,g)
%#codegen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% function zeta = funSurfelev(X,t,Mu,Sohm,rand_phase,omegas,alphas,g)
%%% 3D simulation ship with platform
%%% Wouter de Zeeuw 2012 - Gusto - MSc Thesis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate surface elevation of waves at a grid around the location of vessel from the
% Pierson-Moskowitz spectral density
%
% Inputs:
% X  pose (North, East, .. , Yaw) is used
% t  time
% Mu mean heading waves in I frame
% Sohm wave energies in frequency spectrum
% rand_phase random phases
% omegas wave frequencies
% alphas headings
% g gravity
%
% Example: (full alphas runs at only 6! Hz, therfore the lowest alphas are
% omitted) this saves 20% vectorized runs at 113 Hz.
% loaddata %load data script of simulation, must be run to gen wavefield
% zetaXt =
% funSurfelev(X,t,wave.Mu,wave.Sohm,wave.rand_phase,wave.omegas,...
% wave.alphas,vessel.main.g)

%-- input handling
flag_reducealphas = 1;
flag_vectorized = 1;
north = X(4); east = X(5); yaw = X(9);
% yaw = yaw-2*pi*floor(yaw/(2*pi));% force yaw in 0..2pi
% if yaw == 2*pi;yaw=0;end
% NE = [north;east];
Nalpha = length(alphas);
Nomega = length(omegas);

%-- grid generation
Nx = 30; Dx = 5; Ny=30; Dy=5;
% Xmesh=-(Nx-1)*Dx/2+north:Dx:(Nx-1)*Dx/2+north;
Xmesh=linspace(-(Nx-1)*Dx/2+north,(Nx-1)*Dx/2+north,Nx);
% Ymesh = -(Ny-1)*Dy/2+east:Dy:(Ny-1)*Dy/2+east;
Ymesh = linspace(-(Ny-1)*Dy/2+east,(Ny-1)*Dy/2+east,Ny);
NElist = [repmat(Xmesh(:),[length(Ymesh) 1]),...
    sort(repmat(Ymesh(:),[length(Xmesh) 1]))]; %list of all combinations
xtilde = [cos(alphas(:)) sin(alphas(:))]*NElist';% inertial quasi coord along alphas
% zetaXt = zeros(length(Xmesh)*length(Ymesh),1);

%--- freqency integration
% cosine squared spread (only -1/2pi..1/2pi)
alphas_min_Mu = (alphas-Mu)-((alphas-Mu)>pi)*2*pi;%center around alphas-Mu
cos2 = 2/pi*cos(alphas_min_Mu).^2.* ...
    (((alphas_min_Mu)<=pi/2)+((alphas_min_Mu)>=-pi/2)==2);%select only half
if flag_reducealphas %include the heighest alphas
    [~,idx]=sort(cos2,2,'descend');
    cos2red = cos2(idx(1:3));
    alphas = alphas(idx(1:3)); %only the top3 of original height
    cos2red = cos2(idx(1:3));
    Nalphared = length(alphas);
    xtildered = xtilde(idx(1:3),:);
    rand_phasered = rand_phase(idx(1:3),:);
else
    cos2red=cos2;
    xtildered=xtilde;
    Nalphared=Nalpha;
    rand_phasered=rand_phase;
end
Sohmalpha = repmat(cos2red(:),[1 Nomega]).*repmat(Sohm(:)',[Nalphared 1]);%
deltaomegas = [diff(omegas(:)')/2 0]+[0 diff(omegas(:)')/2];
deltaomegas([1 end]) = deltaomegas([1 end])*2; %frequency steps (non uniform)
% amplitudes zeta of frequencies omega in inertial directions alpha
zeta = (2*Sohmalpha.*repmat(deltaomegas(:)',[Nalphared 1])).^(1/2); %amplitudes

%-- surface elevation
%wave amplitude (surface elevation)
 if ~flag_vectorized
```

```
for j =1:numel(zetaXt); %for loop to save memory
phases_I = repmat(omegas(:)',[Nalphared 1]).^2./g.*...
      repmat(xtildered(:,j),[1 Nomega])...
       -repmat(omegas(:)',[Nalphared 1])*t+rand_phasered;
zetaXt(j) = sum(sum( zeta.*cos(phases_I) ));
end
else %vectorized computation
idx = repmat([1:size(xtildered,1)]',[1 Nomega])';
vecXtilde = xtildered(idx(:)',:)';
vecOmegas = repmat(repmat(omegas(:)',[ 1 Nalphared]).^2./g,[size(vecXtilde,1) 1]);
vecOmegat = repmat(repmat(omegas(:)',[ 1 Nalphared])*t,[size(vecXtilde,1) 1]);
zeta_transpose = zeta';
vecZeta =  repmat(zeta_transpose(:)',[length(Xmesh)*length(Ymesh) 1]);
vecRand_phase = rand_phasered';
vecRand_phase = repmat(vecRand_phase(:)',[length(Xmesh)*length(Ymesh) 1]);
vecPhases_I = vecOmegas.*vecXtilde-vecOmegat+vecRand_phase;
zetaXt = sum(vecZeta.*cos(vecPhases_I),2);
 end %vectorized computation
```

C-18 funTauwave.m

```
function [Tau_wave,zetaXt] = funTauwave(X,t,Mu,Sohm,rand_phase,omegas,alphas,forceRAO_phase,
     forceRAO_amp,g)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% function zeta = funSurfelev(X,t,Mu,Sohm,rand_phase,omegas,alphas)
%%% 3D simulation ship with platform
%%% Wouter de Zeeuw 2012 - Gusto - MSc Thesis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%#codegen
% Calculate surface elevation of waves at location of vessel from the
% Pierson-Moskowitz spectral density and the forces induces by the waves
% according to the force transfer functions from linear wave components
%
% Inputs:
% X  pose state of which elements 4,5 and 9 (north east yaw) are used
% t  current simulation time
% Mu mean wave direction in inertial frame (angle from North)
% Sohm frequency component
% rand_phase Nalpha x Nomega matrix of random phases of wave component
% omegas angular frequencies of wave components
% alphas directions of wave compontents (I frame) assumed same set of
%        angles as the FTF functions (B frame)
%forceRAO_phase force RAO phase shifts for headings x (alphas * DOF )(6)
%forceRAO_amp force RAO amplitudes for headings x (alphas * DOF )(6)
%g gravitational constant


%-- input handling
north = X(4); east = X(5); yaw = X(9);
% force yaw in 0..2pi
yaw = yaw-2*pi*floor(yaw/(2*pi));
if yaw == 2*pi;yaw=0;end
NE = [north;east];
%xtilde = [cos(yaw+alphas);sin(yaw+alphas)]'*NE;% quasi coordinate along alphas
% inertial quasi coordinate along alphas
xtilde = [cos(alphas(:)) sin(alphas(:))]*NE;
% mu = Mu - yaw; %relative wave angle to ship
Nalpha = length(alphas);
Nomega = length(omegas);
%expand FTF amplitude and phase and degree of freedom in M(alpha,omega,DOF)
forceRAO_amp_mat = nan(Nalpha,Nomega,6);
forceRAO_phase_mat = nan(Nalpha,Nomega,6);
for k=1:6
forceRAO_amp_mat(:,:,k) = forceRAO_amp(1:Nomega,(k-1)*Nalpha+1:k*Nalpha)';
forceRAO_phase_mat(:,:,k) = forceRAO_phase(1:Nomega,(k-1)*Nalpha+1:k*Nalpha)';
end

%--- freqency integration
% cosine squared spread (only -1/2pi..1/2pi)
alphas_min_Mu = (alphas-Mu)-((alphas-Mu)>pi)*2*pi;%center around alphas-Mu
cos2 = 2/pi*cos(alphas_min_Mu).^2.*  ...
    (((alphas_min_Mu)<=pi/2)+((alphas_min_Mu)>=-pi/2)==2);%select only half
Sohmalpha = repmat(cos2(:),[1 Nomega]).*repmat(Sohm(:)',[Nalpha 1]);%
deltaomegas = [diff(omegas(:)')/2 0]+[0 diff(omegas(:)')/2];
```

```
53  deltaomegas([1 end]) = deltaomegas([1 end])*2; %frequency steps (non uniform)
54  % amplitudes zeta of frequencies omega in inertial directions alpha
55  zeta = (2*Sohmalpha.*repmat(deltaomegas(:)',[Nalpha 1])).^(1/2); %amplitudes
56
57  %-- surface elevation
58  %wave amplitude (surface elevation)
59  % zetaXt = sum(...
60  %          zeta(1,:).*cos(omegas.^2./g*xtilde(1) - omegas*t + rand_phase(1,:))+...
61  %          zeta(2,:).*cos(omegas.^2./g*xtilde(2) - omegas*t + rand_phase(2,:))+...
62  %          zeta(3,:).*cos(omegas.^2./g*xtilde(3) - omegas*t + rand_phase(3,:))+...
63  %          zeta(4,:).*cos(omegas.^2./g*xtilde(4) - omegas*t + rand_phase(4,:))+...
64  %          zeta(5,:).*cos(omegas.^2./g*xtilde(5) - omegas*t + rand_phase(5,:))+...
65  %          zeta(6,:).*cos(omegas.^2./g*xtilde(6) - omegas*t + rand_phase(6,:))+...
66  %          zeta(7,:).*cos(omegas.^2./g*xtilde(7) - omegas*t + rand_phase(7,:))+...
67  %          zeta(8,:).*cos(omegas.^2./g*xtilde(8) - omegas*t + rand_phase(8,:)) );
68  phases_I = repmat(omegas(:)',[Nalpha 1]).^2./g.*...
69        repmat(xtilde(:),[1 Nomega])...
70         -repmat(omegas(:)',[Nalpha 1])*t+rand_phase;
71  zetaXt = sum(sum( zeta.*cos(phases_I) ));
72
73  %--- rotate amplitude and phase (linearly interpolate) to boat frame
74  % wave period error with 45degree directions is max 1/cos(22.5/180*pi) 8.24%
75  diffalphas = diff(alphas(:)');
76  dalpha = diffalphas(1);%asume constant steps
77  leftindex = find([alphas(:)' 2*pi]>yaw,1,'first')-1;
78  leftindex = leftindex(1);%force scalar
79  if leftindex == 6;
80      rightindex = 1;
81  else
82      rightindex=leftindex+1;
83  end
84  % the first row (alpha_North) should be on the left index spot,circshift by
85  % left index-1 and the right index by rightindex-1 and scale both
86  % components according to the relative distance to the two components via
87  % dalpha
88  f = 1-(yaw-alphas(leftindex))/dalpha;
89  zeta_B = (f)*circshift(zeta,leftindex-1)+...
90           (1-f)*circshift(zeta,rightindex-1);
91  phases_B = (f)*circshift(phases_I,leftindex-1)+...
92             (1-f)*circshift(phases_I,rightindex-1);
93
94  %--- Wave forces
95  Tau_wave = zeros(9,1);
96  Tau_wave(4) = sum(sum(forceRAO_amp_mat(:,:,1).*zeta_B.*cos(phases_B+forceRAO_phase_mat(:,:,1)),2),1)
        ;
97  Tau_wave(5) = sum(sum(forceRAO_amp_mat(:,:,2).*zeta_B.*cos(phases_B+forceRAO_phase_mat(:,:,2)),2),1)
        ;
98  Tau_wave(6) = sum(sum(forceRAO_amp_mat(:,:,3).*zeta_B.*cos(phases_B+forceRAO_phase_mat(:,:,3)),2),1)
        ;
99  Tau_wave(7) = sum(sum(forceRAO_amp_mat(:,:,4).*zeta_B.*cos(phases_B+forceRAO_phase_mat(:,:,4)),2),1)
        ;
100 Tau_wave(8) = sum(sum(forceRAO_amp_mat(:,:,5).*zeta_B.*cos(phases_B+forceRAO_phase_mat(:,:,5)),2),1)
        ;
101 Tau_wave(9) = sum(sum(forceRAO_amp_mat(:,:,6).*zeta_B.*cos(phases_B+forceRAO_phase_mat(:,:,6)),2),1)
        ;
102
103 Tau_wave = Tau_wave(:);
```

C-19 funTauleg.m

```
1  function tau_leg = funTauleg(X,J_lx,f_leg,p_a,p_b2)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %%% funtion tau_leg = funTauleg(X,J_lx,f_leg,p_a,p_b2)
4  %%% part of MSc Thesis Wouter de Zeeuw
5  %%% TU Delft - GustoMSC 2011-2012
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  % function to calculate the forces from the hydraulics
8  % acting on the two bodies from the jacobian velocity transformation
9
10 %-- hydraulic connections
11 R_BtoP = [     cos(X(3)),          0,    -sin(X(3));
12       sin(X(2))*sin(X(3)),  cos(X(2)), cos(X(3))*sin(X(2));
13       cos(X(2))*sin(X(3)), -sin(X(2)), cos(X(2))*cos(X(3))];
14 Rz1 =  [cos(2*pi/3) sin(2*pi/3) 0;
15     -sin(2*pi/3) cos(2*pi/3) 0 ;
```

```
    0 0 1]';
Rz2 = [cos(4*pi/3) sin(4*pi/3) 0;
    -sin(4*pi/3) cos(4*pi/3) 0 ;
    0 0 1]';
a = [p_a  Rz1*p_a  Rz2*p_a]; %hydraulic connection top plate
b2 = [p_b2  Rz1*p_b2  Rz2*p_b2]; %hydraulic connection deck
a_B = R_BtoP'*a; %ship body frame definition of hydraulic connections

if size(J_lx,2)==6; J_lx_ok=J_lx';else J_lx_ok=J_lx;end%jacobian should be 6x3;
%--forces
f_1 = J_lx_ok(1:3,1)*f_leg(1);
f_2 = J_lx_ok(1:3,2)*f_leg(2);
f_3 = J_lx_ok(1:3,3)*f_leg(3);
tau_platform = -J_lx_ok*f_leg(:); %identical to hand calcuation! GOOD
f_platform = tau_platform(1:3); %identical! GOOD
f_boat = -f_platform;
%--moments
torque_platform = -skew(a_B(:,1))*f_1...
                        -skew(a_B(:,2))*f_2...
                        -skew(a_B(:,3))*f_3;%identical! GOOD
torque_boat = skew(b2(:,1))*f_1...
                    +skew(b2(:,2))*f_2...
                    +skew(b2(:,3))*f_3;
%--output
tau_leg = [f_platform(3); torque_platform(1:2); ...
        f_boat(:).*[0 0 1]'; torque_boat(:).*[1 1 0]'];
end
```

C-20 meandiff.m

```
function [meanerr,differr] = meandiff(err)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion [meanerr,differr] = meananddiff(err)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% err - 3x1 vector of legg errors, split in mean and difference from mean
mu=mean(err);
meanerr = mu*ones(3,1);
differr = ([err(1);err(2);err(3)]-mu);
```

C-21 setPos.m

```
function [Xset,HRP] = setPos(X,HRPset)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion [Xset,x_beta_I] = setPos(x_Beta_I)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculates from the present boat pose (eta) and the target inertial
% platform pose HRPset heave-roll-pitch the target state vector
% also returns the current inertial platform pose HRP vector
%-- input handling
eta = X(4:9); eta = eta(:);

%-- rotation from I to B
%%% only (3,3) element needed: angle between z^I and z^B axis is
%%% z^I_n \cdot z^B_n = |z^I_n||z^B_n|cos alpha =
%%% cos alpha = cos(X(7))*cos(X(8))
%%% so z^B = z^I/cos alpha
%%% inertial heave target
z_I_rel = HRPset(1)-eta(3);
Z_B_rel =z_I_rel/(cos(X(7))*cos(X(8)));%becomes inf when pitch or roll->pi/2!

%-- output Xset
Xset = NaN(9,1);
%%% heave
Xset(1) = Z_B_rel; %heave
%%% roll and pitch
Xset(2) = HRPset(2)-eta(4); %roll %approximation axis are not aligned
Xset(3) = HRPset(3)-eta(5); %pitch %approximation axis are not aligned
```

```
29 %%% inherit barge pose
30 Xset(4:9)=eta(:);
31
32 %--- output current x_beta_I
33 %%% rotation matrix I to Bframe
34 % R_ItoB =  [ cos(X(9))*cos(X(8)), cos(X(8))*sin(X(9)),-sin(X(8));
35 %              cos(X(9))*sin(X(7))*sin(X(8)) - cos(X(7))*sin(X(9)), ...
36 %              cos(X(7))*cos(X(9)) + sin(X(7))*sin(X(9))*sin(X(8)), ...
37 %              cos(X(8))*sin(X(7));
38 %              sin(X(7))*sin(X(9)) + cos(X(7))*cos(X(9))*sin(X(8)),...
39 %              cos(X(7))*sin(X(9))*sin(X(8)) - cos(X(9))*sin(X(7)), ...
40 %              cos(X(7))*cos(X(8))];
41 HRP = NaN(3,1);
42 %%% rotate heave and sum roll and pitch
43 HRP(1) = eta(3)+cos(X(7))*cos(X(8))*X(1);
44 HRP(2) = eta(4)+X(2); %approximation, alligned axi, true if theta_p=0
45 HRP(3) = eta(5)+X(3); %approximation, alligned axi, true if phi_b =0;
```

C-22 setleg.m

```
1  function ellset = setleg(t)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %%% funtion ellset = setleg(t)
4  %%% part of MSc Thesis Wouter de Zeeuw
5  %%% TU Delft - GustoMSC 2011-2012
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  % t - 1x1 current simulation time
8  % outputs a set profile for the target leg lenghts
9  % ellset - test function
10
11 ellset0 = [6;6;6];
12
13 if t>15&&t<30
14     ellset = ellset0+sin(t)*[1;-1;-1];
15 else if t>40
16         ellset = ellset0 +sin(t)*[0;-1;1];
17     else
18         ellset=ellset0;
19     end
20 end
21 end
```

C-23 funPlotwireframe.m

```
1  function funPlotwireframe(X,platform,vessel,fig1)%,t,Tend)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  %%% function funPlotwireframe(X,platform,vessel)
4  %%% part of MSc Thesis Wouter de Zeeuw
5  %%% TU Delft - GustoMSC 2011-2012
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  % plots wire frame of 2 bodies for given pose vector
8  % Example (uses simout block in simulink model):
9  % tic;tt=0; while toc+tt<simX.time(end);
10 % i=find(simX.time>toc+tt,1,'first');
11 % X = simX.signals.values(i,:);
12 % funPlotwireframe(X,platform,vessel,2);
13 % title(num2str(simX.time(i)));grid;
14 % xlabel N; ylabel -E;drawnow;end%
15 %
16 % due to the coordinate change the xyz direction correspond to the N (-)E
17 % and (-)D direction
18
19 figure(fig1);
20 % clf(fig1)
21 [az,el]=view;
22 %-- rotation and translations
23 R_BtoP = [  cos(X(3)),   0,    -sin(X(3));
24             sin(X(2))*sin(X(3)),  cos(X(2)), cos(X(3))*sin(X(2));
25             cos(X(2))*sin(X(3)), -sin(X(2)), cos(X(2))*cos(X(3))];
26 c = [0 0 X(1)]';
27 R_ItoB =  [ cos(X(9))*cos(X(8)), cos(X(8))*sin(X(9)),-sin(X(8));
```

```
28              cos(X(9))*sin(X(7))*sin(X(8)) - cos(X(7))*sin(X(9)), ...
29              cos(X(7))*cos(X(9)) + sin(X(7))*sin(X(9))*sin(X(8)), ...
30              cos(X(8))*sin(X(7));
31              sin(X(7))*sin(X(9)) + cos(X(7))*cos(X(9))*sin(X(8)),...
32              cos(X(7))*sin(X(9))*sin(X(8)) - cos(X(9))*sin(X(7)), ...
33              cos(X(7))*cos(X(8))];
34  NED = [X(4) X(5) X(6)]';
35  R_ItoBplanar = [  cos(X(9)), sin(X(9)), 0;
36                  -sin(X(9)), cos(X(9)), 0;
37                           0,        0, 1];
38  %-- box shaped boat
39  Xbox = [0 0 0;1 0 0; 1 1 0; 0 1 0; 0 0 1;1 0 1;1 1 1;0 1 1]';
40  scale = [vessel.main.Lpp vessel.main.B vessel.main.VCO]'; %boat size
41  Xbox = Xbox.*repmat(scale,[1 8]);
42  translate = [-scale(1)/2 -scale(2)/2 0]'; %to origin
43  Xbox = Xbox + repmat(translate,[1 8]);
44  scale = [vessel.main.Lpp vessel.main.B 0]';% waterline (planar approx)
45  Xwl = [0 0 0;1 0 0; 1 1 0; 0 1 0;0 0 0]';
46  Xwl =Xwl.*repmat(scale,[1 5]);
47  translate = [-scale(1)/2 -scale(2)/2 0]'; %to origin
48  Xwl = Xwl + repmat(translate,[1 5]);
49  %-- loading
50  Xboxload1 = [0 0 0;1 0 0; 1 1 0; 0 1 0; 0 0 1;1 0 1;1 1 1;0 1 1]';
51  scale = [30 30 20]'; %boat size
52  Xboxload1 = Xboxload1.*repmat(scale,[1 8]);
53  translate = [-scale(1)/2+30 -scale(2)/2 -scale(3)]'; %to origin
54  Xboxload1a = Xboxload1 + repmat(translate,[1 8]);
55  translate = [-scale(1)/2-30 -scale(2)/2 -scale(3)]'; %to origin
56  Xboxload1b = Xboxload1 + repmat(translate,[1 8]);
57  Xboxload2 = [0 0 0;1 0 0; 1 1 0; 0 1 0; 0 0 1;1 0 1;1 1 1;0 1 1]';
58  scale = [10 10 70]'; %boat size
59  Xboxload2 = Xboxload2.*repmat(scale,[1 8]);
60  translate = [-scale(1)/2 -scale(2)/2 -70]'; %to origin
61  Xboxload2 = Xboxload2 + repmat(translate,[1 8]);
62
63  %hexagon shaped platform
64  Xhex = [0 0 0;sqrt(3)/2 1/2 0; sqrt(3)/2 3/2 0;...
65          0 2 0; -sqrt(3)/2 3/2 0; -sqrt(3)/2 1/2 0;
66          0 0 2;sqrt(3)/2 1/2 2; sqrt(3)/2 3/2 2;...
67          0 2 2; -sqrt(3)/2 3/2 2; -sqrt(3)/2 1/2 2;
68          ]';
69  Xhex=Xhex/2;
70  scale = [25 25 3]'; %boat size
71  Xhex = Xhex.*repmat(scale,[1 12]);
72  translate = [0 -25/2 -3]'; %to origin
73  Xhex = Xhex + repmat(translate,[1 12]);
74
75  %move bodies according to X
76  Xbox = (R_ItoB'*Xbox);
77  Xbox = Xbox + repmat(NED,[1 8]);
78  Xbox = Xbox.*repmat([1 -1 -1]',[1 8]);
79  Xwl = (R_ItoBplanar'*Xwl);
80  Xwl = Xwl + repmat([NED(1:2); 0],[1 5]);
81  Xwl = Xwl.*repmat([1 -1 -1]',[1 5]);
82  Xboxload1a = (R_ItoB'*Xboxload1a);
83  Xboxload1a = Xboxload1a + repmat(NED,[1 8]);
84  Xboxload1a = Xboxload1a.*repmat([1 -1 -1]',[1 8]);
85  Xboxload1b = (R_ItoB'*Xboxload1b);
86  Xboxload1b = Xboxload1b + repmat(NED,[1 8]);
87  Xboxload1b = Xboxload1b.*repmat([1 -1 -1]',[1 8]);
88  Xhex = R_BtoP'*R_ItoB'*Xhex;
89  Xhex = Xhex + repmat(NED,[1 12]) + repmat(R_ItoB'*c,[1 12]);
90  Xhex = Xhex.*repmat([1 -1 -1]',[1 12]);
91  Xboxload2 = R_BtoP'*R_ItoB'*Xboxload2;
92  Xboxload2 = Xboxload2 + repmat(NED,[1 8]) + repmat(R_ItoB'*c,[1 8]);
93  Xboxload2 = Xboxload2.*repmat([1 -1 -1]',[1 8]);
94
95
96  %-- plot
97  Xboxline = Xbox(:,[1 2 4 3 1 5 6 2 3 7 8 5 7 6 8 4 1]);
98  Xboxload1aline = Xboxload1a(:,[1 2 4 3 1 5 6 2 3 7 8 5 7 6 8 4 1]);
99  Xboxload1bline = Xboxload1b(:,[1 2 4 3 1 5 6 2 3 7 8 5 7 6 8 4 1]);
100 Xboxload2line = Xboxload2(:,[1 2 4 3 1 5 6 2 3 7 8 5 7 6 8 4 1]);
101 Xhexline = Xhex(:,[1 2 3 4 5 6 7 2 9 4 11 6 7 8 3 10 5 12 1 8 9 10 11 12 7 6 1]);
102  hold off %delete for recording movie
103 plot3(Xboxline(1,:),Xboxline(2,:),Xboxline(3,:),'k','linewidth',1);
```

```
hold on
plot3(Xhexline(1,:),Xhexline(2,:),Xhexline(3,:),'r','linewidth',1);
plot3(Xboxload1aline(1,:),Xboxload1aline(2,:),Xboxload1aline(3,:),'k','linewidth',1);
plot3(Xboxload1bline(1,:),Xboxload1bline(2,:),Xboxload1bline(3,:),'k','linewidth',1);
plot3(Xboxload2line(1,:),Xboxload2line(2,:),Xboxload2line(3,:),'r','linewidth',1);


%--- add legs
[ell,ln,b2s,as] =funKinv(X,platform.b1,platform.b2,platform.a,platform.alpha,platform.normd1,
    platform.normd2);
b2sI = R_ItoB'*b2s;
assI = R_BtoP'*R_ItoB'*as;
lnI = (R_ItoB'*ln).*repmat(ell',[3 1]);
c_B = [1 -1 -1]'.*(R_ItoB'*c);
for k=1:3;plot3([b2sI(1,k) b2sI(1,k)+lnI(1,k)]+NED(1),...
        -[b2sI(2,k) b2sI(2,k)+lnI(2,k)]-NED(2),...
        -[b2sI(3,k) b2sI(3,k)+lnI(3,k)]-NED(3),'k','linewidth',2);
    plot3([0 b2sI(1,k)]+NED(1),...
        -[0 b2sI(2,k)]-NED(2),...
        -[0 b2sI(3,k)]-NED(3),'k','linewidth',1);
    plot3([0 assI(1,k)]+NED(1)+c_B(1),...
        -[0 assI(2,k)]-NED(2)+c_B(2),...
        -[0 assI(3,k)]-NED(3)+c_B(3),'r','linewidth',1);
end
surf([-60;60],[-60;60],zeros(2),'EdgeColor','none','FaceColor',[199/256 217/256 252/256])
 alpha(.1)
plot3(Xwl(1,:),Xwl(2,:),Xwl(3,:),'b','linewidth',1);

% grid
axis equal
zlim([-10 35])
% az = 45; el=30; %fix view
% az = 120; el=44; %fix view
% az = 360*t/Tend; el=44*t/Tend; %fix view
view(az,el)
```

C-24 funMovieWireframe.m

```
function mov = funMovieWireframe(simX,platform,vessel,filename,fps)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% funtion funMovieWireframe(simX,platform,vessel,filename)
%%% part of MSc Thesis Wouter de Zeeuw
%%% TU Delft - GustoMSC 2011-2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

vidObj = VideoWriter([filename '.avi']);
vidObj.FrameRate = fps;
open(vidObj);
scrsz = get(0,'ScreenSize');%make the frame size fixed
% fig1=figure('Position',[1 1 scrsz(3) scrsz(4)]);%full screen res
fig1=figure('Position',[1 1 scrsz(3)/2 scrsz(4)/2]);%quarter screen
set(fig1,'Renderer','zbuffer')%for getframe to work
T = [0:0.1:simX.time(end)];

cnt = 1;
for t=T;
    currFrame = struct('cdata', [],'colormap', []);
    i=find(simX.time>=t,1,'first');
    if ~isempty(i)
        X = simX.signals.values(i,:);
        funPlotwireframe(X,platform,vessel,fig1,t,simX.time(end));
        title(['Platform on Vessel simulation (wireframe), t = '...
            num2str(.1*round(simX.time(i)*10),'%1.1f')]);
        set(fig1, 'color', 'white')
        drawnow;
        %movie record
    end
    currFrame=getframe(fig1);
    if isempty(currFrame.cdata)
        pause(0.3)
        currFrame=getframe(fig1);
    end
    writeVideo(vidObj,currFrame);
    cnt = cnt+1;
```

```
37 end
38 close(vidObj)
```

C-25 setsurfaceelevation.m

```
1  function setsurfaceelevation(surfelevXt)
2  %#codegen
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  %%% funtion setsurfaceelevation(surfelevXt)
5  %%% part of MSc Thesis Wouter de Zeeuw
6  %%% TU Delft - GustoMSC 2011-2012
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  % sets the heights of the surface node in the VR simulation via extrinic
9  % matlab functions (VR Sink does not allow these manupulations directly)
10 coder.extrinsic('get_param')
11 coder.extrinsic('vrworld')
12 coder.extrinsic('vrnode')
13 coder.extrinsic('getfield')
14 coder.extrinsic('setfield')
15 % Get vrworld associated with the VR Sink found there
16 hw=vrworld(get_param('shipplatform/VR Visualisation/VR Sink', 'WorldFileName'));
17 % Handle to the Sea elevation map node we want to control
18 hn1=vrnode(hw,'Sea');
19 hn2=getfield(hn1,'geometry');
20 % surfelevcurrent = getfield(hn2,'height');
21 % surfelevXt = randn(900,1);
22 setfield(hn2,'height',surfelevXt)
```

C-26 funMPC.m

```
1  function [u_star_t0,du_star_vec_output,Jp_star,alpha_subopt_output,X_p_i,HRP_out] = ...
2      funMPC(Xhat,HRPset,Vt0,u_minmax,mpc_N_set,mpc_dt,mpc_maxitt,...
3      u_tmin1,du_star_vecmin1,M_RBsJAinfJ,M_RBp,p_b1,p_b2,p_a,...
4      p_alpha,p_normd1,p_normd2,G,VCO,flag_mooring,m_p,m_s,g,r_opgp,r_obgs)
5  %#codegen
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  %%% function [u_star_t0,du_star_vec,Jp_star,alpha_subopt] = ...
8  %%%      funMPC(Xhat,HRPset,Vt0,u_minmax,mpc_N,mpc_dt,mpc_maxitt,...
9  %%%      u_tmin1,u_star_vecmin1,M_RBsJAinfJ,M_RBp,p_b1,p_b2,p_a,...
10 %%%      p_alpha,p_normd1,p_normd2,G,VCO,flag_mooring,m_p,m_s,g,r_opgp,r_obgs)
11 %%% part of MSc Thesis Wouter de Zeeuw
12 %%% TU Delft - GustoMSC 2011-2012
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 % Model Predictive Controller for the platform on the ship
15 % the prediction model includes:
16 % * Pose dependent Mass matrix M
17 % * Coriolis forces due to moving refererence frame (base is on the deck)
18 % * Nonlinear kinematic relations of linkages
19 % * Linear hydrostatics
20 % * Pose dependent gravity
21 % * quasi velocity - pose transformation
22 %
23 % Random external effects are ommited hence the solution is a wave free
24 % (bathtub) solution. Wave radiation is omitted for speed of excecution.
25 %
26 % The components of the model are the function files borrowed from the
27 % designed blocks in the shipplatform.mdl model (all unaltered but the
28 % inverse kinematics, which is set to have no error control as this could
29 % compromise feasibility)
30 % M = funM(M_RBsJAinfJ,M_RBp,X)
31 % C = funC(M,M_RBp,V,X)
32 % [~,~,~,~,~,~,~,~,~,J_lx] = funKinv(X,p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2)
33 % tau_leg = funTauleg(X,J_lx,f_leg,p_a,p_b2)
34 % tau_hydrostat = funTauhydrostat(X,G,~,VCO,flag_mooring)
35 % tau_grav = funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs)
36 % [~,J_xv] = funXV(V,X)
37 %
38 % The degree of suboptimality is found for a an horizon of N-1 (this adds
39 % no additional cost to the calculation, but is conservative)
40 % J_star_N(x0) is the optimal J for the full horizon, now calculate
41 % J_star_N-1(x0) and J_star_N1(x1) from the sequence of optimal stage costs and
```

```
% devide ths value by the first stage cost l(x0) to find alpha_subopt
%
% Inputs:
% MPC specific:
% Xhat  Current state (estimated by forward kinematics)
% HRPset setpoint heave roll pitch
% Vt0 Current velocity
% u_minmax constraint (soft) on control level
% mpc_N prediction horizon (steps)
% mpc_dt time steps
% mpc_maxitt  maximum iterations of optimizer
% u_tmin1 previous control level (of previous call to funMPC(.) use delayblk
% u_star_vecmin1 previous optimal control sequecne (hot starting)
% SUBfunctions: (referenced in subfunctions)
% M_RBsJAinfJ,M_RBp,p_b1,p_b2,p_a,p_alpha,p_normd1,...
% p_normd2,G,VCO,flag_mooring,m_p,m_s,g,r_opgp,r_obgs
%
% Output:
% u_star_t0  control signal 3x1
% du_star_vec vector of optimal delta control signals (mpc_N*3 x 1)
% Jp_star performance objective value
% alpha_subopt suboptimalty (1 is good, 0 is poor, negative is infeasible)
%              calculated (conservative) for N-1 horizon

%-- input handling
%%% initial (hot start) delta control signal
% du0 = [du_star_vecmin1((1+3):end);zeros(3,1)];
State_t0 = [Vt0;Xhat(:)];
du0 = du_star_vecmin1(:);
du0 = [du0(4:end); zeros(3,1)];
% du0 = -.5*10^6*ones(mpc_N*3,1);

%-- optimizing NELDER MEAD SIMPLEX
% mpc_dt = 0.5;
% mpc_N = 5;
% mpc_maxitt = 105;
% reqmin = 1e-6; %convergence requirement
% step = -.5e6*ones(mpc_N*3,1);%size of initial simplex
% konvge = 50; %check every itt for convergence mpc_maxitt
% [ du_star_vec_out, Jp_star,icount, numres, ifault ]= nelminfunPerf ( mpc_N*3, du0, ...
%   reqmin, step, konvge,mpc_maxitt,  mpc_N,mpc_dt,u_tmin1,State_t0,...
%     M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
%     p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
%     HRPset,u_minmax);

%-- optimizing GAUSS NEWTON
mpc_dt = 0.5;
mpc_N = 5;
tol = 10^-1;
mpc_maxitt = 10;
lambda_GN = .2;
[du_star_vec_out,Jp_star,histout,costdata] = gaussn(du0,tol,mpc_maxitt,...
    lambda_GN,mpc_N,mpc_dt,u_tmin1,State_t0,...
    M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
    p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
    HRPset,u_minmax);

du_star_vec_output = du_star_vec_out(1:mpc_N*3,1); %FORCED size for simulink
%-- output
HRP_out = NaN(3,1);
%%% rotate heave and sum roll and pitch
HRP_out(1) = Xhat(6)+cos(Xhat(7))*cos(Xhat(8))*Xhat(1);
HRP_out(2) = Xhat(7)+Xhat(2); %approximation, alligned axi, true if theta_p=0
HRP_out(3) = Xhat(8)+Xhat(3); %approximation, alligned axi, true if phi_b =0;
u_star_t0 = u_tmin1+du_star_vec_out(1:3); %next control signal
[~,alpha_subopt,X_p_i]=funPerf(du_star_vec_out,mpc_N,mpc_dt,u_tmin1,State_t0,...
    M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
    p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
    HRPset,u_minmax);%optimal sequence
alpha_subopt_output = alpha_subopt(1);
end %end MPC controller main, below are the used components borrowed from bocks

function dState_p = funAcc(State_p,u_p,...
   M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
   p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2)
% Part of funMPC() WA de Zeeuw
```

```
% >> prediction model
% Solution of the system acceleration equations
% input quasi velocity and pose,
% output quasi acceleration and pose rate
%--- matrices
V = State_p(1:9);
X = State_p(10:18);
%%% mass matrix
M = funM(M_RBsJAinfJ,M_RBp,X);
%%% coriolis matrix
C = funC(M,M_RBp,V,X);
%%% gravity
tau_grav = funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs);
%%% hydrostatics
tau_hydrostat = funTauhydrostat(X,G,[],VCO,flag_mooring);
%%% control force
% Jacobian
[~,~,~,~,~,~,~,~,~,J_lx] = funKinv(X,p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2);
tau_leg = funTauleg(X,J_lx,u_p,p_a,p_b2);
%%% pose rate transformation
[~,J_xv] = funXV(V,X);
%--- state transition with backslash operator

dState_p = [M zeros(9);zeros(9) eye(9)]\ ...
    ( [-C zeros(9);J_xv zeros(9)]*State_p + ...
    [(tau_grav+tau_hydrostat+tau_leg);zeros(9,1)]  );
%=exactly the acceleration of the sim model

end %f()


function l = funl(X_p,HRPset,du,u,u_minmax)
% Part of funMPC() WA de Zeeuw
% >> calculation of stage cost
HRP = NaN(3,1);
%%% rotate heave and sum roll and pitch
HRP(1) = X_p(6)+cos(X_p(7))*cos(X_p(8))*X_p(1);
HRP(2) = X_p(7)+X_p(2); %approximation, alligned axi, true if theta_p=0
HRP(3) = X_p(8)+X_p(3); %approximation, alligned axi, true if phi_b =0;
HRPerr = HRP-HRPset(:);
%%% controller soft penalty on excess control force (overload)
lambda_PHI = .1;
Phi = lambda_PHI*sum(abs(min(u-(u_minmax(1)),0))+abs(max(u-(u_minmax(2)),0)));
%%% quadratic performance index
% P = diag([100 5e4 5e4]); %penalty output
P = 10*diag([100 5e4 5e4]); %penalty output
du_MN = du*10^-6;
lambda_dU=1;
Q = eye(3); %control u penalty
l = HRPerr(:)'*P*HRPerr(:) + lambda_dU*du_MN(:)'*Q*du_MN(:)+Phi; %stage cost
end %funl()

function [Jp,alpha_subopt,X_p_i] = funPerf(du_vec,mpc_N,mpc_dt,u_tmin1,State_t0,...
    M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
    p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
    HRPset,u_minmax)
% Part of funMPC() WA de Zeeuw
% >> calculation performance value (objective
State_i = State_t0(:); %initialize state vector
%% Performance index (sum of cost functions)
l1 = NaN(1);Jp = NaN;lN = NaN; %bypass debugger scripts by preassinging 2x
X_p_i = NaN(mpc_N,9);
for i=1:mpc_N %sum over prediction horizon
    % prepare control signal for state transition and stage cost
    du_i = du_vec(1+(i-1)*3:3+(i-1)*3); %current deltau
    dU = reshape(du_vec(:),[3 mpc_N]);
    u_i = u_tmin1+sum(dU(1:3,1:i),2); %current control level
    % Euler forward proved to be way to instable
    %fixed stepsize RK4, assumed constant control force
    h = mpc_dt;
    k1 = funAcc(State_i,u_i,...
    M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
    p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2) ;
    k2 = funAcc(State_i+k1*h/2,u_i,...
    M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
    p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2) ;
```

```
194      k3 = funAcc(State_i+k2*h/2,u_i,...
195      M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
196      p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2) ;
197      k4 = funAcc(State_i+k3*h,u_i,...
198      M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
199      p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2) ;
200      DeltaState_i = h/6*(k1+2*k2+2*k3+k4);
201      State_iplus1 = State_i+DeltaState_i; %RK4 step
202      %%% stage cost
203      X_p = State_iplus1(10:18); %predicted state
204      X_p_i(i,:)= X_p(:)';
205      l = funl(X_p,HRPset,du_i,u_i,u_minmax); %stage cost
206      if i == 1; %store first stage cost (for calc of alpha)
207          l1 = l;
208          Jp = l1; %initialize Jp
209      else
210          Jp = Jp+l;
211      end
212      if i == mpc_N;
213          lN = l; %store last cost (for calc of alpha)
214      end
215  State_i = State_iplus1;
216  end %for prediciton horizon loop
217  %%% degree of suboptimality
218  Jp_x0_Nmin1 = Jp - lN; %one shorter horizon from first x
219  Jp_x1_Nmin1 = Jp - l1;%one shorter horizon from second x
220  alpha_subopt = (Jp_x0_Nmin1-Jp_x1_Nmin1)/l1;
221  end%funPerformance
222
223
224  %%%%=========================================================%%%%%
225  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
226  %%%%%%%%%  REUSED CODE from shiplaform.mdl  %%%%%%%%%%%%%%%
227  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
228  % M = funM(M_RBsJAinfJ,M_RBp,X)
229  % C = funC(M,M_RBp,V,X)
230  % [~,J_lx] = funKinv(X,p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2)
231  % tau_leg = funTauleg(X,J_lx,f_leg,p_a,p_b2)
232  % tau_hydrostat = funTauhydrostat(X,G,~,VCO,flag_mooring)
233  % tau_grav = funTaugrav(X,m_p,m_s,g,r_opgp,r_obgs)
234  % [~,J_xv] = funXV(V,X)
235  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
236  %%%  REPEATED BELOW  IN IMPLEMENTATION, NOT IN REPORT  %%%
237  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
238  %%%%=========================================================%%%%%
239  % % change fun to  funPerf( pstar ,mpc_N,mpc_dt,u_tmin1,State_t0,...
240  %      M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
241  %      p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
242  %      HRPset,u_minmax);
243  %%% adapted from maxlike code by wouter de zeeuw
244  function [dfr,JAC,base] = numdiff(X,h_N,mpc_N,mpc_dt,u_tmin1,State_t0,...
245      M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
246      p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
247      HRPset,u_minmax)
248  %mpc_N*3+1 fevaluations
249  if nargin < 2
250      h_N  = 1e-8;
251  end
252  df = NaN*X;
253  % oldObjFuncValue = fun(X);
254  oldObjFuncValue = funPerf( X ,mpc_N,mpc_dt,u_tmin1,State_t0,...
255      M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
256      p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
257      HRPset,u_minmax);
258  for i = 1:numel(X)
259      X_new    = X;
260      X_new(i) = X_new(i) + h_N;
261  %    newObjFuncValue = fun(X_new);
262      newObjFuncValue =  funPerf(X_new,mpc_N,mpc_dt,u_tmin1,State_t0,...
263      M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
264      p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
265      HRPset,u_minmax);
266      df(i) = (newObjFuncValue-oldObjFuncValue)/h_N;
267  end
268  df = df(:);
269  dfr = df*oldObjFuncValue;
```

```
270 % JAC = df'; %scalar valued!
271 JAC = df; %scalar valued!
272 base = oldObjFuncValue;
273 end
274 %%%%%%%=================================================%%%%%%%%%%%%%%
275
276 function [x,fc,histout,costdata] = gaussn(x0,tol,maxit,lambda,mpc_N,mpc_dt,u_tmin1,State_t0,...
277     M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
278     p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
279     HRPset,u_minmax)
280 %%% adapted to use no handles! and to separatly approximate the jacobian
281 %%% (=gradient for scalar objective)
282 %%% source:http://www4.ncsu.edu/~ctk/darts/gaussn.m
283 % C. T. Kelley, Dec 14, 1997
284 %
285 % This code comes with no guarantee or warranty of any kind.
286 %
287 % function [x,histout,costdata] = gaussn(x0,f)
288 %
289 % Damped Gauss-Newton with Armijo rule
290 % simple divide by 2 stepsize reduction
291 %
292 % Input: x0 = initial iterate
293 %        f = r^T r/2 = objective function,
294 %            the calling sequence for f should be
295 %            [fout,gout,jac]=f(x) where fout=f(x) is a scalar
296 %              gout = jac^T r = grad f(x) is a COLUMN vector
297 %              and jac = r' = Jacobian of r is an M x N matrix
298 %        tol = termination criterion norm(grad) < tol
299 %        maxit = maximum iterations (optional) default = 100
300 %
301 % Output: x = solution
302 %          histout = iteration history
303 %               Each row of histout is
304 %       [norm(grad), f, number of step length reductions, iteration count]
305 %          costdata = [num f, num grad, num hess] (for gaussn, num hess=0)
306 %
307 % At this stage all iteration parameters are hardwired in the code.
308 %
309 %
310 h_numdiff = 10^-2;
311 alp=1.d-4;
312 if nargin < 3
313     maxit=100;
314 end
315 itc=1; xc=x0;
316 fc = funPerf( xc ,mpc_N,mpc_dt,u_tmin1,State_t0,...
317     M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
318     p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
319     HRPset,u_minmax);
320 [gc,jac] = numdiff(xc,h_numdiff,mpc_N,mpc_dt,u_tmin1,State_t0,...
321     M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
322     p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
323     HRPset,u_minmax);
324 numf=1+numel(xc); numg=1; numh=0;%numerical computation of jacobian feval
325 % ithist=zeros(1,4);
326 ithist = nan(maxit+1,4);%preassign
327 ithist(1,1)=norm(gc); ithist(1,2) = fc; ithist(1,4)=itc-1; ithist(1,3)=0;
328 xtry = nan(maxit+1,numel(xc));
329 xtry(1,:) = xc(:)';
330 while(norm(gc) > tol && itc <= maxit)
331     itc=itc+1;
332     dc=(jac'*jac)\gc;
333     xt=xc-lambda*dc;
334     xc=xt;
335     %numdiff also returns the base
336     [gc,jac,fc] = numdiff(xc,h_numdiff,mpc_N,mpc_dt,u_tmin1,State_t0,...
337         M_RBsJAinfJ,M_RBp,m_p,m_s,g,r_opgp,r_obgs,G,VCO,flag_mooring,...
338         p_b1,p_b2,p_a,p_alpha,p_normd1,p_normd2,...
339         HRPset,u_minmax);
340     numf=numf+1+numel(xc); numg=numg+1;
341     ithist(itc,1)=norm(gc); ithist(itc,2) = fc;
342     ithist(itc,4)=itc-1;
343     xtry(itc,:) = xc(:)';
344 end
345 [fc,idxmin] = min(ithist(:,2));
```

```
346 x=xtry(idxmin,:)'; histout=ithist(1:itc,:); %robustify to only output best of tests
347 costdata=[numf, numg, numh];
348 end
```

Bibliography

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. "Survey Paper - Constrained model predictive control: Stability and optimality". In: *Automatica* 36 (2000), pp. 789–814.

[2] V. R. Ravi, T. Thyagarajan, and M. M. Darshini. "A Multiple Model Adaptive Control Strategy for Model Predictive Controller for Interacting Non Linear Systems". In: *Proc. Int. Process Automation, Control and Computing Conf. PACC*. 2011, pp. 1–8.

[3] T. v. d. Boom. *Model Predictive Control - Course Notes SC4060*. TU Delft, 2008.

[4] K. Åström and R. Murray. *Feedback Systems*. Princeton Univeristy Press, 2008.

[5] S. Qin and T. A. Badgwell. "A survey of industrial model predictive control technology". In: *Control Engineering Practice* 11.7 (2003), pp. 733–764.

[6] K. Åström and T. Hägglund. "The future of PID control". In: *Control Engineering Practice* 9.11 (2001), pp. 1163–1175.

[7] L. A. Sandino, M. Bejar, and A. Ollero. "On the applicability of linear control techniques for autonomous landing of helicopters on the deck of a ship". In: *Proc. IEEE Int. Mechatronics Conf. ICM*. 2011, pp. 363–368.

[8] X.-S. Ge, W.-J. Zhao, and Y.-Z. Liu. "Nonholonomic Motion Planning for a Free-Falling Cat Using Quasi-Newton Method". In: *Technische Mechanik* 1 (2011), pp. 42–49.

[9] R. Findeisen and F. Allgöwer. "An introduction to nonlinear model predictive control". In: *21st Benelux Meeting on Systems and Control*. Vol. 11. 2002.

[10] M. A. Henson. "Nonlinear model predictive control: current status and future directions". In: *Computers and Chemical Engineering* 23 (1998), pp. 187–202.

[11] T. Schei and T. Johansen. "Nonlinear model based/model predictive control with constraints and with/without nonlinear observer". In: *SINTEF Electronics and Cybernetics, Automatic Control*. 1998.

[12] X. Wei. "Application of a Semi-Analytical Method to Model Predictive Control". PhD thesis. Tampere University of Technology, 2007.

[13] K. Muske and J. Rawlings. "Model predictive control with linear models". In: *AIChE Journal* 39.2 (1993), pp. 262–287.

[14] H. Chen and F. Allgöwer. "Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability". In: *Automatica* 34.10 (1998), pp. 1205–1217.

[15] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer, 2011.

[16] L. Grüne and J. Pannek. "Practical NMPC suboptimality estimates along trajectories". In: *Systems & Control Letters* 58.3 (2009), pp. 161–168.

[17] Y. Wang and S. Boyd. "Fast Model Predictive Control Using Online Optimization". In: *Proceedings of the 17th World Congress The International Federation of Automatic Control*. 2008.

[18] L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini. "A stabilizing model-based predictive control algorithm for nonlinear systems". In: *Automatica* 37.9 (2001), pp. 1351–1362.

[19] M. Alamir, N. Sheibat-Othman, S. Othman, et al. "Constrained nonlinear receding-horizon control for maximizing production in polymerization processes." In: *IEEE Transaction on Control Systems Technology*. 2006.

[20] D. Zhiying and W. Xianfang. "Nonlinear Generalized Predictive Control Based on Online SVR". In: *Proc. Second Int. Symp. Intelligent Information Technology Application IITA '08*. Vol. 2. 2008, pp. 1105–1109.

[21] H. Zhao, J. Cao, Z. Li, and Y. Liu. "Nonlinear predictive functional control of recursive subspace model using support vector machine". In: *Proc. Chinese Control and Decision Conf. CCDC*. 2008, pp. 4909–4913.

[22] P. Overschee, B. Moor, D. Hensher, J. Rose, W. Greene, K. Train, W. Greene, E. Krause, J. Gere, and R. Hibbeler. *Subspace Identification for the Linear Systems: Theory–Implementation*. Kluwer academic publishers, 1996.

[23] R. Khan, P. Williams, R. Hill, and C. Bil. "Fault tolerant flight control system design for UAV's using Nonlinear Model Predictive Control". In: *Proc. Australian Control Conf. AUCC*. 2011, pp. 297–302.

[24] L. Cheng, C. sheng Jiang, M. Chen, and M. Pu. "Online-SVR-based GPC control for airframe/engine integrated near-space hypersonic vehicle". In: *Proc. 8th Asian Control Conf. ASCC*. 2011, pp. 682–687.

[25] M. Aminzadeh, A. Mahmoodi, and M. Sabzehparavar. "Dynamic analysis of a 3DoF Motion Platform". In: *International Journal of Robotics: Theory and Application* 1.1 (2009), pp. 12–18.

[26] R. Hedjar and P. Boucher. "Nonlinear Receding-Horizon Control of Rigid Link Robot Manipulators". In: *International Journal of Advanced Robotic Systems* 2 (2005), pp. 15–24.

[27] M. Aminzadeh and M. Sabzehparvar. "Model-based motion tracking control of an electric 3DoF parallel motion platform". In: *Proc. IEEE Aerospace Conf.* 2010, pp. 1–8.

[28] S. H. Koekebakker. "Model based control of a flight simulator motion system". PhD thesis. Technische Universiteit Delft, 2001.

[29] E. S. Nadimi, T. Bak, and R. Izadi-Zamanabadi. "Model predictive controller combined with LQG controller and velocity feedback to control the stewart platform". In: *Proc. 9th IEEE Int Advanced Motion Control Workshop*. 2006, pp. 44–49.

[30] W. Zhang, K. Zheng, H. Cai, X. Bian, and X. Shi. "Predictive control for submergence rescue vehicle station and attitude maneuver". In: *Proc. IEEE Int. Conf. Automation and Logistics ICAL 2008*. 2008, pp. 2516–2521.

[31] A. Khan, C. Bil, and K. E. Marion. "Ship motion prediction for launch and recovery of air vehicles". In: *Proc. MTS/IEEE OCEANS*. 2005, pp. 2795–2801.

[32] K. Kosuge, H. Kajita, and T. Fukuda. "Force control of manipulator/vehicle system floating on the water utilizing vehicle restoring force". In: *Proc. Workshop 4th Int. Advanced Motion Control AMC '96-MIE*. Vol. 2. 1996, pp. 506–511.

[33] H. Kajita, J. Imamura, and K. Kosuge. "Advanced control of manipulator/vehicle system floating on the water". In: *Proc. 24th Annual Conf. of the IEEE Industrial Electronics Society IECON '98*. Vol. 4. 1998, pp. 2397–2402.

[34] B. Kimiaghalam, A. Ahmadzadeh, A. Homaifar, and B. Sayarrodsari. "A purely model predictive control for a marginally stable system". In: *Proc. American Control Conf. 2003*. Vol. 5. 2003, pp. 4293–4298.

[35] P. J. From, V. Duindam, J. T. Gravdahl, and S. Sastry. "Modeling and motion planning for mechanisms on a non-inertial base". In: *Proc. IEEE Int. Conf. Robotics and Automation ICRA '09*. 2009, pp. 3320–3326.

[36] P. J. From, J. T. Gravdahl, T. Lillehagen, and P. Abbeel. "Motion planning and control of robotic manipulators on seaborne platforms". In: *Control Engineering Practice* 19.8 (2011), pp. 809–819.

[37] L. J. Love, J. F. Jansen, and F. G. Pin. "On the modeling of robots operating on ships". In: *Proc. IEEE Int. Conf. Robotics and Automation ICRA '04*. Vol. 3. 2004, pp. 2436–2443.

[38] P. From, I. Schjølberg, J. Gravdahl, K. Pettersen, and T. Fossen. "On the Boundedness and Skew-Symmetric Properties of the Inertia and Coriolis Matrices for Vehicle-Manipulator Systems". In: *Proc. of the 7th IFAC Symp. on Intelligent Autonomous Vehicles*. 2010.

[39] J. Journée and W. Massie. *Offshore Hydromechanics*. Delft University of Technology, 2001.

[40] T. Perez and T. Fossen. "Kinematic models for seakeeping and manoeuvring of marine vessels". In: *Modeling, Identification and Control* 28.1 (2007), p. 1.

[41] W. Cummins. "The Impulse Response Function and Ship Motions". In: *Schiffstechnik* 9 (1962), pp. 101–109.

[42] T. Ogilvie. "Recent progress toward the understanding and prediction of ship motions". In: *5th Symposium on naval hydrodynamics*. Bergen, Norway. 1964, pp. 3–80.

[43] T. I. Fossen. "A nonlinear unified state-space model for ship maneuvering and control in a seaway". In: *International journal of bifurcation and chaos in applied sciences and engineering* 15.9 (2005), p. 2717.

[44] M. Triantafyllou and F. Hover. *Maneuvering and control of marine vehicles*. Department of Ocean Engineering, Massachussets Institute of Technology, Cambridge, USA, 2003.

[45] E. Kristiansen and O. Egeland. "Frequency-dependent added mass in models for controller design for wave motion damping". In: *IFAC Conf. on Maneuvering and Control of Marine Systems MCMC '03*. 2003.

[46] E. Kristiansen, Åsmund Hjulstad, and O. Egeland. "State-space representation of radiation forces in time-domain vessel models". In: *Ocean Engineering* 32.17-18 (2005), pp. 2195–2216.

[47] K. Unneland. "Identification and Order Reduction of Radiation Force Models of Marine Structures". PhD thesis. Norwegian University of Science and Technology, 2007.

[48] T. Pérez and T. Fossen. "Time-vs. frequency-domain identification of parametric radiation force models for marine structures at zero speed". In: *Modeling, Identification and Control* 29.1 (2008), pp. 1–19.

[49] T. Perez and T. Fossen. "A matlab toolbox for parametric identification of radiation-force models of ships and offshore structures". In: *Modeling, Identification and Control* 30.1 (2009), pp. 1–15.

[50] H. E. Frick and D. J. Mottram. "Open sea transfer of articles". Pat. 4,854,800. Aug. 1989.

[51] L. D. Stair. "System to transfer cargo or passenger between platforms while undergoing releative motion". Pat. 4180362. Dec. 1979.

[52] J. van der Tempel, D. J. Cerda Salzmann, J. Koch, F. Gerner, and A. J. Gobel. "Vessel, motion platform, method for compensating motions of a vessel and use of a stewart platform". Pat. 0032543. Feb. 2010.

[53] P. M. Koppert. "Motion compensation device for compensating a carrier frame on a vessel for water motion". Pat. 0024214. Feb. 2012.

[54] V. Hovland and A. Vatn. "Helicopter landing platform having motion stabilizer for compensating ship roll and/or pitch". Pat. 0224118. Sept. 2010.

[55] S. Leske. "Device for the safe transfer of personnel or material from an object configured as a boat to an object moving relative thereto". Pat. 003891A1. Feb. 2011.

[56] T. E. Powell. "Motion Compensated Apparatus". Pat. 5822813. Oct. 1998.

[57] D. Cerda Salzmann. "Ampelmann - Development of the Access System for Offshore Wind Turbines". PhD thesis. Technische Universiteit Delft, 2010.

[58] M. Otten and W. van Berkum. *Bachelor Eindproject: Resonantie van de Ampelmann*. Tech. rep. Technische Univeristeit Delft, 2008.

[59] J. van der Tempel, D. C. Salzmann, T. Mulder, J. Koch, F. Gerner, O. Calkoen, A. Göbel, H. Brinkhuis, R. Lagers, and W. van Korven. "Scale model testing of the Ampelmann". In: *European Wind Energy Conference 2004*. 2004.

[60] W. A. de Zeeuw. *Ship motion compensation platform for high payloads - Technical Report Internship performed at GustoMSC*. Tech. rep. TU Delft/GustoMSC, 2011.

[61] M. Jun and M. Safonov. "Stability analysis of a system with time-delayed states". In: *American Control Conference, 2000. Proceedings of the 2000*. Vol. 2. IEEE. 2000, pp. 949–952.

[62] M. Dekking. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Verlag, 2005.

[63] C. Heij, P. de Boer, P. Franses, T. Kloek, H. van Dijk, et al. *Econometric methods with applications in business and economics*. OUP Oxford, 2004.

[64] P. Y. Papalambros and D. J. Wilde. *Principles of optimal design: modeling and computation*. Cambridge University Press, 2000.

[65] J. Diebel. *Representing attitude: Euler angles, unit quaternions, and rotation vectors*. Tech. rep. Stanford University California, 2006.

[66] A. L. Schwab and J. P. Meijaard. "How to draw Euler angles and utilize Euler parameters". In: *In Proceedings of ASME IDETC/CIE 2006*. Sept. 2006.

[67] D. Li and S. E. Salcudean. "Modeling, simulation, and control of a hydraulic Stewart platform". In: *Proc. Conf. IEEE Int Robotics and Automation*. Vol. 4. 1997, pp. 3360–3366.

[68] C. Hsu and I. Fong. "Motion control of a hydraulic Stewart platform with computed force feedback". In: *Journal of the Chinese Institute of Engineers* 24.6 (2001), pp. 709–721.

[69] T. I. Fossen. *Guidance and Control of Vehicles, Lecture Notes*. NTTU, 2011.

[70] D. J. Rixen. *Mechanical Analysis for Engineering, Lecture Notes*. Delft University of Technology, 2011.

[71] P. J. From. "An Explicit Formulation of Singularity-Free Dynamic Equations of Mechanical Systems in Lagrangian Form—Part one: Single Rigid Bodies". In: *Modeling, Identification and Control* 33.2 (2012), pp. 45–60.

[72] P. J. From. "An Explicit Formulation of Singularity-Free Dynamic Equations of Mechanical Systems in Lagrangian Form—Part Two: Multibody Systems". In: *Modeling, Identification and Control* 33.2 (2012), pp. 61–68.

[73] *WAMIT USER MANUAL Version 7.0*.

Acronyms

H_∞ H-infinity. 6

AR Auto Regressive. 14

ARMAX Autoregressive Moving Average with Exogenous input. 7, 11

CoG center of gravity. 15–17, 44, 53, 54, 62

DoF Degree of Freedom. 11, 12, 15, 20, 22, 43, 44, 52, 77

DP Dynamic Positioning. 43, 63, 78

FTF Force Transfer Function. 16, 64–67

GPC Generalized Predictive Control. 4, 11, 12, 69

LQG Linear-quadratic-Gaussian. 6, 12

LQPC Linear Quadratic Predictive Control. 4

LS Least Squares. 6, 35

LTI Linear Time Invariant. 5, 19, 20, 55

MPC Model Predictive Control. 3–6, 10–14, 17, 70, 72, 82

NARX Nonlinear Autoregressive with Exogenous input. 7

NED North-East-Down. 15, 17, 50, 63, 78

NMPC Nonlinear Model Predictive Control. 6–11, 70, 72

PID proportional-integral-derivative. 3, 6, 70, 72, 82

QIH-NMPC Quasi-Infinite Horizon NMPC. 8

SPCP Standard Predictive Control Problem. 4, 5

SQP Sequential Quadratic Programming. 11

SVM Support Vector Machines. 11

SVR Support Vector Regression. 11

Master of Science Thesis

TU Delft Delft University of Technology

Challenge the future