Trajectory optimisation to reduce sloshing in open liquid filled containers

by

Harald Leonpacher

This thesis is submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy.

School of Engineering
Liverpool John Moores University
Fachbereich Maschinenbau
Fachhochschule München

February 2000

Abstract

This thesis presents a novel combination of fluid and mechanism models to simulate and optimise the motion of an open topped fluid filled container within a warehouse environment.
Preliminary to the investigation in closed or weakly coupled fluid-structure systems a computational fluid dynamics code has been further developed to model and simulate the behaviour of liquid within an open topped container, driven by a time dependent acceleration profile.
In parallel to the study of the model of the comprehensive system, an optimal control algorithm, namely sequential quadratic programming, has been analysed and used to calculate minimal time motions of the given combination of partial- and ordinary differential equations.
Furthermore, this thesis presents a set of results for the minimal time motion of an open topped fluid filled container with various parameter settings. Additional focus is on the selection and performance of optimisation codes in terms of applicability, speed, robustness and accuracy on the given problem.
A description is given of the development of a practical experimental machine to simulate and actually drive a test case. This has been used to illustrate that the solutions produced are feasible in terms of real world implementation. Results are presented to support the calculated simulations and optimisations.
These results have indicated, that the fastest possible motion is not only limited by a maximum acceleration of the cube, but by a time dependent function.
In a final, critical view on the research the used methodology and codes are analysed. This highlights some areas in which improvements could be made in the future.

Acknowledgements

I am very greatful to the bavarian government and the bavarian research foundation for sponsoring this research as part of the FORTWIHR[1] research federation. This group started as an association between the Technical University of Munich (TUM), the Friedrich-Alexander-University of Erlangen-Nuremberg and major companies (e.g. SIEMENS, BMW, HOECHST or MBB). In order to interest additional corporation partners within small to medium size companies, this project has been assigned to the more practical orientated University of Applied Sciences and Research Munich (Fachhochschule München, FHM).

This piece of work would not have been possible without the help and assistance of numerous people and I would like to take this opportunity to thank them.

Firstly, I would like to thank my academic supervisors Dr. S.S. Douglas, Dr. N.H. Woolley from Liverpool John Moores University (LJMU) and Prof. Dr.-Ing. D. Kraft from FHM. Without their guidance, help and fruitful discussions, this Ph.D. could not have been completed.

Secondly, I must thank all the technicians, and assistants at LJMU, FHM and TUM, in particular Paul Wright, who helped me to set up the model and the experimental servo rig.

The help of other people like my colleagues M. Schaupp, A. M. Connor, A. B. Kassim and my dear friends D. and M. McGill must also be greatfully appreciated. They have always been a source of new views and inspiration.

But most of all I am thankful to my wife Daniela to withstand all my moods and tribulations presented by the obsession to complete this thesis.

---

[1] Bayerischer Forschungsverbund für technisch- wissenschaftliches Hochleistungsrechnen, Bavarian Consortium for High Performance Scientific Computing.

Contents

List of Figures

Abbreviations

BFGS	Broyden-Fletcher-Goldfarb-Shanno
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CIM	Computer Integrated Manufacturing
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
DAC	Digital Analog Converter
DFP	Davidon-Fletcher-Powell
FHM	Fachhochschule München
IVP	Initial Value Problem
KT	Kuhn-Tucker
MAC	Marker and Cell technique
NLP	Nonlinear Programming
OCP	Optimal Control Problem
ODE	Ordinary Differential Equations
PDE	Partial Differential Equations
QP	Quadratic Programming
SOR	Successive Overrelaxation
SQP	Sequential Quadratic Programming
TUM	Technical University Munich
ZOH	Zero Order Hold

Chapter 1

Introduction

The introduction is split into four major parts. In the first section, the background and driving forces which led the author to research the subject of trajectory optimisation to reduce sloshing in open liquid filled containers, are illustrated. In the second section the problem is introduced by illustrating first the real world application of the problem, secondly necessary general simplifications and thirdly a preliminary mathematical formulation of the problem. In the third section the objectives and aims of this research work are given. For this purpose a general overall goal together with sub - aims are formulated. The fourth section summarises the contents of the thesis.

1.1 Background to the research project

In order to illustrate the background to this research project, the four major laboratories participating in this work are introduced and their aims and work further analysed. The first inspiration to the problem of optimal free surface motion within a warehouse environment has been initiated by the existence of an automated warehouse within the Laboratory of Machine Tools at the Faculty of Mechanical Engineering in the University of Applied Sciences and Research Munich (Fachhochschule München, FHM). Within the research work of this laboratory the control of rigid parts within the warehouse has been investigated. Additionally latest warehouse methods can be demonstrated. In general no sophisticated control in terms of optimal motion is used to drive these storage facilities.

The second driving force to initiate the optimal control problem is associated with the

work of two research groups. Both research groups, the Mechanisms Research Group in the School of Engineering at Liverpool John Moores University and the Laboratory of Control Systems and Machine Dynamics at the FHM are working on the research and development of optimal path and design planning of rigid mechanisms. On one hand, the group in Liverpool researches the optimal design of mechanisms with the use of evolutionary algorithms and on the other the optimal path planning of a three degree of freedom robot using numerical optimisation algorithms is studied in Munich. Apart from these universities, numerous other research groups around the world are also currently working on optimal design and control of machines and rigid mechanisms.

The third reason for this research work has been fuelled by the authors interest in fluid dynamics, which was initiated during his time as M.Sc. student in the School of Engineering at Liverpool John Moores University. In this school sophisticated modelling and computation techniques for fluid flow in technical applications are investigated.

Based on these interdisciplinary academic foundations the author started the research into the optimal motion of an open topped fluid filled container within a warehouse environment. The research work contained in this thesis has arisen from the question: "Given a fluid filled open topped container, what's the best method to transport the container in the warehouse?", which was initially formulated by the manufacturer of the warehouse. Clearly the answer to this question is based on the sloshing characteristics of the fluid stimulated. To the knowledge of the author there is no work on the optimal transport of fluids within a two or three dimensional environment. Therefore it is both interesting and novel to investigate the optimal control of a warehouse unit transporting an open topped fluid filled container.

A new approach to optimal trajectory planning has been investigated within this research. To generate results, emphasis has been on the combination of different physical properties within one model. In order to analyse the magnitude of difficulty of the problem dealt with, and to specify objectives for the research project, the following section will further illustrate the problem.

1.2 Problem introduction

In order to specify and formulate the problem of the project the general layout of an automatic warehouse must be illustrated. This is done by describing the integrated warehouse at the Fachhochschule München. Based on this information a reduction of the modelling process is proposed and justified, resulting in a small scale model for the automatic warehouse.

This small scale model rig was built at Liverpool John Moores University to experimentally verify the results of the theoretical calculations. The theoretical representation of this rig is still quite cumbersome in its physics, therefore further simplifications are introduced. These simplifications are necessary for the numerical computation of the problem. Additionally the layout of the experimental installation is presented and further discussed. A comparison is initiated, illustrating how the dynamics of such a small scale model can still be compared with the dynamics of a full size automatic warehouse.

Finally the numerical formulation of the problem will be based on the simplified representation of the small scale model rig.

1.2.1 Warehouse representation

Modern automatic storage facilities have emerged from standard warehouse design. A standard, old fashioned warehouse consists of shelves in various shapes and sizes. The height of the shelf is limited by the working domain reachable by the warehouseman or a fork-lift truck. The size in terms of height and depth of the different shelves has been determined by the product to be stored. These warehouses have been quite specialised and cumbersome.

Development and rationalisation have led the drive to modern warehouse layouts where all storage spaces are of equal size. The storage spaces are distributed in a structured mesh, hence every space has a predefined number and position. Additionally some specialised warehouses do store palettes for bulky items and baskets for small items. The shelves are separated with alleys. In these alleys a conveyor system transports the items to be stored. This transport facility is able to reach all storage spaces. In huge warehouses all single alley conveyors are connected together. Finally it is possible to have a single input-output interface for the whole warehouse which can have thousands of storage spaces.

By neglecting the height of such warehouses[1] the basic structure of such a warehouse can be shown as in Figure(1.1). Here several logistic elements are grouped together and linked with a conveyor-cart system.

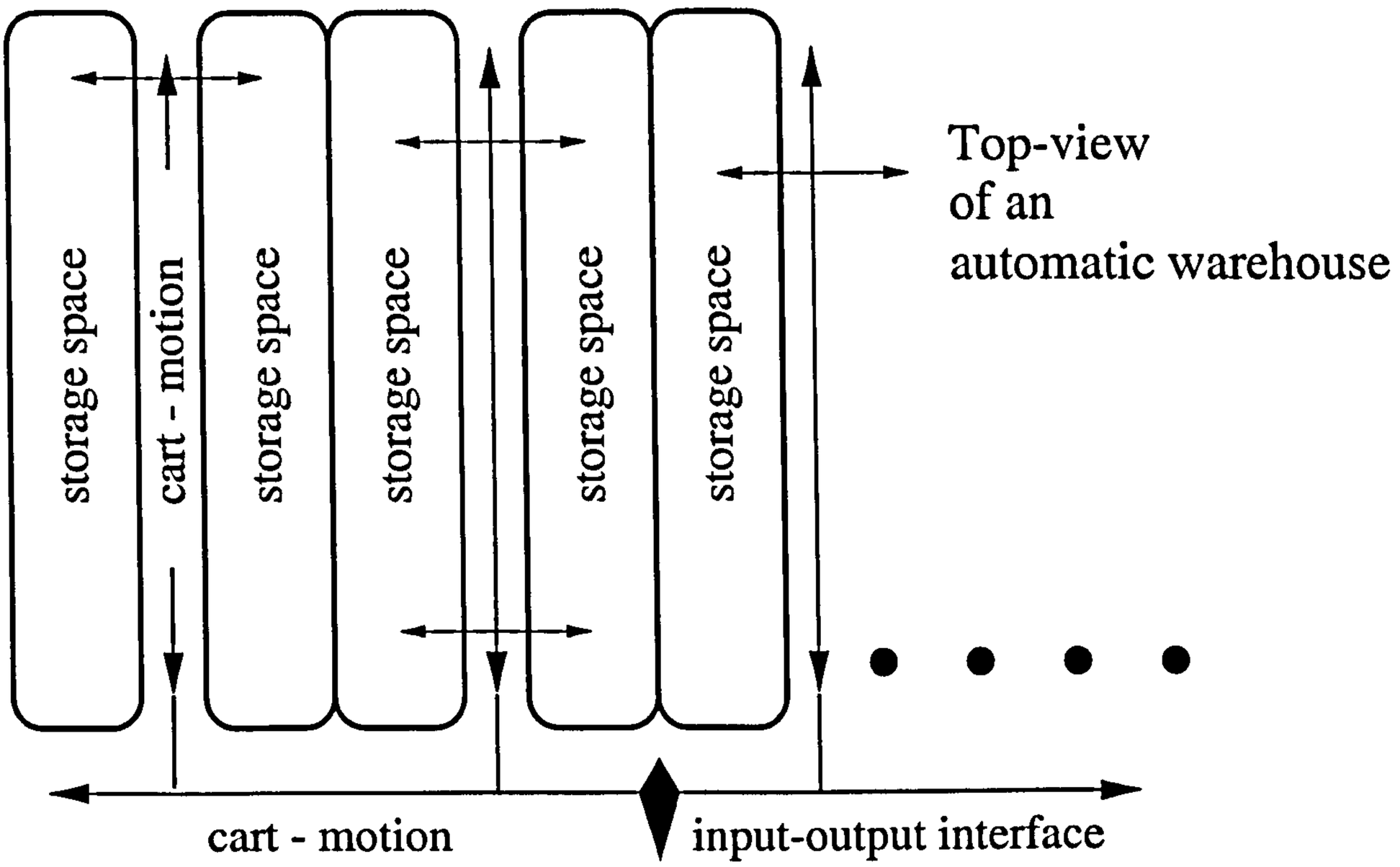


Figure 1.1: General automatic warehouse layout.

Great efforts have been made at the Faculty of Mechanical Engineering at the FHM Munich to establish a CIM environment where the computer aided design (CAD) and the computer aided manufacture (CAM) are linked. This equipment provides testing facilities for dynamic problems. Within these laboratories an automatic warehouse as shown in Figure(1.2) has been built to automatically supply a turning machine with tools and transport the ready made products into the warehouse. The conveyor system can be seen in the middle of Figure(1.2). On the left and the right side of the picture the plastic baskets within the shelves are shown. These baskets are transported from the warehouse interface towards the storing space and vice versa. This warehouse is capable of taking a basket from the interface at quite a low velocity, carrying the basket in an upright two dimensional space to its final destination in the warehouse at maximum speed and placing the basket into the storage space again with a low velocity.

A comprehensive model of this mechanical motion would need to include:

[1] Automatic warehouses have been built with a height over 50 metres.

Figure 1.2: Warehouse at the FHM.

- Modelling of the structural mechanics.
  The motion of the conveyor influences the stability of the shelves. There is possibly a difference between the desired final position of the cart and its actual final position due to displacement of the structure.

- Modelling of the structural dynamics.
  The transporting system which can be located between the motor output motion and the actual basket motion has a dynamic behaviour (e.g. leadscrew dynamics).

- Modelling of the motor controller dynamics.
  There is a difference between the programmed motion profile and the desired motor input motion profile. This transfer function must be modelled and simulated.

- Modelling of the motor dynamics.
  There is also a difference between the motor input motion and the motor output motion, again this has to be modelled and simulated.

In particular the three motions of the cart (warehouse interface – main conveyor – storage

space) and its interaction with the rigid structure would need to be simulated. In the case of transporting an open fluid filled container, the fluid within the container must be modelled and simulated as well. Here one would need the following modelling capabilities:

- Full three dimensional modelling.
  The transport takes place in three dimensional space, furthermore, the fluid container is normally of circular shape.
- Modelling of viscosity.
  This physical property is very important due to its influence on the reduction of the oscillating sloshing because of internal friction.
- Turbulence modelling.
  Especially in high velocity motion the fluid could react with not just laminar but turbulent motion.
- Compressibility modelling.
  The more different fluids are used, the more important this kind of physical representation will become, due to the fact that some fluids are more compressible than others.
- Modelling of surface tension.
  Like the modelling of compressibility, this can become very important for specific fluids, like mercury.

Additionally thermal heat conduction or chemical reaction could influence the motion. This could happen if the motor runs hot or the fluid within the container reacts with the air or the container itself.

With current modelling programmes and supercomputers it is still not possible to model, combine and simulate all the given physical properties. The problem must be reduced in order to be modelled, simulated and finally optimised. The major reason for reducing the complexity of the model is the cost of computation and modelling. Due to the fact that the model must be verified, the number of parameters influencing the model should be manageable. Hence, the specific influence on the model of every single parameter can be thoroughly investigated. Therefore it has been intended to start the study of free surface motion within an automatic warehouse only with a two dimensional model. Within this model the computational costs are of order $\mathcal{O}(n^2)$, where n is the number of mesh-cells in

one spatial direction[2]. A three dimensional model for the fluid motion is more than n-times more expensive in terms of computational costs.

To reduce the model to 2D, the following reductions must be specified and performed:

1. The start motion of the basket, from the warehouse interface onto the cart, and the final motion, from the cart into the shelf, are neglected. These motions are assumed to be normally in slow motion, therefore the motion in the fluid container is not very severe and need not be investigated.

2. The fluid container is limited to two dimensions, hence the container must be of rectangular shape with infinite length in the Z-direction. In general, the friction influence of the front and back walls can be neglected.

Additional simplifications are necessary for the representation of the fluid. For the initial considerations the physical properties can be reduced to a single representative fluid. The fluid to be used is water, due to the fact that a lot of substances in the food and chemical industry are water-based. To model water, one would need at least a model for a Newtonian fluid being viscous, incompressible and, assuming moderate sloshing behaviour, having a laminar fluid flow. There is no chemical reaction in the fluid and the surface tension is quite low and can be neglected (see Section (3.8) for illustration).

The assumption of laminar flow is essential. Turbulence modelling is still a field of intense research and therefore very difficult to handle. To capture all vortices of the turbulence flow, the control volumes of the grid must be very small (at least as small as the smallest vortices). This leads to a very high number of mesh cells resulting in very long calculation times for a single flow field calculation. This is not desirable within the optimisation.

Due to the variety of warehouses which have been built it has been desired to focus on a very elementary version. This version should include only the parts which are necessary for any kind of warehouse. It is recognized that the structural dynamics of the cart have to be included within the modelling process. These dynamics from the motor shaft and the driving system of the cart (e.g. a leadscrew) are influenced by the motion of the fluid within the container. The structure of the warehouse is normally stiff enough to be neglected. The dynamics of the motor controller and the motor itself are very different for every single type of machine and control. In general, the output motion profile of the motor is the only interesting influence of the drive concerning the motion of the cart.

[2]Assuming an equally spaced, structured grid.

Furthermore the observation equipment for the sloshing behaviour of the fluid during the cart motion should be as simple as possible. It has been decided that the equipment to do so in the warehouse at the FHM would be too expensive (It would be necessary to develop a mobile, and very light measuring equipment to be mounted on the cart). Therefore the development of a small scale model has been proposed to verify the results from the optimisation. Within the Mechanisms Research Group in the School of Engineering at John Moores University Liverpool, Osypiw[82] developed a two dimensional servo motor driven rig to drive a holding device in an automatic fuselink assembly. This rig has been

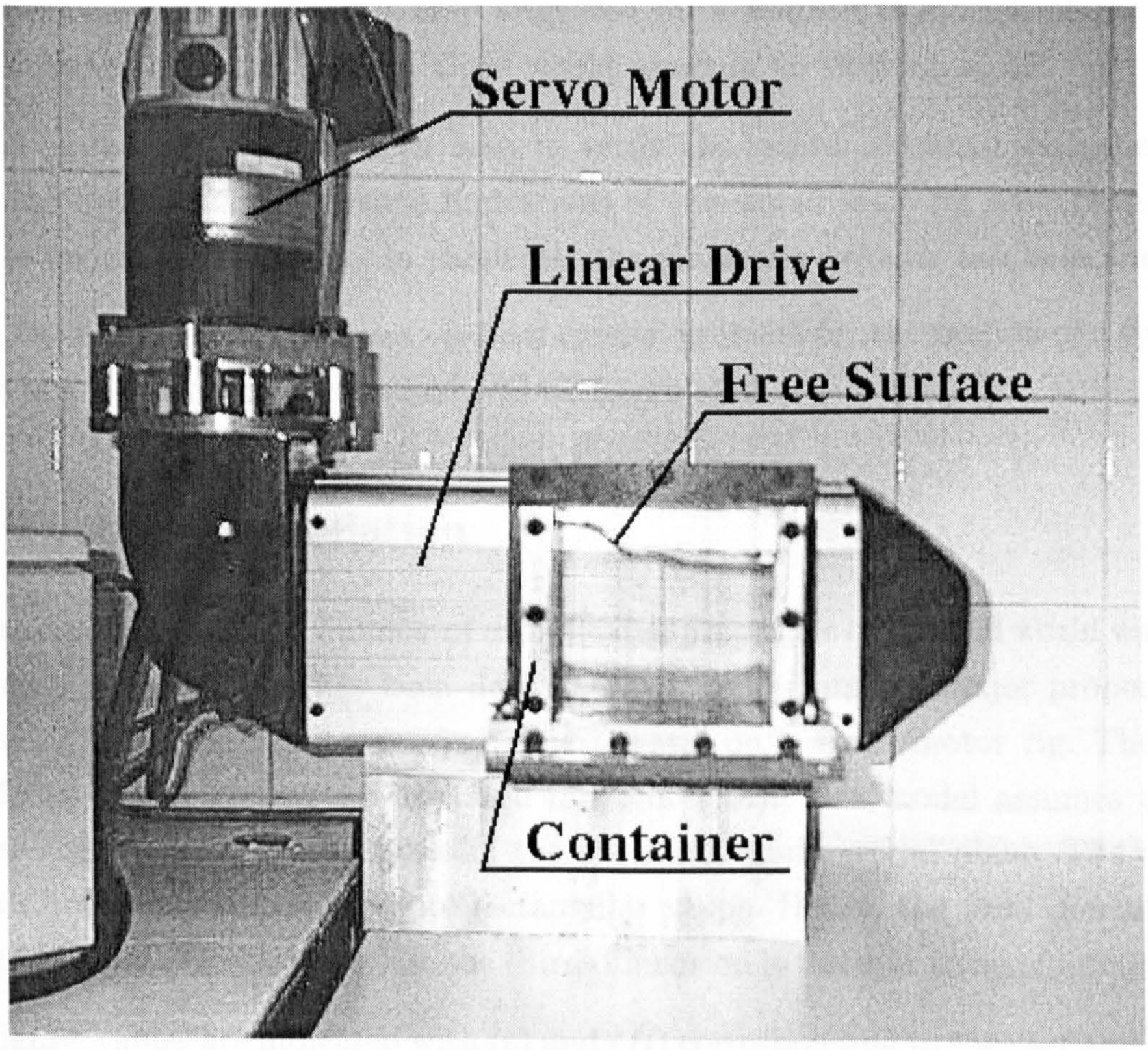


Figure 1.3: The experimental rig.

modified (see Figure(1.3)) to fit the needs for the experimental simulation of a warehouse environment.

Two linear conveyors can be mounted together to simulate an upright two dimensional motion domain within a warehouse. To drive the two dimensional motion, the container is mounted on the vertical conveyor. This vertical conveyor with its actuator and the container

are mounted on the horizontal conveyor. This system results in a very high inertia for the horizontal motion. In order to investigate the influence of the fluid onto the motion of the conveyor more thoroughly the low inertia version has been used in Figure(1.3). This means that the horizontal conveyor does not need to transport the vertical conveyor and its actuator, hence the ratio of the mass of the fluid and the total mass to be transported (mass of the fluid, of the container and of the holding device) by the horizontal conveyor is higher, with the result that the fluid motion has more influence on the motion of the conveyor. Another reason to use this reduced one dimensional motion is the fact that the sloshing behaviour of the fluid is only triggered by a horizontal motion of the cart. A motion limited to a vertical motion alone would result in no sloshing at all.

This small scale model rig has been used to verify the results obtained with the simulation and the optimisation. Physical limitations of this small scale rig have been used as boundaries for the optimisation. In particular the maximum velocity has been limited.

Based on these limitations the time optimal control problem for the motion of a fluid filled container in a warehouse environment will be formulated.

1.2.2 Problem formulation

In order to simplify the large number of complicated properties of the real world warehouse, a small scale model system has been derived which represents the major properties of a warehouse. This model can be verified experimental on a servo motor rig. The general layout of this model system is illustrated in Figure(1.4). This model assumes that in a warehouse the movements take place in the horizontal and vertical plane. The container filled with liquid is assumed to be of rectangular shape. Hence, the fluid domain can be represented in two dimensions, while the third dimension is thought to be of infinite length.

In this Figure, values are indicated with (c) and (fl) corresponding to the cart and the fluid, respectively. The length (l) and height (h) indicate the dimensions of the fluid filled cube. The main movement of the fluid will occur while moving the container in the horizontal X-direction using the acceleration-vector of the cart (F_x). The centres of gravity of the cart and the fluid are indicated with (S). The force-vector (F_{fl}) is the force of the fluid on the cart. Sloshing can only occur if a force orthogonal to gravity exists. The force-vector of gravity acts in the Y-direction and has not been given in Figure(1.4). Investigations illustrated in this thesis will only concern horizontal, one dimensional movement of the

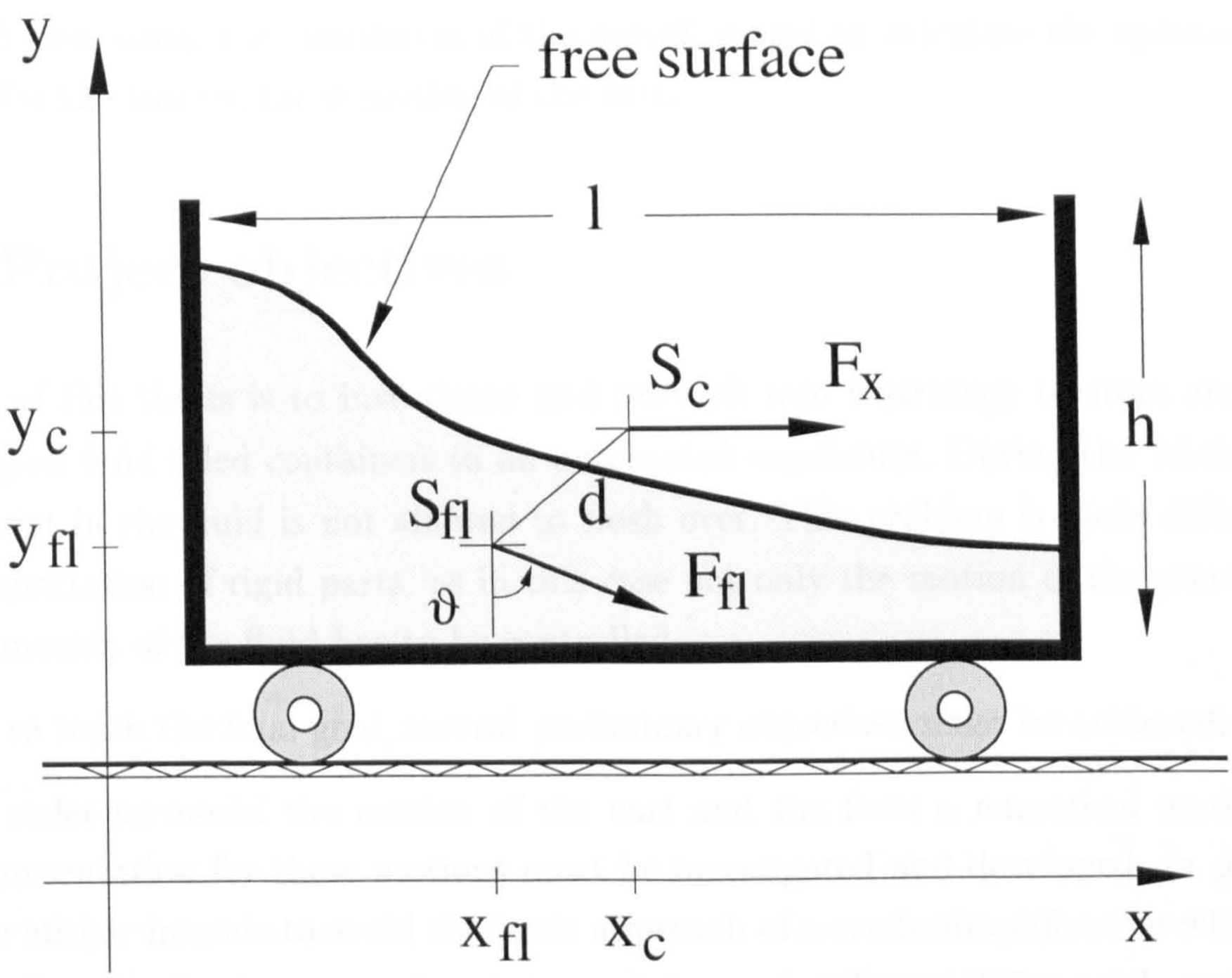


Figure 1.4: Model of the dynamic system.

platform.

In particular, the cart of the small scale servo rig will be modelled using common mechanical laws of the interaction of masses, springs and damping devices. Their mathematical representation results in a system of ordinary differential equations (ODE). These equations are the transfer functions correlating the input data of the driving motors and the output motion of the cart.

Fluid flow is modelled and solved for numerically by using finite discretisation of the fluid domain. There are three different major discretisation strategies, finite volume-, finite element-, and finite difference techniques. Main properties of interest of the fluid domain will be velocity and pressure at any given point in space and time. Free surface motion introduces additional information such as surface tension and surface shape.

Based on these models a comprehensive model for the whole system will be generated. This comprehensive model will be used to simulate the behaviour of the fluid and the cart due to a specified design or control function. An optimisation procedure will be studied

and developed, using the simulation of the overall model to calculate the optimal control function for the time optimal motion of the cart.

1.3 Project objectives

The aim of this thesis is to investigate and research into a strategy to store and extract open topped fluid filled containers in an automated warehouse. During the whole process a constraint in the fluid is not allowed to slosh over. This problem is more difficult than the transportation of rigid parts, as in this case not only the motion of the platform, but also the motion of the fluid has to be controlled.

In order to reach the final goal, several preliminary objectives must be achieved:

1. In order to model the motion of the cart and the fluid a numerical mathematical representation for these motions must be investigated and developed. In particular, the author intends to avoid the basic approach of a crude simplification which would involve the development of a system of damped ordinary differential equations, to model the motion of the fluid. By building computational blocks, to be combined within the optimisation, users should be able to adapt the efficiency and accuracy of the modelling codes to the capabilities of modern computing facilities. Hence, when the speed of computers increase future users will be able to incorporate more physical properties to be modelled within the code without the need for restructuring the modelling and optimisation programmes.

2. The modelling process of the research must also involve the combination of the fluid model and the model for the mechanical system. Currently there are different approaches under research. Particular focus must be on the accuracy of the results and their affiliation to efficiency. Eventually, the author intends to develop a combined fluid- structure interaction model. This overall model must be able to calculate the response of the fluid and the cart based on the input motion of the drive.

3. Furthermore this model must be integrated into an optimisation strategy. In order to do so, different optimisation strategies must be investigated and the most appropriate for the problem calculations must be used.

4. Generally, the manufacturer of the warehouse is interested in optimal motions of the

cart. In particular time and energy optimal motions.

5. The final aim of the research is to experimentally validate the theoretical results in a practical verification. This feedback is necessary to justify the various steps of the research and to formulate proposals for further improvement of the models, the optimisation and the calculation of the process.

As a summary of the objectives, the following can be stated: The aim of the investigation is to integrate fluid dynamics analysis with machine dynamics within an optimisation strategy for application within automated warehouse technologies. This type of problem is a special case of fluid-structure interaction. The goal is to enable optimal transport paths to be found for liquid-filled open containers within a warehouse.

1.4 Thesis structure

This thesis is split into eight chapters that describe the overall research programme, the investigation of the various computational techniques and their combination within the optimisation and studies the results of the performed optimisations.

Chapter (2) contains a comprehensive literature search and illustrates the state of the art in this field of research. This literature research was carried out throughout the duration of the work and is intended to highlight existing research and illustrate the need for and novelty of the proposed project.

Chapter (3) illustrates the basic structure and coding of the fluid model. Furthermore, some additions for the representation of transient force impact are given.

The aim of Chapter (4) is to illustrate the model of the mechanical system, the difficulties of combining the different models and their integration within the optimisation algorithm. Here justifications for the use of the mechanical model and the specific optimisation code are given.

Chapter (5) presents the numerical background for the simulation and optimisation of the problem. The results, obtained within this research project are collected and studied within Chapter (6). Results from theoretical calculations and practical experiments are presented and verified.

Chapter (7) is a discussion of the results of the project in relation to the project objectives. It aims to further illustrate the obtained results from a more distant point of view.

Finally, Chapter (8) concludes the thesis, outlines the contribution to knowledge of this

research project and also includes a number of suggestions for possible future research on the subject.

The references, cited within the thesis are collected within the References.
The Appendices include some code additions for the model of the fluid motion and the optimisation code. Additionally copies of published papers concerning the work in this area are included.

1.5 Nomenclature

d	time dependent distance between S_{fl} and S_c
h	height of the container
l	length of the container
F_{fl}	fluid force
F_x	external force
S	centre of gravity
$\boldsymbol{x}, \vec{x}(x, y, z)$	position vector with cartesian components x, y, z

Chapter 2

Literature Review

The goals of the research reported in this thesis are twofold. In the first instance a realistic model representing an open topped vessel containing liquid, being moved in one dimension by an external force is sought. In the second instance the model is to be integrated within an optimisation procedure and hence optimal trajectories can be obtained.

The work integrates three separate bodies of knowledge. Fluid dynamics knowledge is required to enable a free surface motion to be accurately modelled and predicted. Mechanical system modelling is required to enable the servo driven transport system to be established. And finally optimisation knowledge is required to identify which of the many potential optimisation approaches is best suited to the problem.

This literature review starts with a review of the fluid dynamics focused on free surface flow in open topped containers, the second part of the review explores the modelling of dynamic mechanical systems and particular emphasis is given to work that has been reported about fluid structure interaction. Finally, the third part of the review illustrates the current state of research in the various methods of optimisation.

Furthermore, the literature review on these three major topics has been further focused based on certain limitations. These limitations were:

- The available computational resources,
- The possible investment in software and
- The tackled knowledge base.

The available computational resources limits the comprehensiveness of the model, the simulation and the optimisation. For instance a genetic algorithm as an example for a modern optimisation scheme is best performed on multiple machines. The problem can be distributed easily over several CPU's. Due to the relatively huge computation costs of such an optimisation scheme it is only advantageous on simple models or on huge computer platforms.

The investment in software limits mainly the comprehensiveness of the modelling and simulation tools. Modern, sophisticated codes for modelling free surface flow can be quite expensive. Additionally it was necessary to integrate the model for the mechanical system and to adapt the CFD-code to the requirements of the optimisation. These enhancement need the privilege of access to the source code. In general, developers of sophisticated CFD-codes do not provide their source code, or if so only under high costs.

The tackled knowledge base was a limitation in terms of subjects which have already been investigated quite thoroughly. It has not been intended to further investigate modelling strategies for free surface motion or the modelling and simulation of mechanical systems. Furthermore, it was not within the scope of the research to develop new strategies of optimisation, but to profitably develop a strategy to tackle such a difficult and non-linear model within an optimisation scheme.

This chapter is split in three major parts. In the first part different strategies for the modelling of the free surface motion within CFD-codes will be reviewed and some codes will be discussed. A sub-part will focus on some new strategies to model free surface motion.

The second part of the literature review is focused on the mechanical system. Some basic foundations are given for the representation of the warehouse in terms of ordinary differential equations. In addition some notes are given for the solution strategies to solve such systems of ODEs.

An additional focus is on the problems of fluid-solid interaction and the control of a fluid. Here some computational work will be presented which has already been undertaken in this field. This subject has re-emerged due to the increase of computational resources and power of recent computers and the developments within mathematics and computer sciences.

The final part of this chapter is dedicated to available optimisation codes and their applicability to the given research problem.

The review concludes by identifying the techniques which will be built upon to attain the

research objectives.

2.1 Modelling of free surface flow

The history of modelling fluid systems has been a history of experiments in water channels and wind tunnels (Smith et al.[106], [4], Pope[85]). Even though the governing equations for Newtonian fluid dynamics, the unsteady Navier–Stokes equations, have been known for 150 years or more, such experiments have been the only possible way to reduce costs in the developing stage of a product. The steady improvement in the speed of computers and memory size since the late 1950s has led to the emergence of computational fluid dynamics (CFD). With this tool it is possible to evaluate solutions for the Navier–Stokes equations. The Navier–Stokes equations represent the physical properties of a flow field. Following Chapmann et al.[16] and Fletcher[31], CFD provides five major advantages compared with experimental fluid dynamics:

1. Lead time in design and development is significantly reduced.
2. CFD can simulate flow conditions not reproducible in experimental model tests.
3. CFD provides more detailed and comprehensive information.
4. CFD is increasingly more cost-effective than wind-tunnel testing.
5. CFD produces a lower energy consumption.

In order to simulate the fluid motion it is necessary to model the fluid and container layout within a CFD-code. There are several different strategies to evaluate a free surface model to calculate the fluid motion. Hawken et al.[47] give a comprehensive overview and literature review on adaptive node movement techniques. These techniques attempt to update the representation of the fluid domain at certain time steps, hence moving the mesh with the calculated movement of the free surface.

One example of a programme using these techniques and in particular utilising 'spines' to track or guide the free surface is FIDAP[30]. Spines are straight lines passing through the free surface nodes which determine a direction for the free surface to move. These spines are elements of the regular mesh and not necessarily perpendicular to the free surface.

Another example, using node movement techniques is given in Soulaïmani et al.[107] where the problem of large amplitude sloshing is used as a test case. The external forces are of the form $f = (\mathrm{A}g \sin \omega t, -g)$ for the two spatial dimensions and therefore only used to determine the frequency of the uniformly accelerated fluid. The same test case is used by Huerta et al.[55] in the context of arbitrary Lagrangian–Eulerian (ALE) Petrov–Galerkin finite element techniques. Also Tezduyar et al.[110][111] used this test case to confirm their technique of deforming-spatial-domain/space-time procedure where the deformation of the spatial domain with respect to time is taken into account automatically.

In Behr et al.[9] the test case of large amplitude sloshing is extended to three spatial dimensions. An oscillating external excitation is used and the objective of this calculation is to assess the capabilities of the space-time finite element formulation as a method for handling deforming domain problems.

Fluid mechanics research at the Technical University Erlangen–Nürnberg includes free surface flow. A CFD-code named FASTEST has been developed to fit problems occurring when combining free surface calculations with parallel programming. Extensions to 3D modelling have been considered. The approach, presented by Egelja et al.[24], belongs to the adaptive grid Eulerian methods and consists of solving discretised mass and Navier–Stokes equations on a non-orthogonal moving grid using the finite volume approach in connection with a pressure-correction technique.

A different approach, using particles moving on a fixed mesh, has been proposed and translated into a software programme by Griebel et al.[45]. The Navier–Stokes equations are solved on a rectangular control mesh. The properties of the mesh-cell determines the further movement of the particles which are free to move in the whole domain. They are redirected at the walls due to certain algorithms. Based on the new particle distribution a new mesh is applied to the domain.

The literature review on the field of free surface movement reveals there is a broad area of research modelling the fluid flow. To perform the calculation of an optimal control for the cart it has been necessary to use a CFD-code with the best balance between accuracy and computational efficiency. Therefore it was necessary to find the most suitable CFD-code for the modelling problem. The decision on what type of representation should be used as a trade off between computational efficiency, accuracy and other criteria has been investigated and studied by Leonpacher[69].

Leonpacher classified and investigated among others the following CFD-codes:

1. Commercial (not freely available) CFD-codes
   - ANSYS/FLOTRAN [3]
   - FIDAP [30]
   - FLUENT [34]
2. Shareware, public-domain and freeware CFD-codes.
   - CLAWPACK [71]
   - SINDA/FLUINT at "http://www.webcom.com/~crtech"
   - FCT [2]
   - NSC2KE [74]
   - FEMLAB [26]
   - NAST2D [45]

A decision matrix was specified and used to select the most appropriate code for the modelling and simulation of the free surface motion. The selected code was NAST2D developed by Griebel et al.[45]. In comparison to all other codes, this code resulted to be the best on the given problem due to the following characteristics:

1. The code is capable of modelling an incompressible, laminar, viscous, unsteady, free surface fluid flow based on the Navier–Stokes equations.
2. A finite difference scheme is used, which is quite simple and cost effective in terms of calculation time.
3. The source code is freely available, and thus the platform to operate the code is arbitrary.
4. Only one additional module concerning external forces need be developed.
5. The developers of the code at Munich Technical University are easy to reach for communication.

2.2 The mechanical system

The mechanical system is built of several components which interact as a multi-body system. There are motors, gears and rigid bodies with defined characteristics. These characteristics are modelled within transfer functions. Several authors (Driels[21], Emanuel et al.[25], Prentis[91]) have derived the characteristic behaviour of the prescribed elements and their interaction. The overall performance of the system can be represented in a system of ordinary differential equations (ODE).

There has been extensive study (Sciavicco et al.[101], Bishop[10], Schaupp[97]) modelling and simulating mechanical systems within software packages such as MATLAB[72] (including SIMULINK), MAPLE (e.g. Kraft[66]) and SIMPACK[104].

The techniques to solve a system of ordinary differential equations are well known, mature and several algorithms have been developed, including the well established initial value solver of Euler or Runge–Kutta–Fehlberg (Fehlberg[27]), or more recent techniques such as collocation (e.g. Ascher et al.[5], Bulirsch et al.[13]).

In order to reduce computational costs, the well established Runge–Kutta–Fehlberg formulae will be used.

2.3 Control of fluid, fluid-structure interaction

In contrast to the calculation of the motion of the fluid, control of fluid motion is intended to alter the effects of the mechanical environment on the fluid. This is the case in the problem of optimal trajectory planning for an open fluid filled container. Two different approaches are distinguishable. On the one hand an alteration of the design of the environment can be performed, or on the other hand a variation of the external energy which triggers the motion can be performed. The second approach has been studied within this research project.

Some investigations have been undertaken to evaluate the influence on the fluid of an altered mechanical design. Muto et al.[77] tackled a problem of vibrational phenomena regarding flow in a fuel tank during an earthquake. The aim of the work was the development of effective submerged blocks to restrain the fluid from sloshing with high amplitudes.

Popov et al.[86] present and discuss the liquid behaviour in rectangular road containers undergoing a turning or braking manoeuvre. Experimental data was compared with calculated results performed with a marker–and–cell method (First established by Harlow et al.[46] and developed and documented in Hirsch[50][51] and Hoffmann et al.[53][54]). In this and further investigations Popov et al.[87][88] on cylindrical and rectangular road containers the structure of the container has been investigated and optimised. The movement is said to be inalterable and has not been given any focus in terms of control.

Another design optimisation problem associated with fluid motion has been under research by Cliff et al.[17]. A duct flow with a shock was studied, where the location of the shock was treated as a parameter. The study was concerned with optimal cross-sectional layout of the duct at the position of the shock.

This research involved also a study by Shenoy et al.[103] into the optimal heat transfer from a solid body to a stationary, incompressible and convection driven fluid flow. Here the design parameter is the temperature distribution on the inflow boundary to achieve a predefined value on the outflow boundary.

Due to the increasing demands for accuracy, the interactions of different physical phenomena are no longer neglected or simplified, but are taken into account in detail. These developments are possible due to the improved algorithms in the field of mathematics and computer sciences, and the increase of available computational capacity.

The FORTWIHR[1] Research Group, the sponsors of this work, has among other things been focusing on fluid-solid interaction problems. Following Bungartz et al.[14] different interaction possibilities between the following physical properties can be specified as follows:

- Structural dynamics
- Fluid dynamics
- Heat conduction
- Electromechanics
- Chemical reactions

He defines two different coupling strategies for the combination of the different physical interactions, namely weakly coupled systems and strongly coupled systems. These coupled

[1]Research group for high performance scientific and engineering computation.

systems are characterised by the mode of computation. A strongly coupled system is solved for every represented physical property within the smallest time step of the simulation. In a weakly coupled system the represented physical properties are calculated one after another. This coupling problem has been investigated and solved in this research project as well and will be further discussed in Section (4.3).

Another project by Dütsch et al.[22] has focused on the accuracy of the prediction of a fluid-structure interaction problem compared to an experiment. Here the fluid damped oscillations of a lamina at large initial amplitudes has been investigated.

Nomura[78] focuses on the development of a closed coupled algorithm to model and simulate fluid-structure interaction problems. This investigation has been interested in the motion of a fluid due to a given, fixed excitation.

Fursikov[35] recently published a contribution about optimal control of Navier–Stokes problems. In this article the body force is regarded as the control. Necessary and sufficient conditions of minimum and uniqueness of the solution were studied mathematically. The major drawback of the obtained results is the used simplification of the Navier–Stokes equations resulting in abstract, ordinary differential equations. Hence the accordance with reality is no longer guaranteed and practical applicability is greatly restricted.

To the knowledge of the author there is no source dealing with the control of the external energy to influence the fluid motion.

2.4 Optimisation schemes

In general, there are two different optimisation tasks. In the first task, the general parameter optimisation problems, a normally non-linear objective function based on several, time-independent parameters and limited by state constraints has to be optimised. The most important limitation in this sub-family of optimisation is the time-independence of the design variables.

These problems can be solved, apart from various other strategies, with genetic algorithms, where the natural process of survival of the fittest based on mutation and selection is used to calculate a new 'generation' of parameters, as described by Goldberg[42]. A general overview of optimisation with genetic and selective algorithms is given by Kinnebrock[60].

Another solution strategy for parameter optimisation applications is tabu search which has been developed by Glover[38]-[41]. The search is constrained by classifying certain of its moves as not allowed, the search direction being set free by a short term memory function, providing 'strategic forgetting'.

A genetic algorithm has been used by Connor[18] to evaluate an optimal five bar mechanism layout for a given trajectory to follow. Another often referenced problem for this category of optimisation is the travelling salesman problem([60][42]).

The second general idea of optimal control is the calculation of an optimal solution of an object function based on a system of non-linear equations limited by state constraints with time dependent parameters. This problem has often been utilised successfully as a two point boundary value problem (e.g. Ascher et al.[6], Bulirsch et al.[13] and Kraft[63]).

There are several different approaches as described by Gill et al.[36] to solve these kind of problems. Another helpful book to accompany an engineer in solving a practical optimisation problem is More et al.[75] where an insight in the different branches of optimisation is given and several different optimisation software packages are listed and categorised due to their solution algorithms.

The problem of time minimal movement of an open fluid filled container is an optimal control problem with a set of time dependent parameters. These kind of problems are often called trajectory optimisation problems. The general approach to calculate such optimisation problems involves 'hill climbing' in some form. Hill climbing describes the search for the fastest descent on the surface of states of the fluid due to the parameters and the dynamics of the system. Hence the solution domain is said to be convex and uniformly formed with presumably several local minimal solutions.

Trajectory path planning problems result in a two-point boundary-value problem as described by Kirk[61] or Gill[36]. They can be solved using direct shooting or collocation methods and subsequent application of a quadratic programming (QP) algorithm.In comparison, indirect shooting or indirect collocation methods incorporate the state constraints into the model function and create additional differential equations. They are often extended to multiple shooting methods (see Diekhoff et al.[20], Bulirsch et al.[13]) to increase the stability of the algorithm. The drawback of larger computational efforts to solve the additional differential equation is compensated by the integrated optimisation. These methods do not result in a boundary value problem. If a solution to the system can be found,

this solution will be optimal.

Direct shooting in conjunction with sequential quadratic programming (SQP) (see eg. Gill et al.[37]) has been used by Kraft[63][65] for the calculation of an optimal trajectory for a six–degree–freedom robot.

In the context of optimal flow control using SQP-methods Ravindran[92] presented a sequential quadratic programming method and its implementation to treat optimal Dirichlet control problems associated with steady Navier–Stokes equations. Due to the restriction on steady flow, this approach has not been applicable to the transient problem of trajectory optimisation involving sloshing fluid motion.

Further study has been undertaken by Barcley et al.[7] on the solution of large scale optimisation problems using SQP algorithms. These algorithms calculate a new direction (see Murray et al.[76]) and a new step-size due to gradient calculations on the surface of the state. The use of a direct collocation method has been described by von Stryk[109] to calculate the optimal ascent of the lower stage of a Sänger–type vehicle.

Several different optimisation strategies have been tested by Schittkowski et al.[100]. The SQP optimisation method emerged as quite effective and robust on a variety of test problems.

Based on these findings the MATLAB OPTIMIZATION TOOLBOX[73] has been used as basis for the optimisation of the optimal trajectory problem. The different modules within this toolbox can be exchanged. There are different solvers for the quadratic programming subproblem like SOLNP by Ye[113].

Following Kassim[59] additional boundary conditions in the optimisation must be specified to increase the effectiveness of the cart motion in terms of friction and energy loss. Norton[79] has been used to develop specific acceleration and jerk functions for the motion of the cart. These limitations do result in an efficient and smooth motion profile.

This chapter has covered some of the recent literature that is relevant to this research. To the knowledge of the author there is no source dealing directly with the proposed work. The control of fluid motion within a predefined and fixed structural environment is new and novel. The investigations and studies of this research project allowed the calculation of optimised trajectories for open fluid filled containers.

2.5 Nomenclature

$\mathbf{A}$	general matrix of values
f, $\vec{f}(f_x, f_y, f_z)$	force, force vector with cartesian components (f_x, f_y, f_z)
g	natural gravity
t	time
ω	angular velocity

Chapter 3

Modelling of fluid motion

This chapter deals with theory related to the modelling of fluid flow, particularly modelling of a fluid system with a free surface in the manner to be used within the simulation of the optimal control problem.

The approach by Griebel et al.[45] for modelling fluid motion with a free surface will be described. A finite difference code (NaSt2D) for free surface motion has been chosen based on a discussion of the problem by Leonpacher[68]. The program is a two dimensional solver for the incompressible, transient Navier–Stokes equations including the energy equation and incorporating free surface boundary conditions.
In addition, some notes are presented concerning necessary alterations to this code.

3.1 Modelling of the free surface motion

To model the sloshing fluid motion the CFD-code NaSt2D was chosen[1] based on the following criteria:

- The code was able to perform free surface sloshing.
- The code was accessible in its source form.
- Steady and quick feed back with the developer of the code was possible.

It was possible to add modules to the code to allow

[1] See Leonpacher[68].

- Computations with time varying acceleration,
- Change of accuracy of the sloshing values,
- Integration of the mechanical model,
- Alterations to fit the CFD-code into an optimisation cycle.

Based on the initial idea of the research, to evaluate strategies for transporting fluids in automated warehouses, the properties for these fluids (e.g. water, ink or other chemical liquids) can be specified. The fluid aimed to be simulated is transient, laminar, incompressible and viscous and is therefore represented mathematically by the Navier–Stokes equations.

NaSt2D is based on the Marker-and-Cell method (MAC) developed by Harlow et al.[46] and has been especially designed for fluids with the previous defined properties in a domain with open boundaries. It uses finite differences for discretisation on a structured equidistant staggered grid, central and upwind (donor-cell) discretisation for the convective terms and a first order explicit time stepping scheme.

The programme was developed for educational purposes at a computer science institute. Thus, it is easy to understand and easy to implement but it is not a "state of the art" programme. But in spite of the relative age and simplicity of the code it is flexible and quite powerful.

3.2 The Navier–Stokes equations

In order to describe a flow field mathematically, some definitions for the flow field and the fluid must be specified. For this reason a mathematical domain $\Omega \subset \mathcal{R}^N$ $(N \in \{2,3\})$ in the time interval $t \in [0, t_{end}]$ is specified. The motion of the fluid is characterised by the velocity field $\vec{u} : \Omega \times [0, t_{end}] \longrightarrow \mathcal{R}^N$ the pressure distribution $p : \Omega \times [0, t_{end}] \longrightarrow \mathcal{R}$ and the density of the fluid $\varrho : \Omega \times [0, t_{end}] \longrightarrow \mathcal{R}$.

Assuming incompressible flow then the density $\varrho(\vec{x}, t) = \varrho_\infty = \text{const}$. For transient viscous flow, the fluid motion is described by a system of partial differential equations (PDE) of

non-dimensionalised[2] form

$$\frac{\partial}{\partial t}\vec{u} + (\vec{u} \cdot \text{grad})\vec{u} + \text{grad}p = \frac{1}{Re}\Delta\vec{u} + \vec{f} \tag{3.1}$$

$$\text{div}\vec{u} = 0 \tag{3.2}$$

Equation (3.1) describes the conservation of linear momentum and Equation (3.2) the conservation of mass. No heat source or sink in the system is assumed and the temperature increase in the fluid due to friction is neglected, thus allowing the conservation of energy equation to be neglected. Equation (3.1) is called the Navier–Stokes equation where $Re = \frac{\varrho_\infty u_\infty L}{\mu}$ is the dimensionless Reynolds number and $\vec{f}$ are external forces, such as gravitation, acting on the fluid. In two dimensional space, the operators div and grad are defined as following:

$$\text{grad}u = (\partial u/\partial x, \partial u/\partial y), \qquad \text{div}\vec{u} = \partial u/\partial x + \partial v/\partial y.$$

and therefore

$$(\vec{u} \cdot \text{grad})\vec{u} = \begin{pmatrix} (u\partial u)/\partial x + (v\partial u)/\partial y \\ (u\partial v)/\partial x + (v\partial v)/\partial y \end{pmatrix}$$

3.3 Dynamic similarity of fluid flows

In numerical fluid dynamics nearly all calculations are based on non-dimensionalised equations such as the non-dimensionalised Navier–Stokes equations given in the previous section. The idea of dimensionless representation is based on the dynamic similarity of a small scale model compared to its large scale real world application. It is possible to combine the characteristic values of fluid flow (namely a characteristic length L, velocity u_∞, dynamic viscosity μ and characteristic density ϱ) to dimensionless values in a way that statements are possible about the correlation of model and real world.

In Figure(3.1) the domain Ω^* has been scaled down from the domain Ω by $S < 1$. The geometries Ω and $\Omega^* = S \cdot \Omega$ are similar.

The dimensionless values are generated as following:

$$\text{dimensionless value} = \frac{\text{value with dimension}}{\text{correlation value with the same physical dimension}}$$

In particular the following characteristics must hold for the correlation values

[2] See Hoffmann[53].

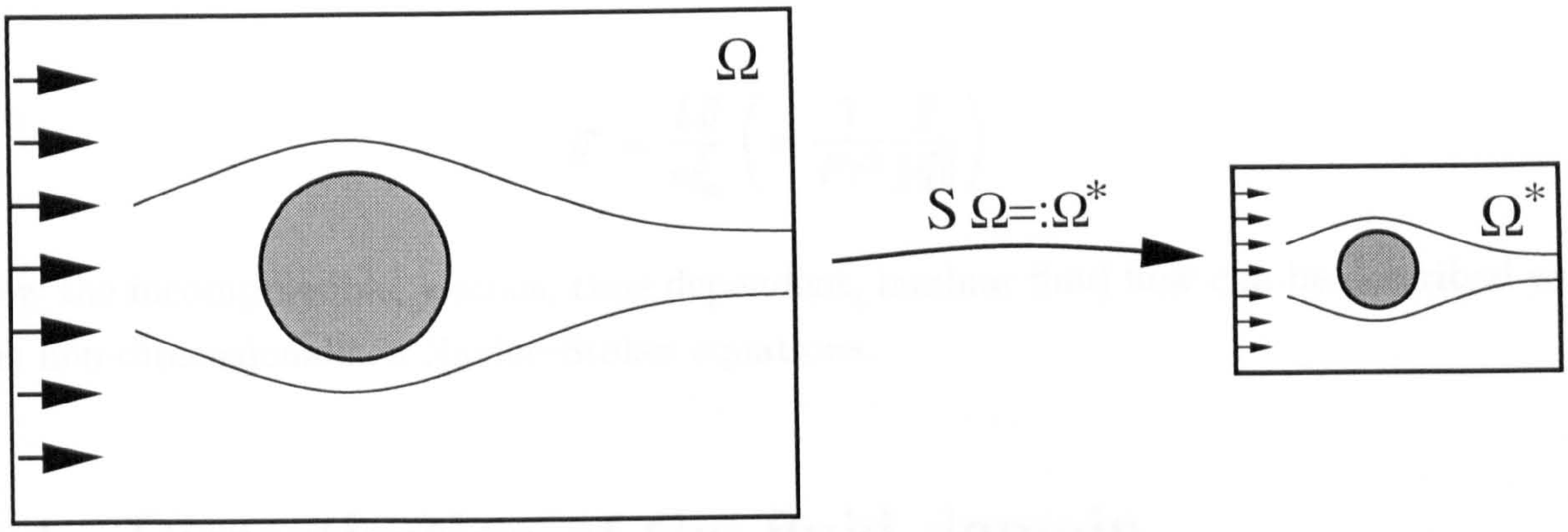


Figure 3.1: Similarity of fluid flows.

- constant for the problem,
- must be known initially,
- must be characteristic for the problem.

The CFD-code NaSt2D uses the following dimensionless values for the Navier–Stokes equations:

$$\vec{x}^* = \frac{\vec{x}}{L}, \quad t^* = \frac{u_\infty t}{L}, \quad \vec{u}^* = \frac{\vec{u}}{u_\infty}, \quad p^* = \frac{p - p_\infty}{\varrho_\infty u_\infty^2}. \tag{3.3}$$

Where $L, u_\infty, p_\infty, \varrho_\infty$ are scalar constants. The transformation of the Navier–Stokes equations using the variables (3.3) yields to the following system:

$$\frac{\partial}{\partial t^*}\vec{u}^* + (\vec{u}^* \cdot \mathrm{grad}^*)\vec{u}^* + \mathrm{grad}^* p^* = \frac{\mu}{\varrho_\infty u_\infty L}\Delta^* \vec{u}^* + \frac{L}{u_\infty^2}\vec{f} \tag{3.4}$$

$$\mathrm{div}^* \vec{u}^* = 0 \tag{3.5}$$

Where the operators grad^*, Δ^* and div^* are related to $\vec{x}^*$ and no longer to $\vec{x}$. Therefore, fluid flow with similar geometries $\Omega^* = L \cdot \Omega$, $L > 0$ are dynamically similar if the corresponding parameters $\mu, u_\infty, \varrho_\infty, L$ and $\vec{g}$ of the two fluid flows do result in identical dimensionless values for the Reynolds- and the Froude-number,

$$Re = \frac{\varrho_\infty u_\infty L}{\mu}, \quad \text{and} \quad Fr = \frac{u_\infty}{\sqrt{L||\vec{g}||}}$$

where $||\vec{g}|| = \sqrt{g_x^2 + g_y^2}$ is the length of vector $\vec{g}$. For the conservation of linear momentum it is also necessary to non-dimensionalize the gravitational forces, hence

$$\vec{g}^* = \frac{L\vec{g}}{u_\infty^2} \left(= \frac{1}{Fr^2} \frac{\vec{g}}{\|\vec{g}\|} \right)$$

Now the incompressible, viscous, time dependent, laminar fluid flow can be described with the non-dimensionalized Navier–Stokes equations.

3.4 Discretisation of the fluid domain

Control volumes and calculation points must be specified to interpolate the velocities and the pressure in the fluid domain. The following section describes how the CFD-code NaSt2D discretises the fluid domain on those control volumes and calculation points to solve the continuous flow problem on a mesh with a limited number of mesh points.

Boundary conditions for the edges of the fluid domain will be given. Additionally the discretisation of the time derivative, for the given transient problem will be illustrated.

For later reference and better illustration, the Navier–Stokes equations will be rewritten, neglecting the (∗) for clarity. The conservation of linear momentum can be given in the following form:

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} &= \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} + f_x, \\ \frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} + f_y, \end{aligned} \tag{3.6}$$

and the continuity equation or conservation of mass becomes:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{3.7}$$

3.4.1 Discretisation of the velocities and the pressure

In two dimensional space a rectangular domain is approximated with an equally spaced mesh in both dimensions. The boundaries (dimension of the cube and the fluid domain, length of (a) and height of (b)) of the two dimensional $\mathcal{R}^2$ domain Ω are $\Omega = [0, a] \times [0, b] \in$

$\mathcal{R}^2$, where the X-, and Y-directions are divided into i_{max} and j_{max} equally spaced cells, respectively. Therefore the size of an individual cell is:

$$\delta x = \frac{\mathrm{a}}{i_{max}} \quad \text{and} \quad \delta y = \frac{\mathrm{b}}{j_{max}}$$

The CFD-code uses a staggered grid as in Figure(3.2) to discretise the Navier–Stokes

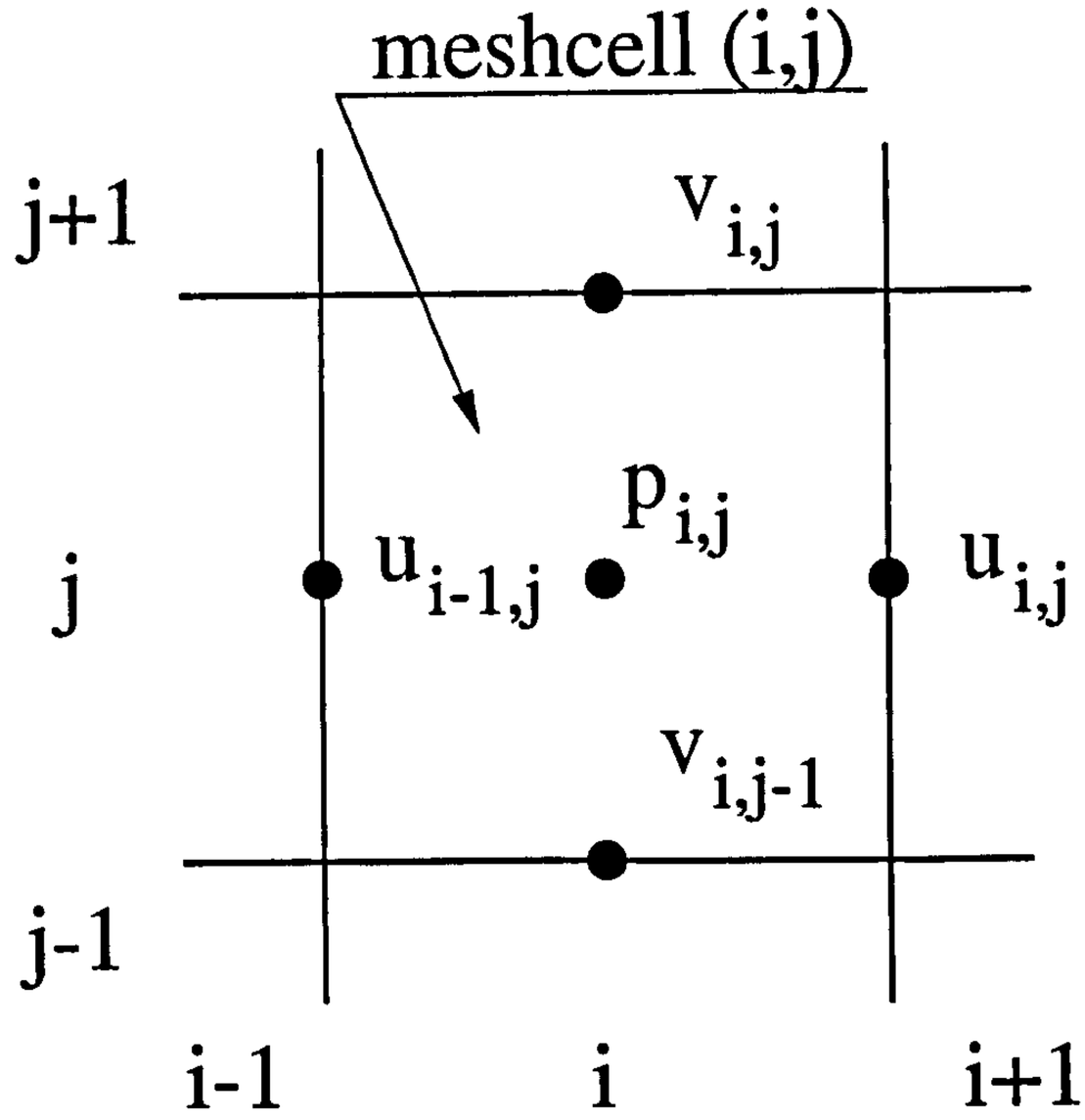


Figure 3.2: Staggered grid.

equations on the mesh. A staggered grid allows the coupling of variables and consequently improves stability constraints. By solving the velocities on different mesh points then the pressure, oscillation in the pressure can be reduced.

The values of u, v and p in mesh-cell (i, j) are shifted with respect to each other by half a cell, therefore not all cell values at a boundary of the domain will be on the boundary. To overcome this problem an additional layer of boundary cells is added to all four sides of the initial mesh domain. The values of u, v and p on the edge of the boundaries are subsequently interpolated.

The continuity equation (Equation (3.7)) is discretised in the middle of cell (i, j), $i = 1, \ldots, i_{max}$, $j = 1, \ldots, j_{max}$. The differential operators $\partial u/\partial x$ and $\partial v/\partial y$ are replaced with central differences

$$\left[\frac{\partial u}{\partial x}\right]_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\delta x}, \qquad \left[\frac{\partial v}{\partial y}\right]_{i,j} = \frac{v_{i,j} - v_{i,j-1}}{\delta y}.$$

The first equation of the conservation of linear momentum (Equation (3.6)) for u is discretised midway between the vertical mesh edges, whereas the second part of equation (3.6) for v is discretised midway between the horizontal mesh edges.

The second derivatives $\partial^2 u/\partial x^2$, $\partial^2 u/\partial y^2$, $\partial^2 v/\partial x^2$ and $\partial^2 v/\partial y^2$, the so called diffusive terms, are replaced by the discrete terms analogous to

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_i = \frac{1}{\delta x}\left(\left[\frac{\partial u}{\partial x}\right]^{cd}_{i+\frac{1}{2}} - \left[\frac{\partial u}{\partial x}\right]^{cd}_{i-\frac{1}{2}}\right) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{\delta x^2},$$

where cd denotes central differences with half step size.

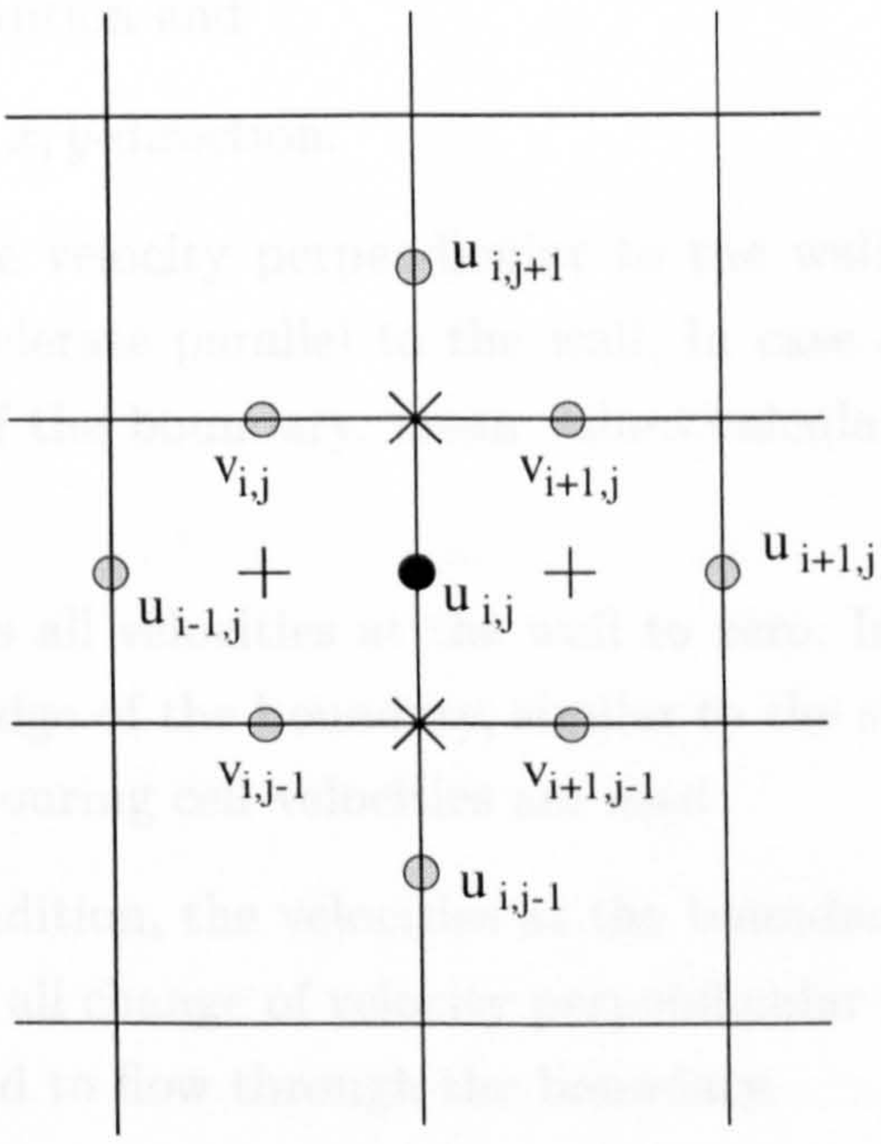


Figure 3.3: Values to be used to discretise u.

The convective terms of Equation (3.6) $\partial(u^2)/\partial x$, $\partial(uv)/\partial x$, $\partial(uv)/\partial y$ and $\partial(v^2)/\partial y$ are more difficult to discretise. As an example the discretisation of $\partial(uv)/\partial y$ at the middle of the right edge of cell i, j will be illustrated (See black dot in Figure(3.3)). To represent this term discrete, suitable values for the product uv can be found at intermediate points, drawn in Figure(3.3) with $(\times)$.

The discretisation of $\partial(u^2)/\partial x$ is also performed with central differences of half step size, using values indicated in Figure(3.3) with $(+)$.

In Appendix Section (A.1.2) the discretisations of all terms in the Navier–Stokes equations

are given. In addition, some comments are listed with regard to the stability conditions of the code.

3.4.2 Boundary conditions

The following boundary conditions can be chosen in NaSt2D

- Slip condition,
- Non slip condition,
- Inflow or outflow condition and
- Periodic condition in x, y-direction.

The slip condition sets the velocity perpendicular to the wall to zero. In addition, the fluid is not allowed to accelerate parallel to the wall. In case of values of velocity being not situated at the edge of the boundary, mean values calculated from the neighbouring cell-velocities are used.

The non slip condition sets all velocities at the wall to zero. In case of values of velocity being not situated at the edge of the boundary, similar to the slip condition, mean values, calculated from the neighbouring cell-velocities are used.

When using the inflow condition, the velocities at the boundary must be set by the user. The outflow condition sets all change of velocity perpendicular to the outflow boundary to zero, thus allowing the fluid to flow through the boundary.

The periodic conditions can only be applied on two opposite sides. On these sides the values of velocity and pressure are the same.

3.4.3 Discretisation of the time derivative

To discretise the derivatives of time $\partial u/\partial t$ and $\partial v/\partial t$ in Equation (3.6) at time t_{n+1} the *Euler* procedure is used, hence differential quotients of first order

$$\left[\frac{\partial u}{\partial t}\right]^{(n+1)} = \frac{u^{(n+1)} - u^{(n)}}{\delta t}, \qquad \left[\frac{\partial v}{\partial t}\right]^{(n+1)} = \frac{v^{(n+1)} - v^{(n)}}{\delta t}, \tag{3.8}$$

where the superscript (n) represents the current point in time.

3.5 The time loop in NaSt2D

With the initial values for u and v, the outer time loop is started at $t = 0$. The time is increased by δt in every iteration loop until the final time t_{end} has been reached. At time step n the value of t_n is known and the value for t_{n+1} is to be calculated. The discretised values of the time derivatives in equation (3.8) are substituted into equation (3.6) resulting in

$$\begin{aligned} u^{(n+1)} &= F - \delta t \frac{\partial p}{\partial x}, \\ v^{(n+1)} &= G - \delta t \frac{\partial p}{\partial y}, \quad \text{with} \end{aligned}$$

$$\begin{aligned} F &= u^{(n)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + f_x \right], \\ G &= v^{(n)} + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} + f_y \right]. \end{aligned}$$

F and G are represented at time level (n) and $\partial p/\partial x$ and $\partial p/\partial y$ are represented at time level $(n+1)$, hence

$$\begin{aligned} u^{(n+1)} &= F^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial x}, \\ v^{(n+1)} &= G^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial y}. \end{aligned}$$

This representation can be interpreted as being explicit in the velocities and implicit in the pressure. With the help of the conservation of mass the pressure at time level $(n+1)$ will be calculated, leading to the Poisson equation for the pressure $p^{(n+1)}$ at the $(n+1)$th time step

$$\frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial^2 p^{(n+1)}}{\partial y^2} = \frac{1}{\Delta t} \left(\frac{\partial F^{(n)}}{\partial x} + \frac{\partial G^{(n)}}{\partial y} \right). \tag{3.9}$$

To solve the Poisson equation, boundary values for the pressure must be specified. Equation (3.9) is finally discretised in space (see Appendix Section (A.1.2)). It can be shown that this structure of computation will finally lead to a system of $(i_{\max} \times j_{\max})$ equations with $(i_{\max} \times j_{\max})$ unknown values $p_{i,j}$, $i = 1, \ldots, i_{\max}$, $j = 1, \ldots, j_{\max}$. This can be solved with an appropriate iterative solution strategy for linear systems of equations. A well known

classical strategy is the Gauss–Seidel algorithm (see e.g. van Kan et al.[58]). This algorithm has quite a slow convergence rate when used on practical problems. The programme NaSt2D uses successive over-relaxation (SOR), where usually the SOR parameter (ω) is specified[3] to be $\omega = 1.7$ (see Stoer et al.[108] for a discussion on the relaxation parameter ω).

This procedure of pressure correction is often referenced as a semi-implicit method for pressure linked equations or Simple procedure (see Patankar et al.[84]).

3.6 Free surface modelling in NaSt2D

The modelling of a free surface includes, apart from the calculation of the velocities and the pressure also the computation of the layout of the fluid domain Ω at every time step. This can be done with particles which are incorporated in the fluid domain. These particles are transported from time step to time step by the known velocities. With the knowledge of the position of these particles the computational domain can be separated into the transient fluid- and the empty "air" domain.

Within NaSt2D it is not intended to simulate two phase flow. The code has been developed to simulate 2D-fluid flow, where empty domains are simulated as a vacuum. Additionally the surface tension must be insignificantly small and the fluid incompressible.

At the start, the initially fluid filled part of the calculation domain is filled with particles. Usually there are between 9 and 16 particles in every fluid filled cell. The number of particles in a single cell is a trade off between computation accuracy and speed. In Section (3.8) some thoughts are discussed regarding a modification to the distribution of particles in the fluid filled domain.

The position of the particles at time t^n is known. With knowledge of the velocities u^{n+1} and v^{n+1} it is possible to calculate the new positions at time t^{n+1}. To increase the accuracy of the free surface tracking, the velocities u, v are calculated for every single particle due to its position on the mesh. The particular velocity of a single particle is calculated based on the four neighbouring velocities. The neighbouring velocities in u-direction for $u(x, y)$

[3] In the parameter file.

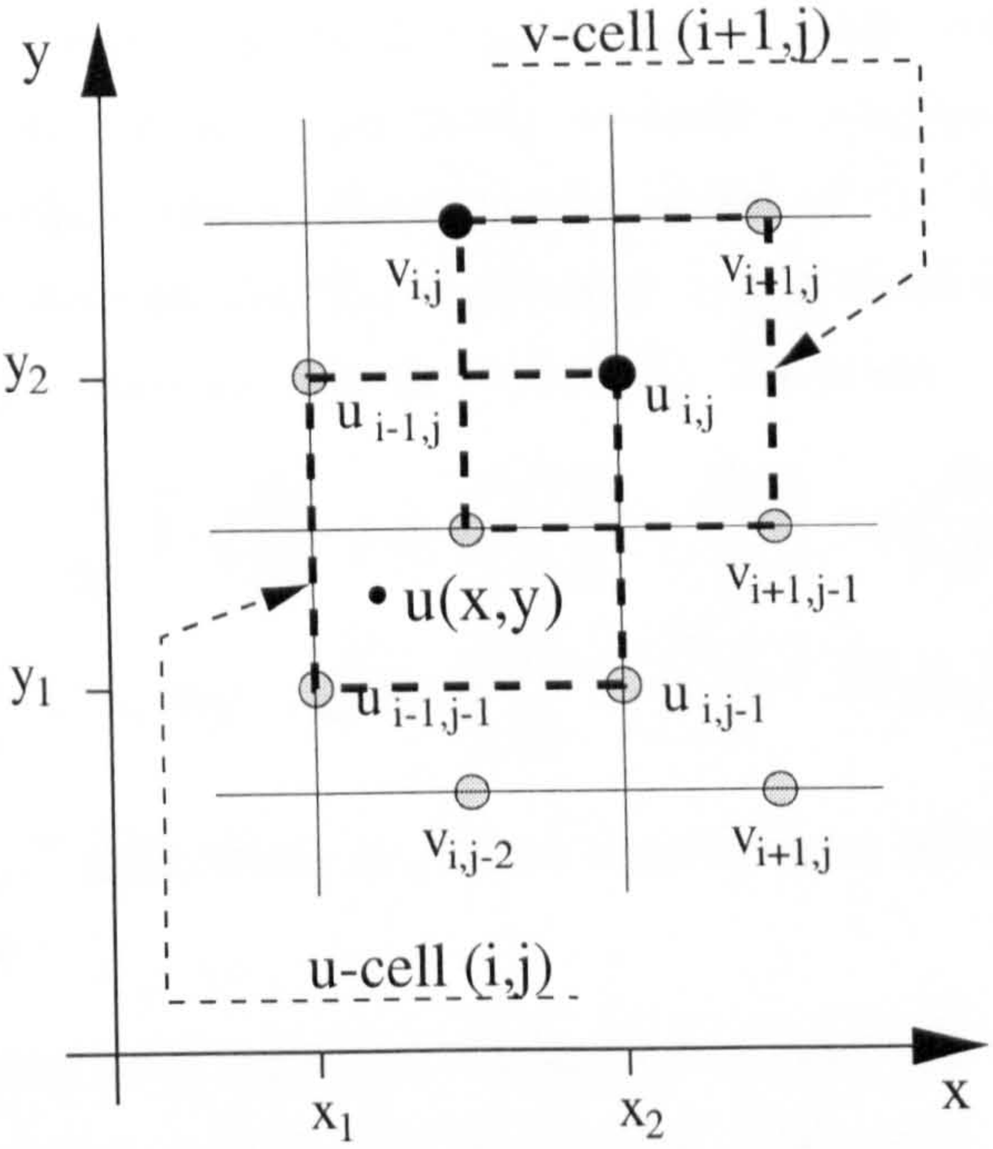


Figure 3.4: Velocity u-cell (i, j).

in Figure(3.4) would be:

$$\begin{aligned} u_{i-1,j-1} &= u_1, \quad u_{i,j-1} = u_2, \\ u_{i-1,j} &= u_3, \quad u_{i,j} = u_4. \end{aligned}$$

The velocity $u(x, y)$ can be calculated using bilinear interpolation

$$\begin{aligned} u(x,y) &= \frac{1}{\delta x \delta y}[(x_2 - x)(y_2 - y)u_1 + (x - x_1)(y_2 - y)u_2 \\ &\quad + (x_2 - x)(y - y_1)u_3 + (x - x_1)(y - y_1)u_4]. \end{aligned}$$

Subsequently the new positions must be evaluated. There exist three categories:

- Empty cells: Cells without particles,
- Inner cells: Cells with particles and no border to empty cells,
- Surface cells: Cells with particles and border to empty cells.

Generally, the surface cells approximate the line of the surface. For a more precise knowledge of the surface the particles themselves must be observed[4]. The mathematical model of the discretisation of the free surface is based on the definition of a fluid stress vector on the free surface. The physical condition on the boundary layer of two different fluids is

[4]See Section (3.8).

characterised by the force, acting on the boundary layer. Thes boundary conditions on the mesh cells are based on the physical law that the force acting on an interface is directed perpendicular to the interface and is direct proportional to the mean curvature of the interface. Based on these relations and the limitation to one fluid in vacuum, discretised in two dimensions following equations can be evaluated. They are

$$p - \frac{2}{Re}\left(n_x^2\frac{\partial u}{\partial x} + n_x n_y\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + n_y^2\frac{\partial v}{\partial y}\right) = 0 \qquad (3.10)$$

$$2n_x m_x\frac{\partial u}{\partial x} + (n_x m_y + n_y m_y)\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + 2n_y m_y\frac{\partial v}{\partial y} = 0. \qquad (3.11)$$

The expressions $\vec{n}(n_x\ n_y)^T$ and $\vec{m}(m_x\ m_y)^T$ are unit vectors normal and tangential to the free surface, respectively.

Several different flow situations must be recognised to set the boundary conditions (see Equations(3.10, 3.11)) of the surface cells accurately. They are

1. Surface cells with one empty neighbouring cell;
   (See Figure (3.5))
2. Surface cells with two empty neighbouring cells, the empty cells being at one corner of the cell;
3. Surface cells with two empty neighbouring cells, the empty cells being at opposite sides of the fluid cell;
4. Surface cells with three empty neighbouring cells and
5. Surface cells with four empty neighbouring cells.

The general calculation procedure will be illustrated for the case that one neighbouring cell is empty.
The free surface is assumed to be nearly parallel to the mesh lines at a cell. Therefore either n_y and m_x or n_x and m_y are very small, hence equation (3.10) will be

$$p = \frac{2}{Re}\frac{\partial u}{\partial x} \qquad \text{or} \qquad p = \frac{2}{Re}\frac{\partial v}{\partial y}$$

with the first equation representing the vertical and the second equation representing the horizontal boundaries. Equation (3.11) reduces to

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

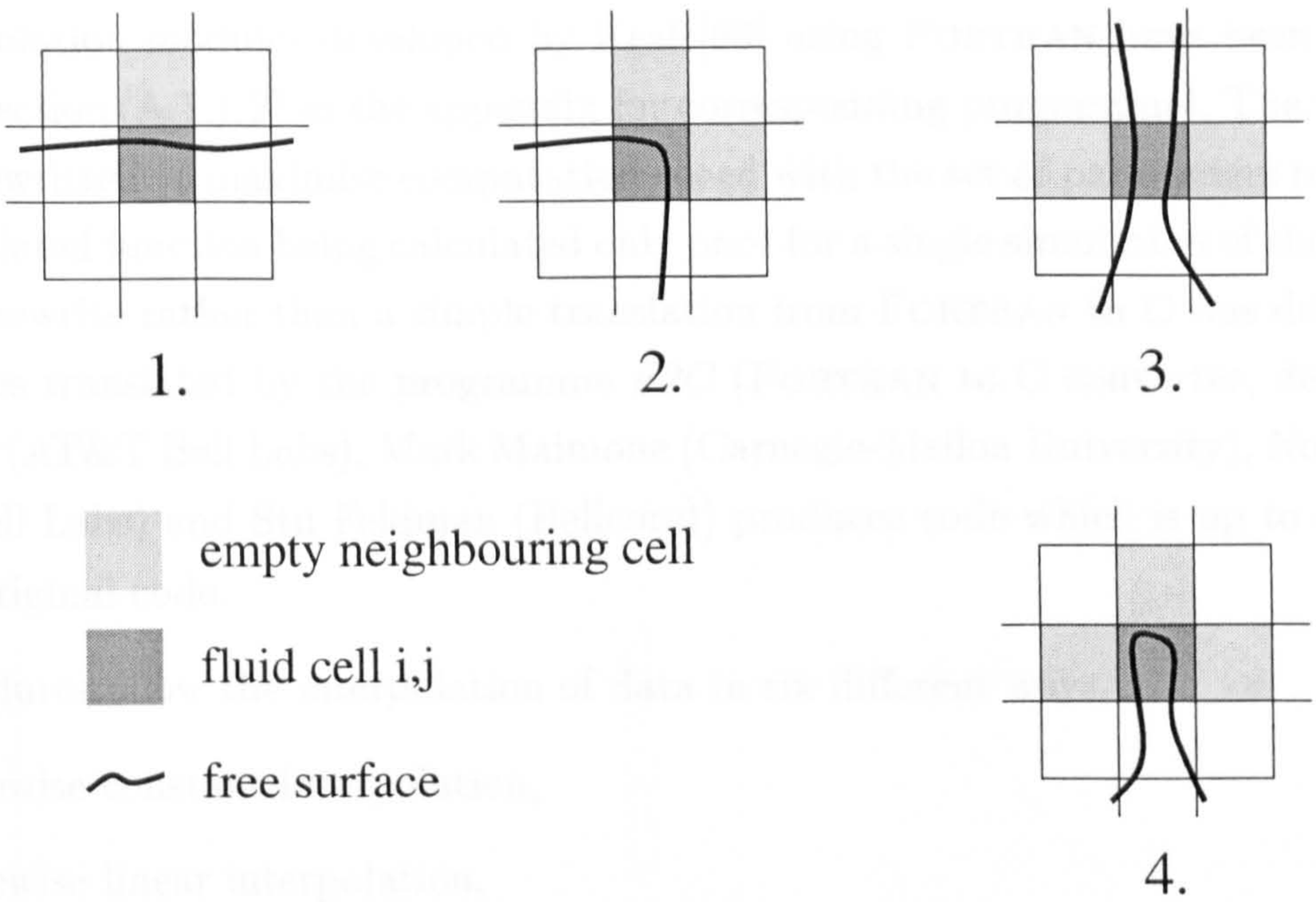


Figure 3.5: Free surface boundary situations.

The discretisation is dependent on the flag[5] of the cell, which identifies the empty sides of the surface cell.

3.7 Implementation of variable accelerations

The original version of NaSt2D was not capable of modelling and simulating a fluid flow in an acceleration field with varying forces. When modelling and simulating the complete experimental rig, i.e. modelling both mechanical and fluid dynamics, this problem had to be solved. The general idea was to implement sub-procedures which would:

- Supply an acceleration profile in terms of time-acceleration pairs, readable from an external data file.
- Supply a variety of interpolation schemes ranging from simple to sophisticated.
- Allow the selection of interpolation schemes to calculate the acceleration at different times.

[5] NaSt2D uses a flag register for every cell to verify the identity (empty-, fluid- or surface cell) of the cell.

The interpolation modules developed by Kraft[63] using FORTRAN have been translated to C (see Section (A.1.1.2) in the appendix for corresponding programme). The whole code has been rewritten to maximise computation speed with the set of parameters representing the interpolated function being calculated only once for a single simulation of the fluid flow. This code rewrite rather than a simple translation from FORTRAN to C was done because programmes translated by the programme *F2C* (FORTRAN to C converter, developed by David Gay (AT&T Bell Labs), Mark Maimone (Carnegie-Mellon University), Norm Schryer (AT&T Bell Labs) and Stu Feldman (Bellcore)) produces code which is up to 60% slower than the original code.

The procedures allow the interpolation of data in six different ways.

- Piecewise constant interpolation,
- Piecewise linear interpolation,
- Piecewise cubic interpolation,
- Piecewise cubic exponential spline interpolation,
- Piecewise cubic exponential spline interpolation, with tension parameters set by the user,
- Akima's interpolation.

Further details can be found in Kraft[63], de Boor[11] and Hilberg[49].

3.8 Accuracy of NAST2D

In order to gain confidence in the representation of the free surface fluid motion using NAST2D various experiments were undertaken to verify the accuracy of the finite difference code NAST2D. In particular, a comparison has been undertaken between NAST2D and an off the shelf, commercial CFD-code. These calculations have been accompanied by experimental verification. Additionally some numerical investigations of NAST2D have been made.

Initially a well known commercial CFD-code, FIDAP[30], was used to calculate the fluid motion for a given acceleration of the container, the results being compared to the predictions from NAST2D.

The comparison must be performed with the same computational model for both CFD-codes. The computational model of the physical process is limited by the differing limitations of the CFD-codes. On the one hand NASt2D is not capable of modelling surface tension, it is assumed to be zero. On the other hand FIDAP can only perform linear interpolation for the approximation of an external force acting on a rectangular container filled with fluid. Therefore the fluid must have a very low surface tension. The liquid chosen for the comparison is glycerin.

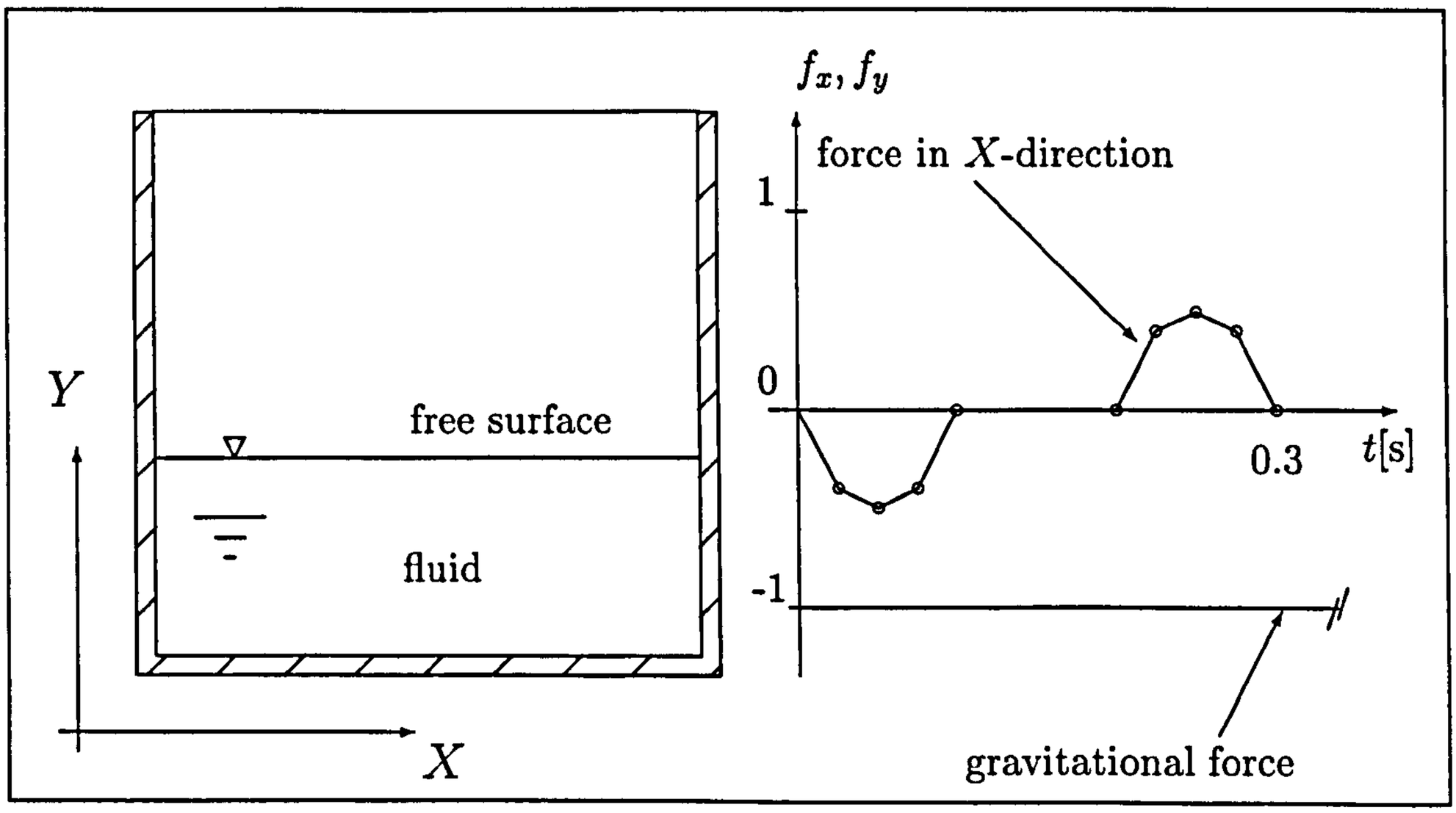


Figure 3.6: Initial layout of the container and the profile of the applied forces.

In Figure(3.6) the general, two dimensional layout of the container with the fluid region and the data for the forces acting on the fluid are given.

The shown force profiles indicate a time independent gravitational force in the Y-direction, and a time dependent accelerating force in the X-direction.

Due to the behaviour of the free surface several limitations had to be applied. Within FIDAP, the mesh had to be spaced equally in X, Y-direction, resulting in a mesh consisting of 30 fluid filled cells in the X-direction and 12 fluid filled cells in the Y-direction. Additionally the surface tension had to be specified, very small but greater than zero. Otherwise the liquid tended to stick at the boundaries and the distinct oscillating behaviour could not be observed at the boundaries.

The external force impact on the fluid region finishes at $t = 0.3$[s]. Afterwards the distur-

bance of the free surface will gradually decline. The calculations are performed in the time interval ($0.0 \leq t \leq 1.0$[s]). Within this time the maximum amplitude is reached.

Indicators for the movement of the free surface are the surface heights at the left and right boundary, changing with time. The transient amplitude heights of the free surface at the left and right boundary have been observed and plotted in Figures(3.7), (3.8). They show a distinct oscillating behaviour.

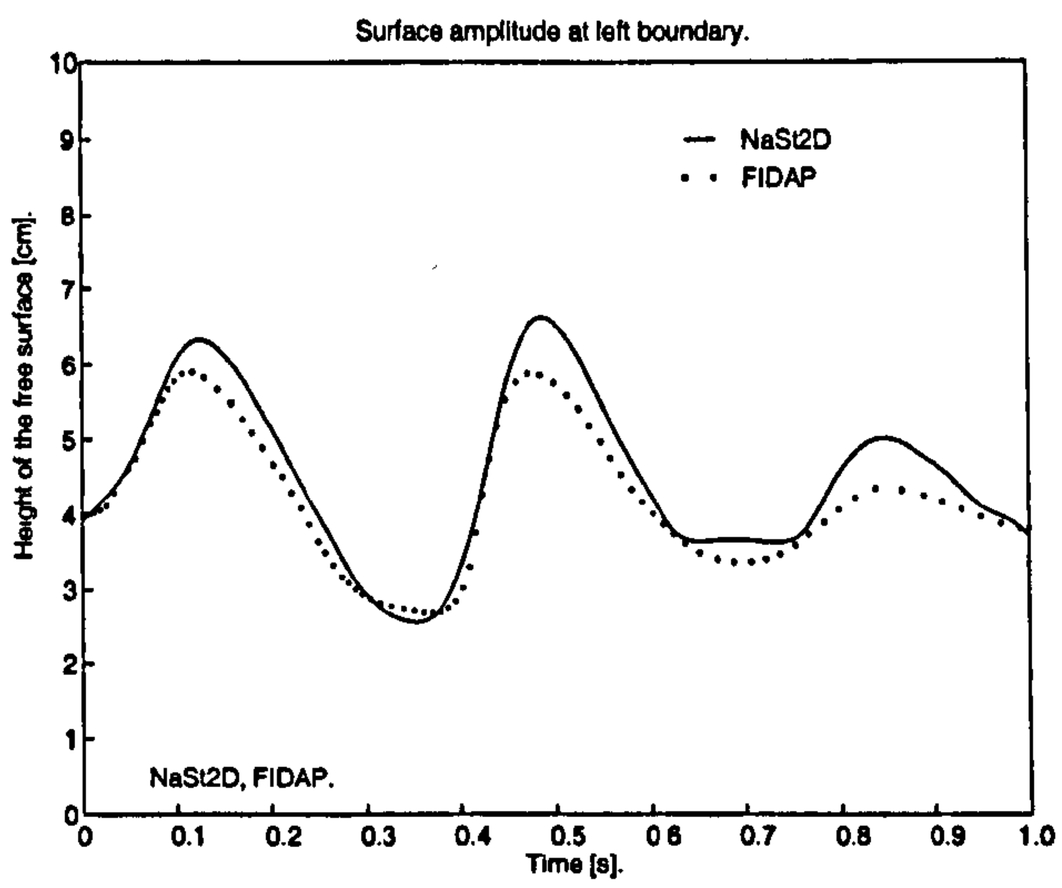


Figure 3.7: Amplitude at left boundary.

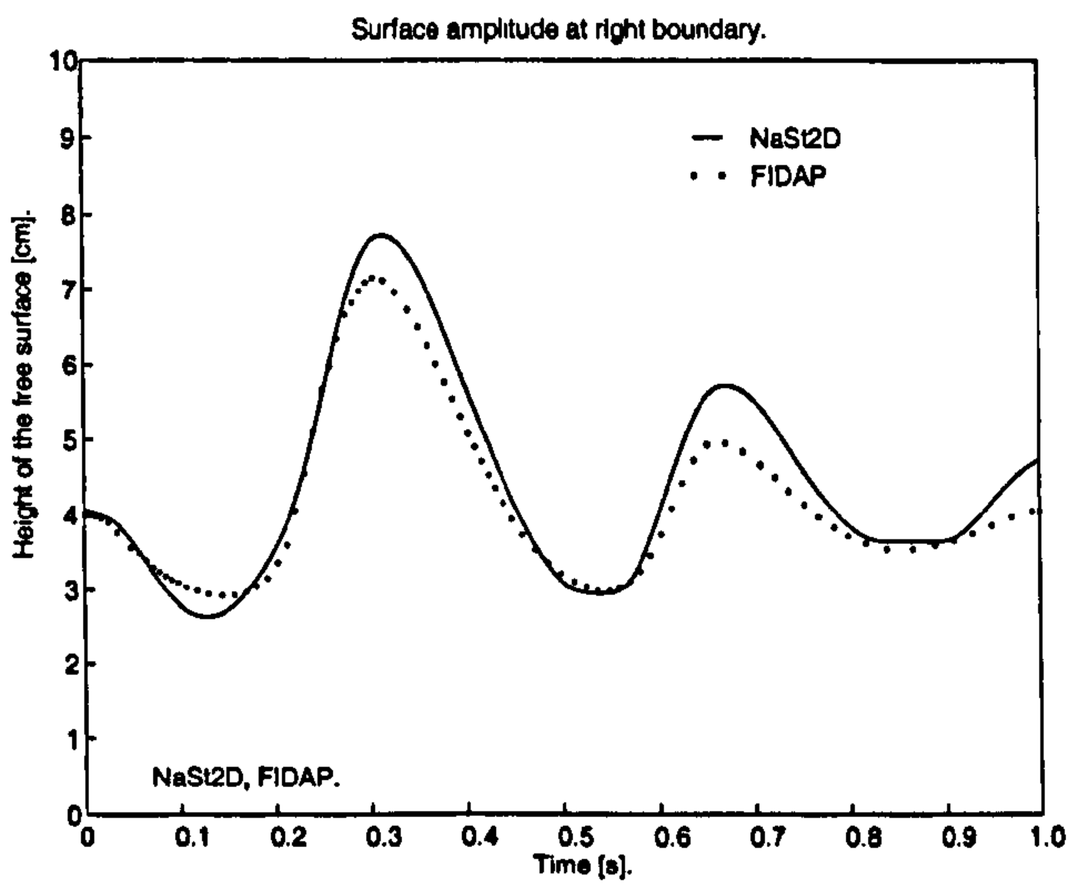


Figure 3.8: Amplitude at right boundary.

The different plots in Figures (3.7) and (3.8) for NaSt2D and FIDAP are quite similar. The main feature, the wavelength is nearly the same over the whole time-period. The major difference is in the height of the maxima and minima. It can be observed that the sloshing within FIDAP is always smaller than within NaSt2D. This is due to the representation and tracking of the free surface within FIDAP using spines. They are not as flexible as particles and can not follow large free surface motion particularly well. In addition the effects of the very small, but existing surface tension is another reason for the lower sloshing of the surface within FIDAP.

In these early stages of the research a milling machine was used to simulate the motion of a small scale model. An experiment was undertaken to simulate the motion of a fluid within a 100[mm]×100[mm]×100[mm] cube accelerated by the working bed of the milling machine. In order to compare predicted results with reality, the motion has been captured with a video. Within this video, the position of the cube has been observed against a static grid in the background. This information has been used to calculate the motion of the free surface with NaSt2D and to compare it with the motion within the video.

These experiments illustrated that, in most cases, even the amplitudes calculated by NaSt2D are not as high as in reality. Additionally, the higher amplitudes predicted by NaSt2D compared to FIDAP act as an additional safety margin against sloshing within the optimal control calculations.

The calculations have shown that the simple finite difference code NaSt2D is capable of obtaining the same results as the commercial CFD-code FIDAP for the specified sloshing case. Further investigations are necessary to get a broader view of the differences between the codes. It is clear that FIDAP is the more sophisticated CFD-code with a capability of solving a much greater variety of flow situations, whereas NaSt2D can be faster and as good for a limited number of problems.

In order to increase the accuracy of the finite difference code NaSt2D several modifications have been proposed and partially integrated into NaSt2D. The code tracks the free surface with particles which are initially uniformly distributed over the whole fluid domain. Some investigations have been undertaken to estimate the influence of non-uniform particle distributions.

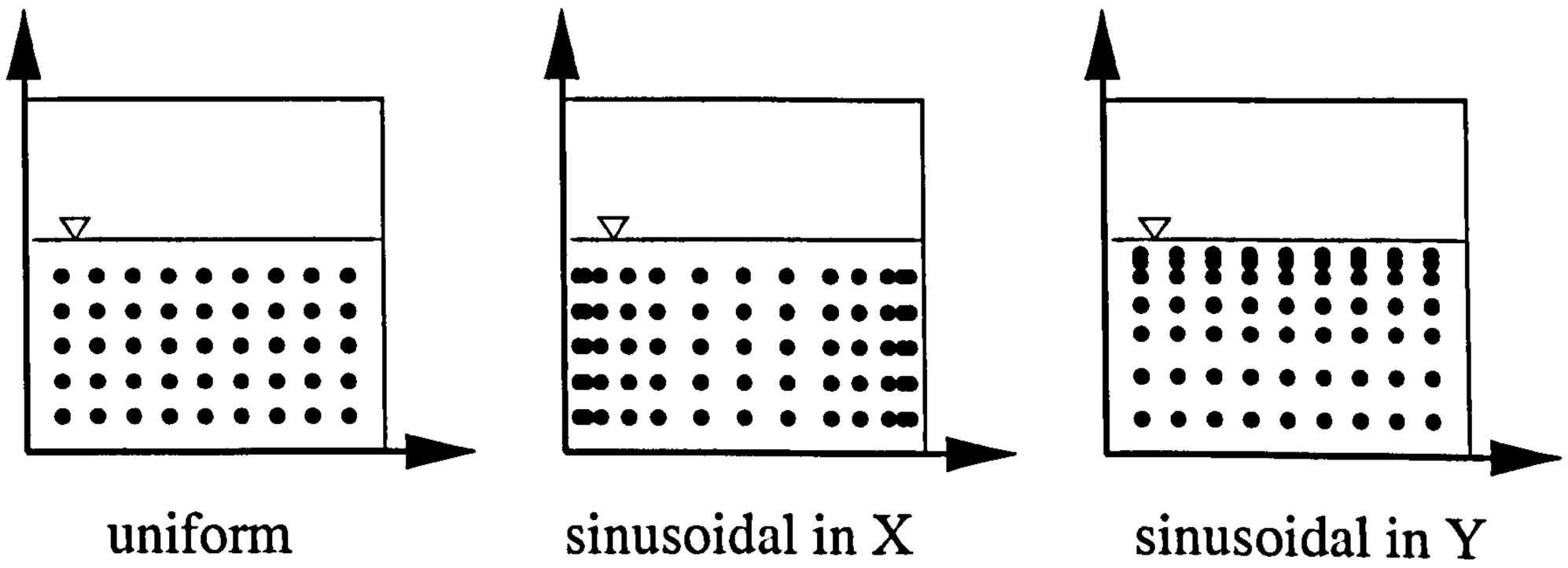


Figure 3.9: Variation of particle distribution.

In Figure(3.9) the investigated configurations, uniform distribution, sinusoidal distribution in the X-direction of the cube and sinusoidal distribution in the Y-direction of the cube are shown schematically. It has been observed that none of these special configurations resulted in a more accurate or faster calculation. It can be shown that the uniform distribution has advantages over the other distributions, because in the sinusoidal cases it can happen that some fluid cells have no particles. This situation can not be tolerated within the calculation.

In order to increase the accuracy of the surface amplitude results, two different modes have been programmed. The first mode is concerned with the CFD calculation within the optimisation, hence the principal aim is to run the programme as quickly as possible. Thus the information should be supplied with as little additional computation as possible. This has been achieved with the focus on the particles. The Y-coordinate of the particles in a specified boundary layer is stored and analysed, with only the Y-coordinate of the highest particle on both sides being stored for later processing.

The second mode is concerned with the final calculation and generation of fluid motion data. Here the amount of information is important and the overall time for the calculation is of secondary interest. A second mesh was designed to cover the fluid domain. This mesh has no influence on the calculation. For surface tracking purposes the mesh cell size in X-direction is normally the same as the calculation mesh cell size. The spacing in the Y-direction is far more interesting and can be divided into user defined cell heights.

To understand the behaviour of the code several parameter variations were undertaken to evaluate their influence on the motion of the fluid.

The following influences have been investigated:

- Mesh spacing in X- and Y-direction,
- Filling state of surface cell,
- Variation of magnitude of disturbance
- Variation of time step control.

The theoretical result for any given problem of sloshing in an open fluid filled container is the return of the free surface to its initial surface height, a result which should be achieved if no fluid sloshes over the vertical boundaries.

However, the calculated results obtained from NASt2D proved that some variation within the amount of fluid occurred. This difference was always negative, being a loss of fluid. Based upon the experimental results using the investigated influences the following summary can be stated.

The deviation is proportional to the magnitude of the motion. A greater magnitude of motion gives greater variations in the velocity and pressure in the fluid domain, hence greater changes take place within any given time-step.

The more accurate the code is, in terms of time-step and domain decomposition, the smaller is the deviation of the calculated and theoretical results. The accuracy of the code is more sensitive to a broad mesh spacing in the X-direction than in the Y-direction. In general, NaSt2D is capable of performing precise, accurate and practically comparable results. The accuracy of the results is dependant upon the configuration of the calculations (mesh-spacing, time step control) and are therefore proportional to calculation time. The better the results, the longer the calculation being taken to simulate the fluid flow. It has been observed that it is possible to achieve results within a bandwidth of 5% deviation in a moderate calculation time of about 60 seconds.

The initial filling height of the surface cells has no influence on the accuracy of the calculation. It is only an important parameter for the visualisation of the sloshing. The variation of the magnitude of disturbance revealed high sensitivity of the computational algorithm. The correlation between disturbance and sloshing behaviour is further discussed in Section (6.1).

Additionally, it has been observed, that the maximum amplitude of the sloshing motion is less influenced by these accuracy features. Since the maximum amplitude is the main criteria for the selection of an optimal trajectory, the author is very confident in the ability of NaSt2D to realistically simulate sloshing behaviour.

3.9 Summary of fluid modelling

This chapter has presented the theory of modelling of fluid flow, in particular modelling the special case of a fluid system with free surface motion.

The specific approach, NaSt2D, by Griebel et al.[45] for modelling a fluid motion with free surface has been described. To do so, basic fluid-mechanical theory has been illustrated. This included the representation of the Navier–Stokes equations and the theory of non-dimensionalisation in terms of dynamic similarity of fluid flows.

The particular discretisation of the rectangular calculation domain and the time derivative are given. The calculation of the time step, using the SIMPLE pressure correction algorithm and successive over-relaxation has been illustrated.

Some thoughts have been given to the representation and tracking of the free surface

within NASt2D. In order to integrate the fluid model within the optimisation of the cart trajectory, a module for using time dependent variable acceleration has been investigated and written for the code. It has been shown that the finite difference code NASt2D has all capabilities to simulate the required free surface motion. The specifications investigated and developed in Leonpacher[68] for the selection of the CFD-code have been adequate.

In order to justify the use of the finite difference code NASt2D to model the behaviour of the free surface motion a final focus has been on the accuracy of the code. From these findings, the following can be concluded:

- NASt2D is very much capable of simulating a free surface flow.
- The accuracy of the obtained results is dependent on the fineness of the rectangular grid and parameters of the code.
- The correlation between accuracy of the results and computation time of the algorithm is very advantageous, because the there is a potential to reduce the computation time with just a small loss of accuracy.

These results have been the originator to base the further investigations of this work on the finite difference, computational fluid dynamics code NASt2D.

3.10 Nomenclature

a, b	dimension of the fluid domain
f, $\vec{f}(f_x, f_y, f_z)$	force, force vector with cartesian components (f_x, f_y, f_z)
Fr	Froude number
g	natural gravity
i, j	position within the CFD mesh in X- and Y-direction
i_{max}, j_{max}	number of fluid cells in X- and Y-direction
L	characteristic length
$\vec{m}(m_x\ m_y)^T$	unit vector tangential to the free surface
(n)	current state of time
$(n+1)$	future state of time $t^{(n+1)} = t^{(n)} + \Delta t$
$\vec{n}(n_x\ n_y)^T$	unit vector normal to the free surface
N	dimension
p, p_∞	pressure, characteristic pressure
$\mathcal{R}$	domain of real numbers
Re	Reynolds number
S	scaling factor
t	time
$\boldsymbol{u}, \vec{u}(u, v, w)$	velocity vector with cartesian components u, v, w
u_∞	characteristic velocity, (within the calculations a mean velocity of the fluid within the cube has been used)
$\boldsymbol{x}, \vec{x}(x, y, z)$	position vector with cartesian components x, y, z
ϱ, ϱ_∞	density, characteristic density
μ	dynamic viscosity
ω	relaxation factor
Ω	domain, volume

Chapter 4

Overall model of the fluid-mechanic system

This chapter is concerned with the completion of the model process and the combination of the fluid and the structural dynamics.

In order to do so a standard approach for modelling and solving the motion of the mechanical system is introduced. In particular the mechanical system is modelled using ordinary differential equations and solved with a Runge-Kutta scheme.

Furthermore, two different methods for the combination of the models of the solid and the fluid mechanical system are studied. These methods are the approaches of weakly and strongly coupled systems. The investigations of these coupling approaches results in the selection of the appropriate scheme and the implementation of the scheme by the author.

Finally the conclusion presents an overview on the properties and characteristics of the studied subjects within this chapter.

4.1 Modelling of the mechanical system

The modelling of the mechanical system is based on the simplifications introduced in Section (1.2.1). Due to the fact that observation of the fluid motion within a real world storehouse environment is very expensive, a small scale experimental rig has been used to evaluate experimental data. This rig can be accurately moved in two dimensional upright

space, and as such reflects a motion domain similar to the main motion domain of a warehouse. The horizontal motion of the rig can be illustrated as in Figure(4.1). The fluid container is linked to a transmission belt and runs on a fixed track. This general model can

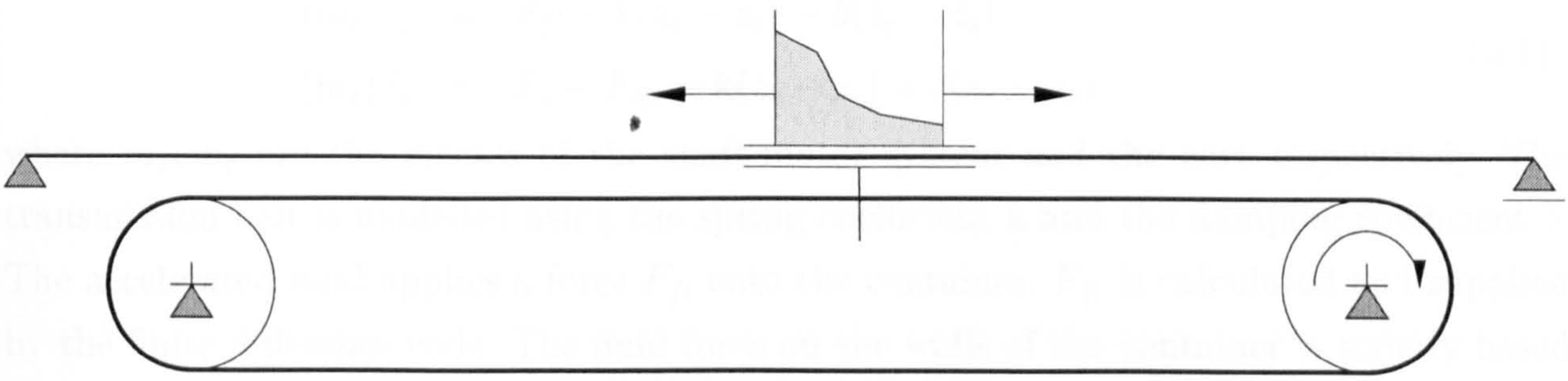


Figure 4.1: Illustration of the mechanical system.

be transformed into a schematic model to represent forces, motion, weights and friction.

4.1.1 Specific model of the experimental rig

The cart is modelled as a cube, fixed with a spring and a damper to a device which is sliding with friction over ground. The friction between the device and the ground represents the friction on the track and the friction of the turning shafts. The spring and the damper represent the elasticity in the transmission belt.

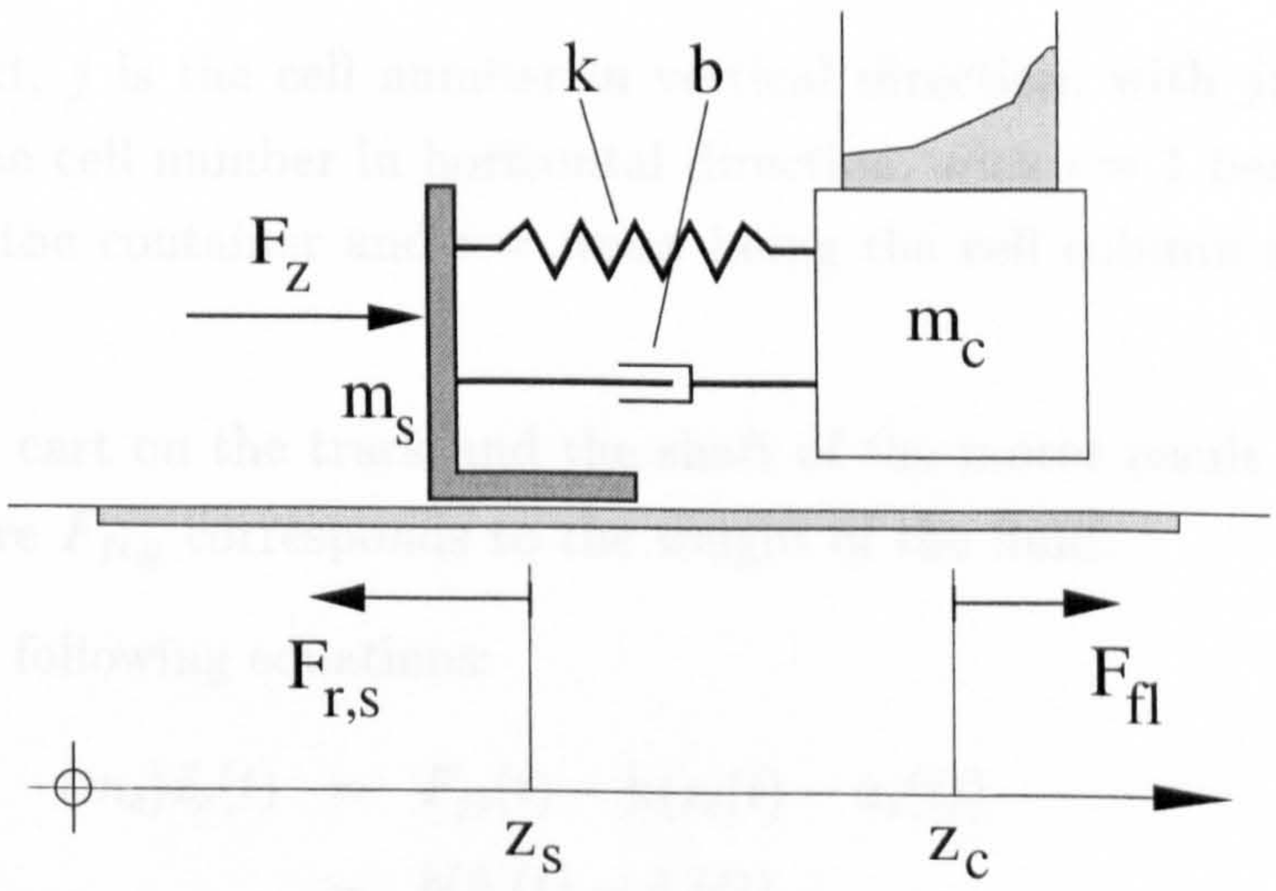


Figure 4.2: Model of the mechanical system.

The following equations can be formulated to model the dynamic behaviour of the mechanical system:

$$\begin{aligned}(m_c)\ddot{z}_c &= F_{fl} - \mathrm{k}(z_c - z_s) - b(\dot{z}_c - \dot{z}_s) \\ (m_s)\ddot{z}_s &= F_z - F_{r,s} + \mathrm{k}(z_c - z_s) + b(\dot{z}_c - \dot{z}_s)\end{aligned} \tag{4.1}$$

where m_s, m_c are the masses of the shaft-motor system and the cart respectively. The transmission belt is modelled using the spring coefficient k and the damping coefficient b. The accelerated fluid applies a force F_{fl} onto the container. F_{fl} is calculated and supplied by the finite difference code. The fluid force on the walls of the container is strictly based on the pressure difference between the left and the right boundary of the fluid domain. The following equation is used to calculate F_{fl} from the dimensionless variables of pressure p, vertical cellsize Δy and depth of the container z :

$$F_{fl} = \sum_{j=1}^{jmax} \left[(p_{imax,j} - p_{1,j})\, \Delta y\, z\, \frac{Re^2\, \mu^2}{\varrho} \right] \tag{4.2}$$

using the nondimensionalized variables illustrated within equation3.3, with use of

$$p_\infty, = 0 \qquad \text{and} \qquad Re = \frac{\varrho_\infty u_\infty L}{\mu},$$

equation (4.2) can be reduced to

$$F = p \cdot A.$$

Within this context, j is the cell number in vertical direction, with $jmax$ being the top cell row and i is the cell number in horizontal direction, with $i = 1$ being the cell column at the left wall of the container and $i = imax$ being the cell column at the right wall of the container.

The friction of the cart on the track and the shaft of the motor result in $F_{r,s} = \mu_s((m_s + m_c)g + F_{fl,y})$, where $F_{fl,y}$ corresponds to the weight of the fluid.

This results in the following equations:

$$\begin{aligned}(m_c)\ddot{z}_c(t) &= F_{fl}(t) - \mathrm{k}(z_c(t) - z_s(t)) \\ &- b(\dot{z}_c(t) - \dot{z}_s(t)) \\ (m_s)\ddot{z}_s(t) &= F_z(t) - \mu_s((m_s + m_c)g + F_{fl,y}(t)) \\ &+ \mathrm{k}(z_c(t) - z_s(t)) + b(\dot{z}_c(t) - \dot{z}_s(t))\end{aligned}$$

These equations are used to form a system of first order ordinary differential equations (ODEs) with $z_1 = z_c$, $z_2 = \dot{z}_c$, $z_3 = z_s$ and $z_4 = \dot{z}_s$:

$$\begin{aligned}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= \frac{1}{(m_c)}(F_{fl} - \mathrm{k}(z_1 - z_3) - b(z_2 - z_4) \\
\dot{z}_3 &= z_4 \\
\dot{z}_4 &= \frac{1}{m_s}(F_z - \mu_s((m_s + m_c)g + F_{fl,y}) + \mathrm{k}(z_1 - z_3) + b(z_2 - z_4)
\end{aligned} \tag{4.3}$$

This system of ODEs must be solved to know the motion of the shaft and the cart with the fluid container, based on the external force on the system and the pressure force of the fluid at every single time step.

4.1.2 Mechanical motion solved with Runge–Kutta 5(4)[0]

The motion of the mechanical system is generally regarded as an initial value problem. There are many algorithms capable of solving such a problem. They are classified into 'start'-techniques and 'follower'-techniques. The 'start'-techniques like Euler–Cauchy or Runge–Kutta solve a system of ordinary differential equations as the name implies from the start. Whereas the 'follower'-techniques like the Milne-procedure rest upon an initial solution of a 'start'-technique to calculate additional points with lower computational costs.

In general, the system of ordinary differential equations according to equations (4.3) can be formulated as an initial value problem

$$\dot{z}(t) = \phi(t, u(t), z(t)), \qquad z(t_0) = z_0, \tag{4.4}$$

using the time derivative $\dot{z}(t)$ and initial conditions z_0.

To solve the ordinary differential equations of the problem of cart motion with a sloshing fluid, a 5-ary Runge–Kutta procedure according to Fehlberg[27] has been implemented[1]. This is well proven because of the accuracy, robustness, general simplicity and wide use (see [72][83][104]) of these procedures within mathematical programming.

The general form of this Runge-Kutta procedure with integrated step control can be formulated as

$$\phi_i = \phi\left(t_k + \alpha_i h_k, z_k + h_k \sum_{j=1}^{i-1} \beta_{ij}\phi_j\right), \quad i = 1, \ldots, q, \tag{4.5}$$

$$z_{k+1} = z_k + h_k \sum_{i=1}^{s} \gamma_i \phi_i. \tag{4.6}$$

These procedures consist of equations of order p and q, where $q \geq p$ (in general $q = p+1$). The coefficients α_i, β_{ij} and γ_i characterise the Fehlberg Runge-Kutta equations. See Tables (4.1, 4.2) for the values of these coefficients.

In equations (4.5) and (4.6) h_k is the step size, z_k and z_{k+1} are the approximation to the solution of equation (4.4) at times t_k and $t_{k+1} = t_k + h_k$. The step size h_k is set by a comparison of the actual error to the admissible error. To do so, $\delta(t_k)$ is calculated using

[0]Runge–Kutta 5(4)means, that the solver is of order 5 with an error correction procedure of order 4.
[1]See Section A.1 for C-implementation.

the coefficient set $\bar{\gamma}_i$ and the solution ϕ_i of equation (4.5)

$$\delta(t_k) = h_k \sum_{i=1}^{s} \bar{\gamma}_i \phi_i.$$

This actual error is compared with the admissible error $\tau(t_k) \geq \theta$ using the following approach

$$h_k = \min \left[h_{\max}, \alpha h_{k-1} \left(\frac{\tau}{\delta_k} \right)^{\frac{1}{p+1}} \right], \tag{4.7}$$

where the $h_{\max} = \frac{1}{16}(t_f - t_0)$, $\alpha \in (0,1]$[2] and $p = 4$.

i	α_i	β_{ij}				
1	0	0				
2	$\frac{1}{4}$	$\frac{1}{4}$				
3	$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$			
4	$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$		
5	1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$	
6	$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$

Table 4.1: Coefficients for Runge–Kutta RK5(4) formulation

4.2 Verification of the mechanical system modelling

The dynamics of the mechanical system within the experimental rig could only be measured approximately. The main problem of the specifications of the dynamics is the separation of the dynamics of the servo motor input side and the servo motor output side.

[2] In general MATLAB[72] uses $\alpha = 0.8$.

i	1	2	3	4	5	6
γ_i	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	
$\bar{\gamma}_i$	$-\frac{1}{360}$	0	$\frac{128}{4275}$	$\frac{2197}{75240}$	$-\frac{1}{50}$	$-\frac{2}{55}$

Table 4.2: Weighting–coefficients of the Runge–Kutta procedure RK5(4) by Fehlberg

The main components of the dynamic system of the controller and actuator of the experimental rig are shown in Figure(4.3). The values of the gains (K, K_d, K_a, K_v, K_p) are studied in Section (4.2.1). It was possible to observe the following data:

- Command input at the motion controller.
  (qualitatively)
- Encoder counts from the encoder mounted on the servo motor.
  (qualitatively)
- Visual video data from the motion of the cart and the fluid.
  (quantitatively)

The information from the command input, if fully decoupled from other influences can be given accurately. The information from the encoder is a mixture of several physical influences. These influences correspond to the responses of the motion controller, the amplifier, the servo motor and the plant.

The visual video data from the motion of the cart and the fluid can supply a quantitative answer to the general questions of:

- Has the correct cart motion been performed?
  (Start point, final point)
- Has the fluid motion responded in the calculated manner?
  (Sloshing behaviour, amplitudes of the fluid, frequency of the fluid)

The motion of the cart and the fluid is very quick. The optimal motion needs only 0.5[s] from the start point to the destination. A standard video capture allows only 26 pictures per second recording time. That means that there are only 13 pictures available for the whole movement. The results obtained from these captures are therefore not very accurate.

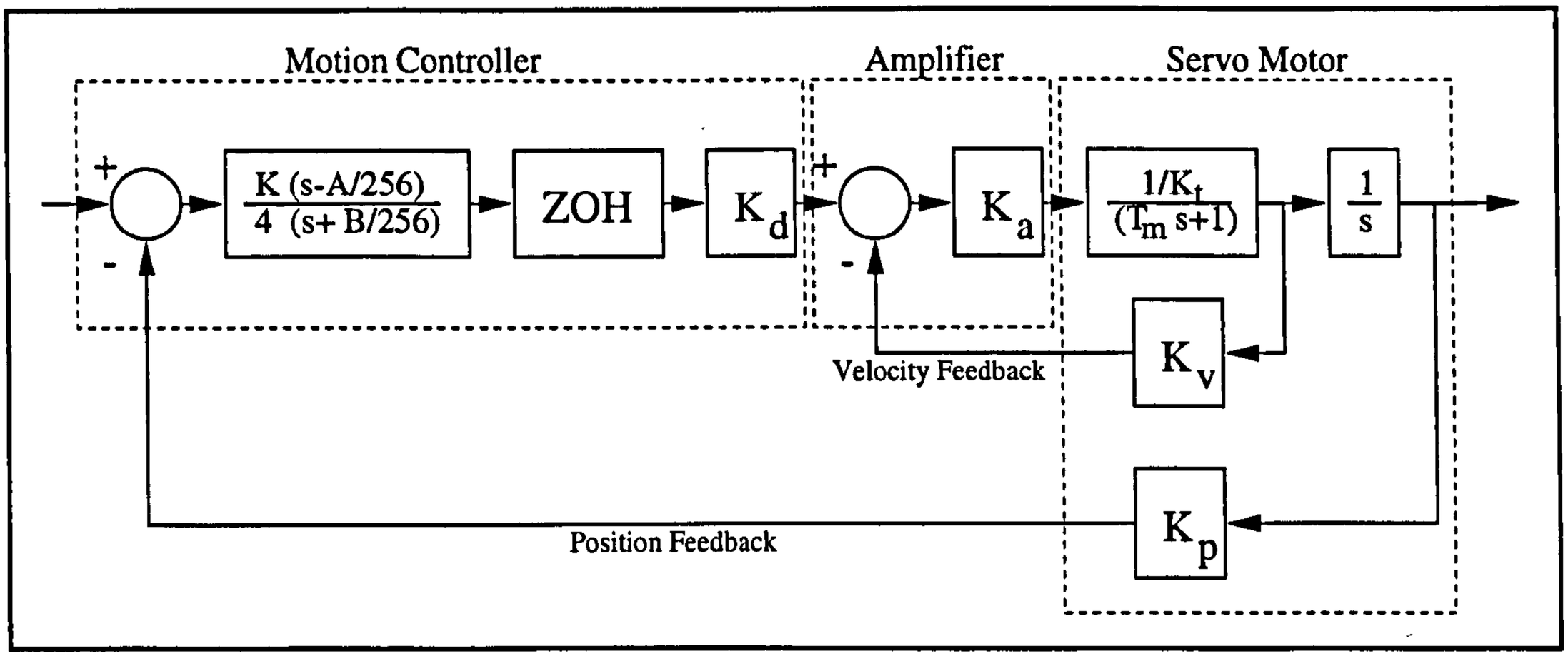


Figure 4.3: Block diagram of servo-rig control scheme.

But the general questions, which have been given before, can be answered uniquely due to their nature of having only two results: true or false.

The controller, amplifier and servo motor system used in this experimental rig has been used before. Osypiw[82] has modelled and evaluated the characteristic dynamics of the system. This information enables work to be undertaken on the specific problem of fluid container motion. To use his informations, a computational model of the controller, amplifier and servo motor has been built with the help of MATLAB and SIMULINK.

4.2.1 SIMULINK modelling of the servo control system

The first model built was for the complete control of the servo motor. This includes the motion controller, the amplifier and the servo motor dynamics. Figure(4.4) shows the layout of this dynamical system.

Several values are set by the hardware of the system (cf. to Figure(4.3)). The digital analogue converter (DAC) generates a voltage (± 5 volts) proportional to the filter's output, the gain K_d is calculated by:

$$K_d = \frac{10}{256} = \frac{2 \cdot \text{voltage output of DAC}}{\text{number of bits for DAC}} = 0.039$$

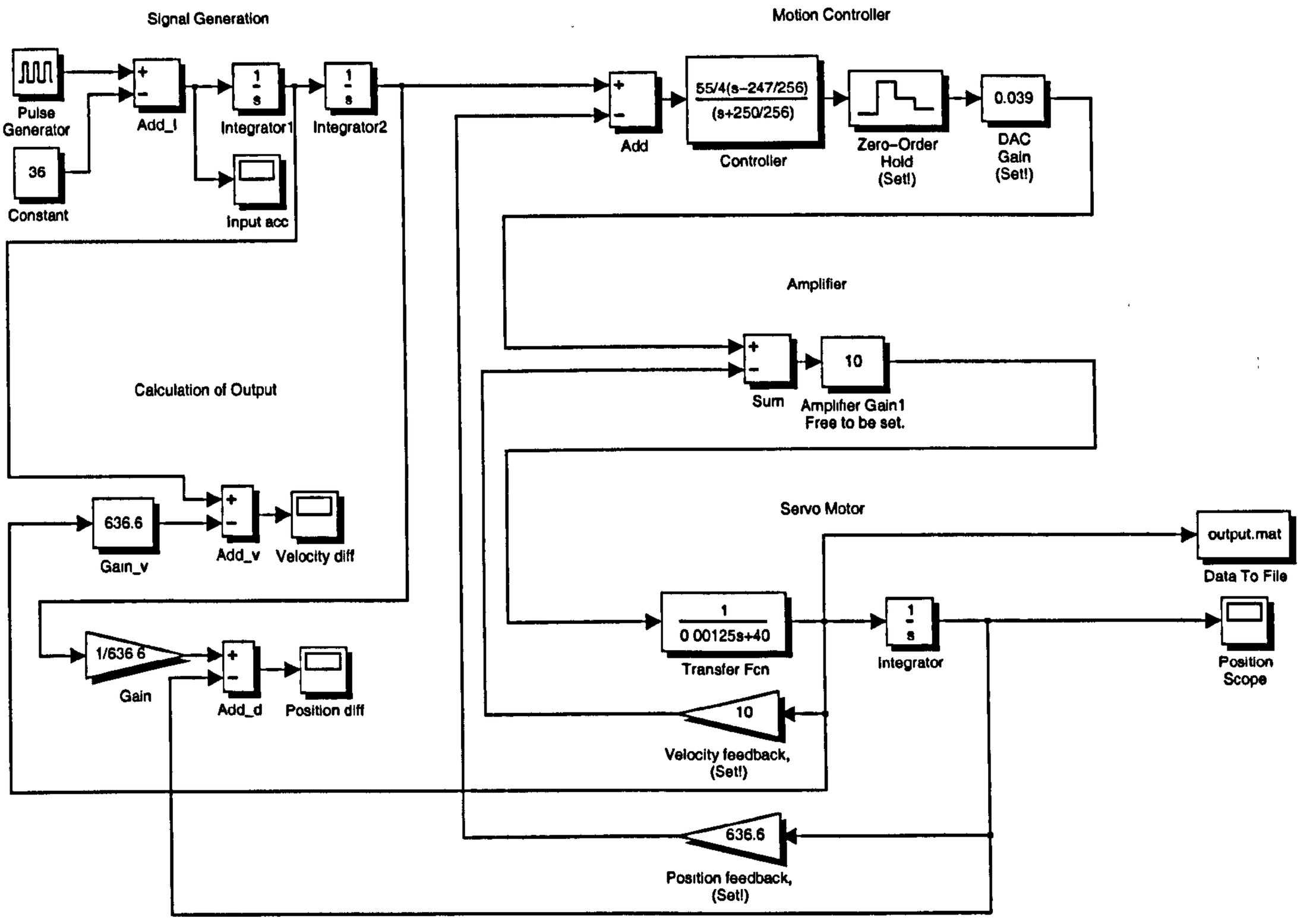


Figure 4.4: SIMULINK representation of the servo-rig control.

Position feedback gain K_p is dictated by the resolution of the encoder employed. The encoder output is proportional to the motor angular position. For the encoder used in the experimental rig, the position feedback gain corresponds to:

$$K_p = \frac{4000}{2\pi} = \frac{\text{encoder counts}}{\text{per revolution}} = 636.6$$

The compensator filter values of filter gain K, zero value A and pole value B have been set on the findings of Osypiw[82]. He observed detrimental effects on the trajectory of the servos outside the limits of $30 < K < 65$. Furthermore the controller has been found sensitive to changes outside the following limits of A, B: $235 < A < 256$, $100 < B < 256$.

The sampling period (t) is set by the sample timer register of the controller, using the following formula:

$$t = 16(T+1)\frac{1}{2 \text{ MHz}}$$

where T is the contents of the controller's time register, which is set by the user, whilst the 2 [MHz] is the frequency of the computer's clock. Experiments with various sampling times revealed that the controller remains stable with sampling times up to 0.41 [msec], which is a register value of 50, beyond this limit the system becomes unstable.

The Max-250 pulse width modulator amplifier with the amplifier gain K_a, which has been used within the experiments must be set manually and cannot be set by software. The K_a gain is adjusted so that the motor rotates at the appropriate speed relative to the input signal.

For the model simulation, the gain of the transfer function of the servo motor is not known. It is set to one and the amplifier gain is used to adjust the system. The mechanical time constant T_m is found experimentally so that the system becomes unstable at sampling times greater than 0.41 [msec].

These values have been used to set up the system of the the motion controller, the amplifier and the servo motor dynamics.

The following data has been generated for the given configuration of the servo motor control:

- Input acceleration, velocity, position.
- Final output velocity, position.
- Velocity difference of input and output velocity.
- Position difference of input and output position.

These findings are illustrated in Figure(4.5).

A trapezoidal acceleration profile is used as the excitation of the system. The corresponding position profile for this movement is also given. Because the servo control system is based on position control, the error for the position is therefore quite small (note the logarithmic time scale). The position error is smaller than 2 % from time 0.05 [s] on. The velocity error is more severe on certain points in time. This behaviour is expected, because the acceleration changes with an infinite slope at these points. The behaviour generated by the simulation with these configurations can be observed and compared with the experimental rig. In particular, that the position can be reached with a maximal error of ± 1 encoder counts of the servo motor. Additionally, there is no oscillation of the servo at the desired

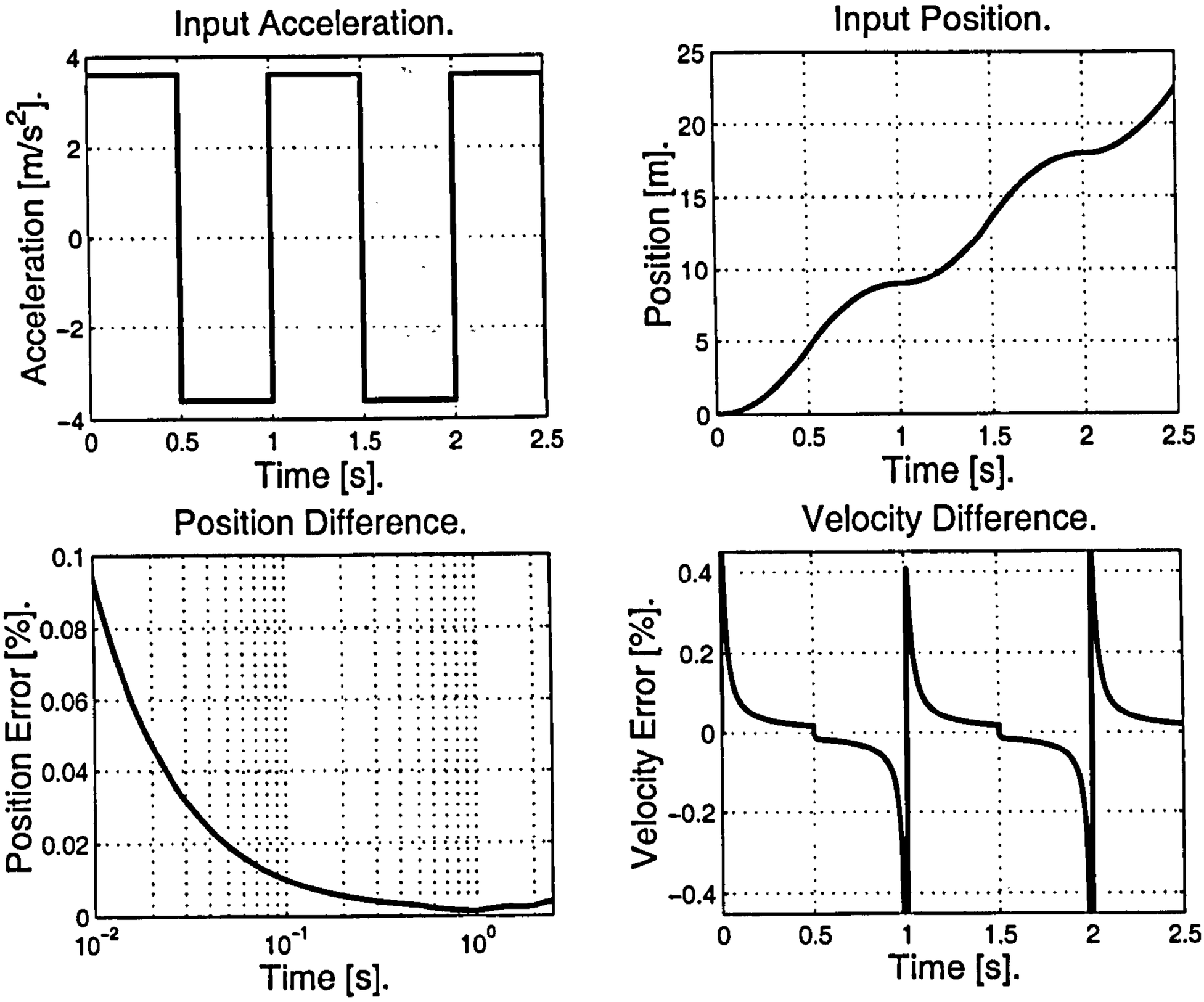


Figure 4.5: The dynamics of the servo control unit.

position.

4.2.2 SIMULINK modelling of the servo-cart system

A second model has also been built to model the servo output system. This includes the mechanical elements of the experimental rig configuration. In particular the spring and damper model of the transmission belt, the interaction of the various masses and the disturbance of the fluid. The properties of this dynamical system are illustrated in Figure(4.2). Figure(4.6) shows the layout of this dynamical system which again has been modelled within SIMULINK.

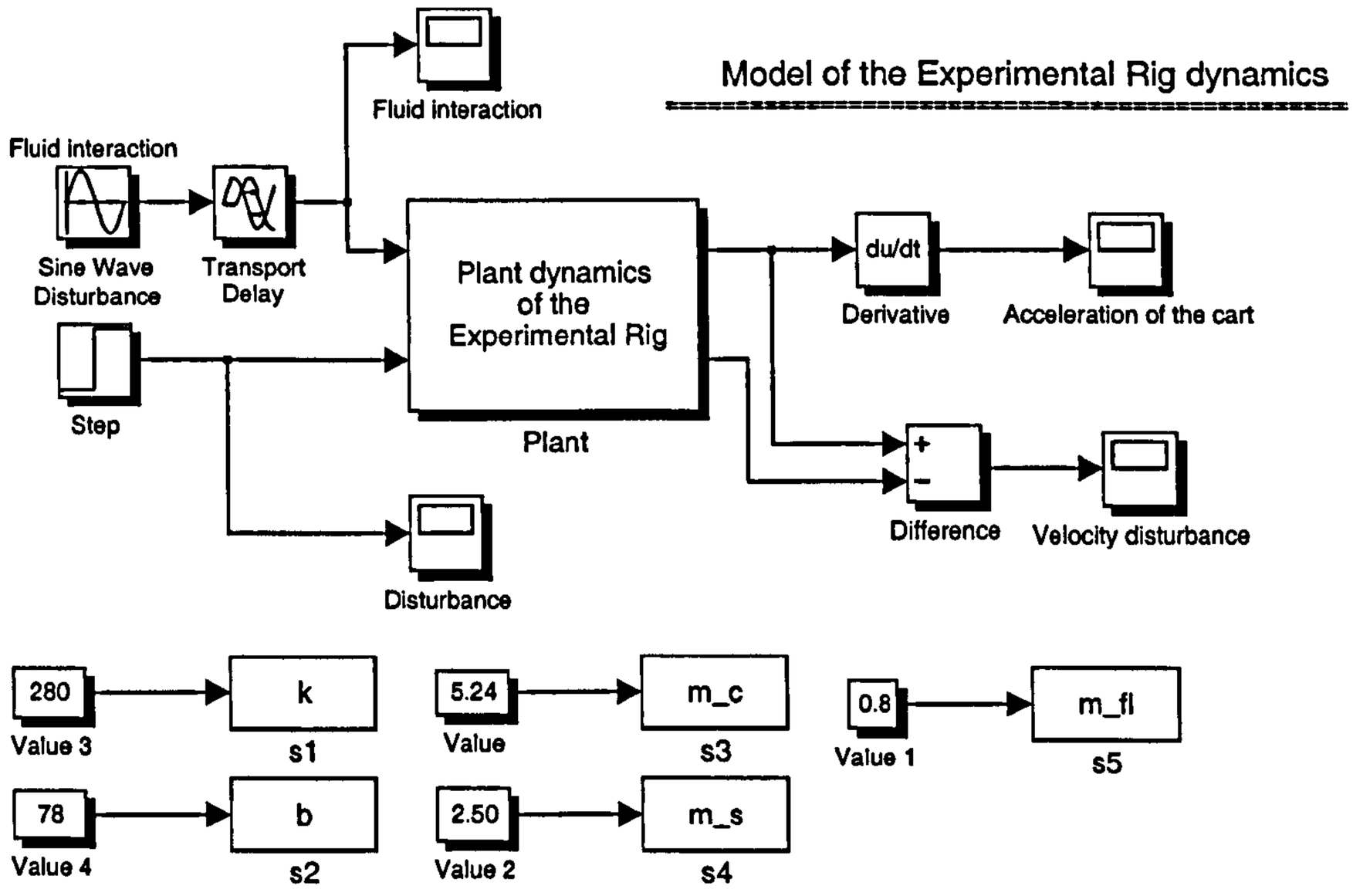


Figure 4.6: SIMULINK representation of the servo-rig control.

The plant dynamics are ordinary differential equations based on the model and equations given in Section (4.1.1). The following parameters must be set:

- The spring coefficient k of the transmission belt.
- The transmission belt damping coefficient b.
- The mass m_s of the servo shaft and joint shafts.
- The mass m_c of the cart.
- The mass m_{fl} of the fluid.

A step input force in the motion controller combined with an oscillating force, generated by the fluid has been used to simulate the behaviour of the transport-cart system. The response of this system can be observed in Figure (4.7). Here the initial input acceleration, the corresponding actual acceleration of the cart and the velocity difference of servo output and cart motion are shown.

To calculate these results the following settings for the parameters of the experimental rig have been used. The container is 80 % filled with water, resulting in a mass of $m_{fl} = 0.8$ [kg]. The shaft of the servo motor and the joint shafts have a mass of $m_s = 0.45$ [kg]. The cart

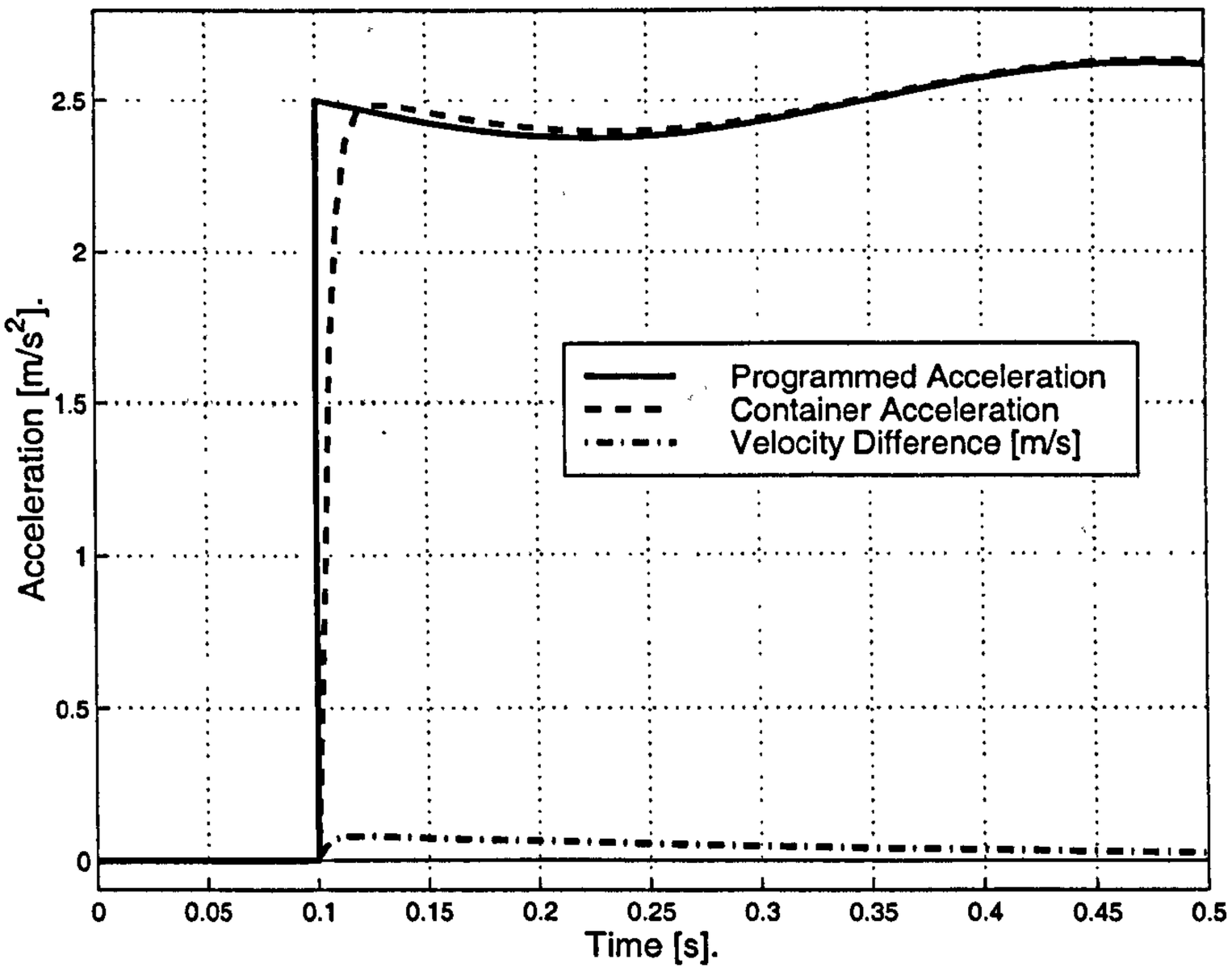


Figure 4.7: Simulation results of the warehouse.

and the container have a mass of $m_c = 2.55\,[\text{kg}]$. The spring coefficient of the transmission belt is approximated with $k = 350\,[\text{N/m}]$ and the damping coefficient with $b = 110\,[\text{kg/s}]$. These values result in a representation of the plant mechanics. These dynamics represent the stiff mechanics of the experimental rig.

4.3 Combination of both models of dynamics

In the case of trajectory optimisation of a fluid filled container the combination of the model of the fluid motion and the mechanical system is the essential part to be performed. This is essential since the dynamics of the mechanical system and the dynamics of the fluid interact and influence each other. Within the combination of the dynamics two different strategies can be distinguished, weakly coupled systems and strongly coupled systems. Both approaches will be described in detail.

Additionally, the simulation of the fluid motion and the mechanical system and their combination has to suit the optimisation procedure which will be used. The general focus is on an optimisation procedure which is only interested in a sloshing behaviour in reaction

to a specific parameter setting and its calculation.

The overall optimisation strategy can be observed in Figure(4.8). An initial set of parameters ($\mathbf{x}^0$) is fed into some modelling and simulation module of the solid and fluid mechanics. Here the response of the mechanical system and the fluid motion due to their characteristic transfer functions and mathematical descriptions is calculated. The simulation module supplies information such as the amplitudes of the surface at the boundaries ($\mathbf{fun}(\mathbf{x}^k)$), the violation of the constraints ($\mathbf{g}(\mathbf{x}^k)$) and their gradients ($\nabla\mathbf{fun}(\mathbf{x}^k)$, $\nabla\mathbf{g}(\mathbf{x}^k)$). Before this information is used within the optimisation module, it can be observed and modified if necessary through the user interface.

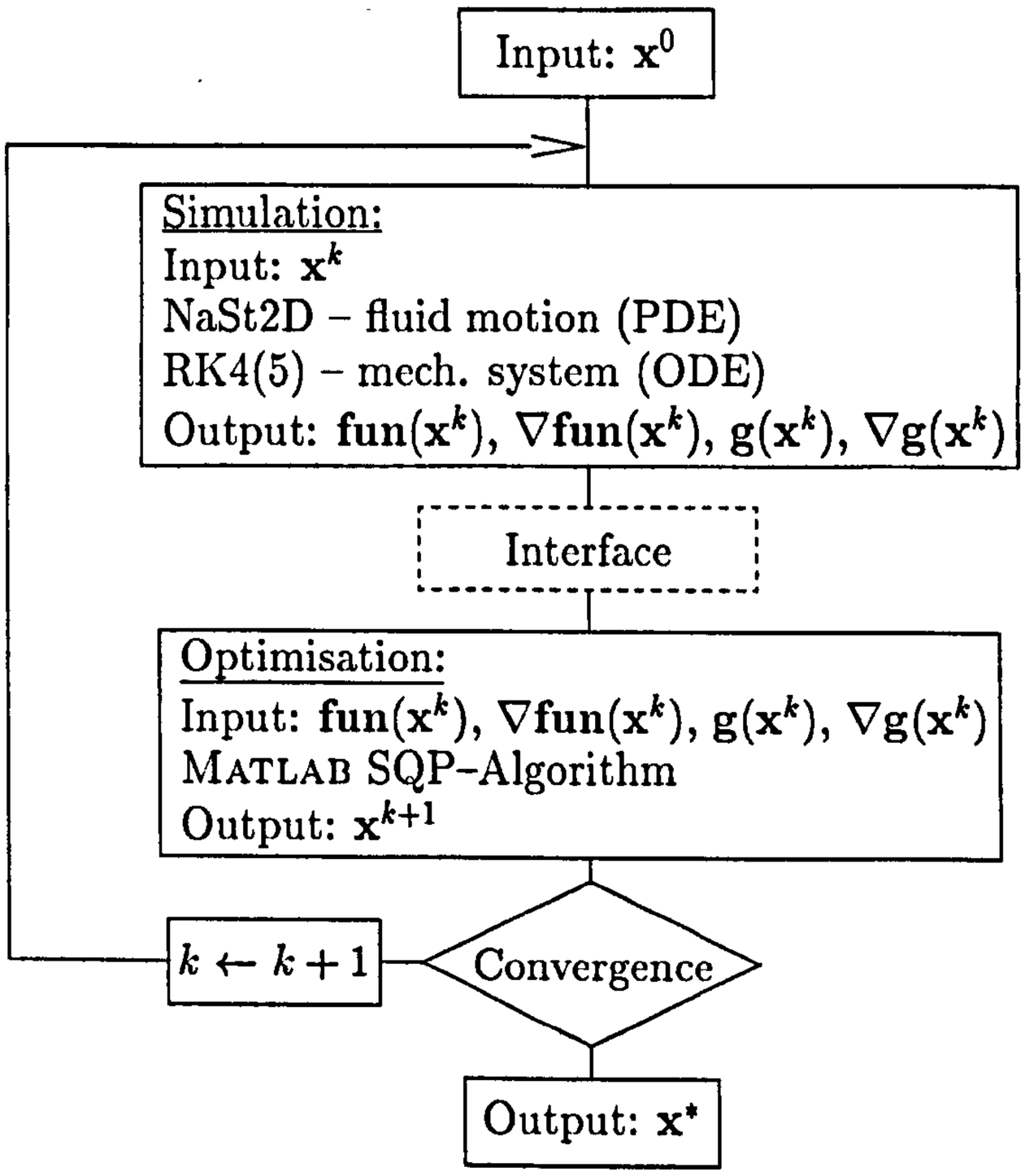


Figure 4.8: Optimisation strategy.

The optimisation module utilises a hill climbing technique to determine the alterations on the parameter values of the acceleration profile and the time. The step-direction and the step size can be calculated due to the gradients of the function and the constraint violations. In comparison an evolutionary algorithm could be used as well. In Figure(4.8), a (0) indicates initial predefined values, (k) indicates a set of parameters within the optimisation

cycle and (*) denotes optimised values.

The control parameters for the mechanical system (e.g. time function of shaft-angle or number of revolutions) are influenced by the acceleration profile for the calculation of the fluid motion and vice versa. There are two main strategies which can be followed to simulate the combination of the fluid motion and the mechanical system. The first simulation strategy involves alternating computation of the fluid motion and the mechanical system with cross interaction of the calculated results. In general the simulation using such a routine will converge with an increasing number of alternating calculations. This can be described as weakly coupled approach.

The second approach solves alternating the fluid motion and the mechanical system within each time step of the calculation of the fluid motion. In general the simulation using such a routine will converge with a decreasing time step size within the calculation of the fluid motion. This latter approach can be classified as being strongly coupled. Each strategy is discussed in detail as follows.

4.3.1 Weakly coupled combination strategy

The calculation of the response of the overall system is illustrated in Figure(4.9). The simulation of the mechanical system and the fluid motion uses a direct shooting method to determine the response of the system due to its stimulus. This can be formulated as an initial value problem (IVP). The steps are as follows:

Step 1. An acceleration distribution, whether supplied initially from the user or by the optimisation module drives the solid mechanical system, where the fluid is modelled as a rigid body. The IVP solver calculates the response of the mechanical system and passes the slightly altered acceleration profile to the next step.

Step 2. The updated profile is used to determine the reaction of the fluid, the pressure distribution and the force impact on the walls of the container due to the dynamics of the fluid flow.

Step 3. With the information of the influence of the fluid on the mechanical system, the model of the mechanical system can be simulated again. The initial acceleration profile changes once more. This time with the fluid flow information gained via using the acceleration profile calculated within the 1^{st} step.

Step 4. The objective function and the most important constraint of not sloshing over, can only be calculated while simulating the fluid motion. Therefore the final simulation of the overall system must involve the fluid dynamics. This calculation uses the new acceleration profile gained after step (3).

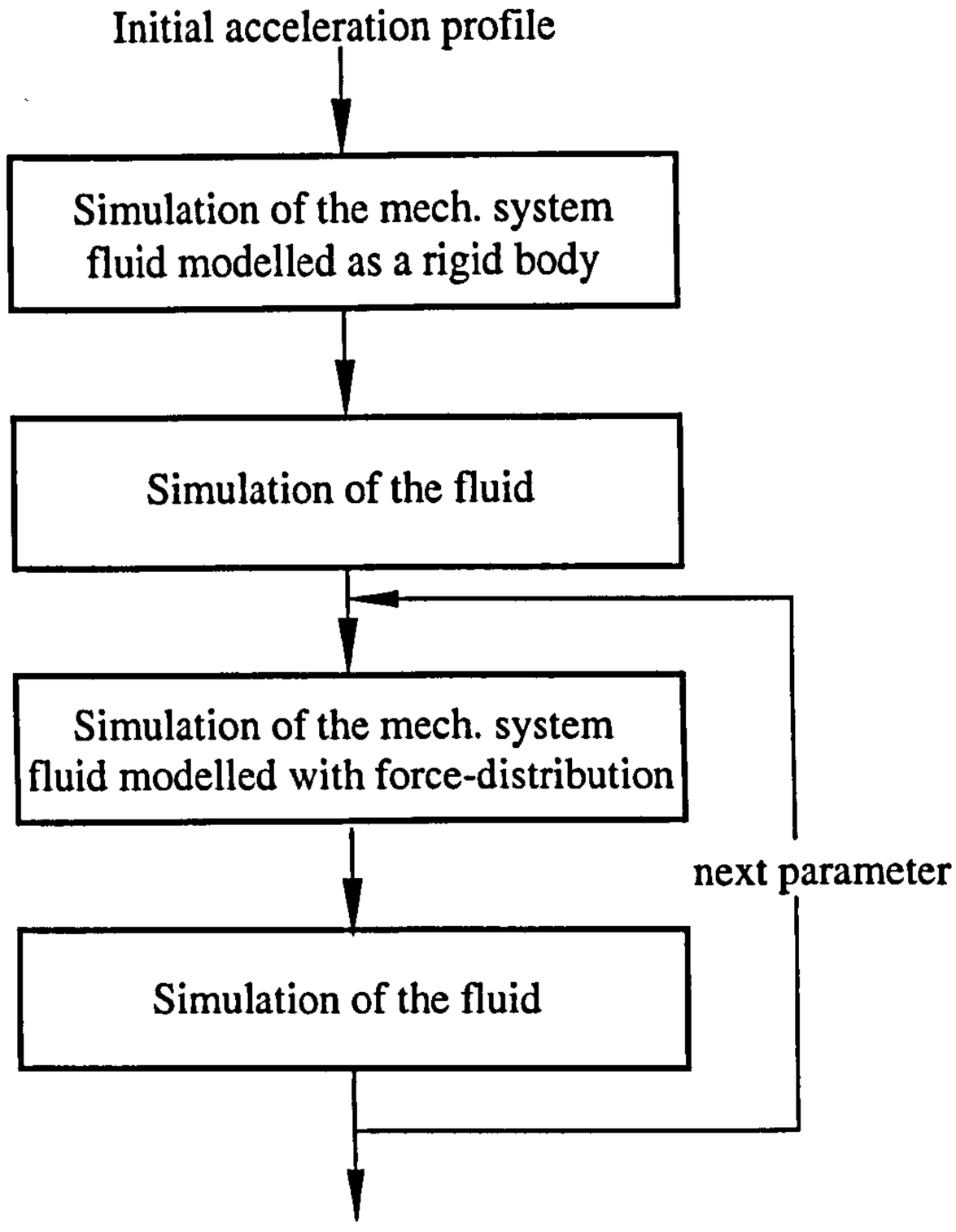


Figure 4.9: Combination of the dynamics.

This integrated strategy of weakly coupled alternating simulation of the fluid motion and the mechanical system results in an iterative process. The accuracy of the resulting simulated overall model is dependent on the number of iterations. The error of the result decreases while increasing the number of iterations.

After four single simulations, the mechanical system has influenced the fluid motion and vice versa. In Section (4.3.3) results are given concerning the accuracy of the simulation module when using two simulations of the mechanical system and two simulations of the fluid motion (illustrated in Figure(4.9)) or only one simulation of the mechanical system and one simulation of the fluid flow starting one step later in Figure(4.9).

The first two simulation steps of Figure(4.9) can be skipped within one iteration of the optimisation algorithm because the alterations within the parameter variation to obtain gradients for each parameter are very small. This is only true for the case that the effects of a slight alteration on the acceleration will not influence the behaviour of the fluid rapidly while simulating the mechanical system.

However, within the optimisation algorithm a fundamental problem is the evaluation of a new set of parameters based on the information obtained from previous calculations. The way this information is provided and their contents are crucial. Therefore it has to be examined whether this weakly coupled approach of alternating simulation gives a sufficiently precise result.

4.3.2 Simultaneous, closed coupled simulation

In this approach the mechanical system is solved within every time-step of the simulation of the fluid motion since the time step within the computation of the fluid motion is very much restricted. The time step within the fluid simulation is calculated depending on the velocity of the particles within the fluid. The particles are not allowed to travel more then the size of a single mesh cell within one time step of the fluid simulation. Hence the time steps in a transient computation are quite small, and the variation within the mechanical system can only be very small. Additionally the properties within the mechanical system can instantly influence the fluid motion. It is for these reasons that this approach is defined as a close coupled approach.

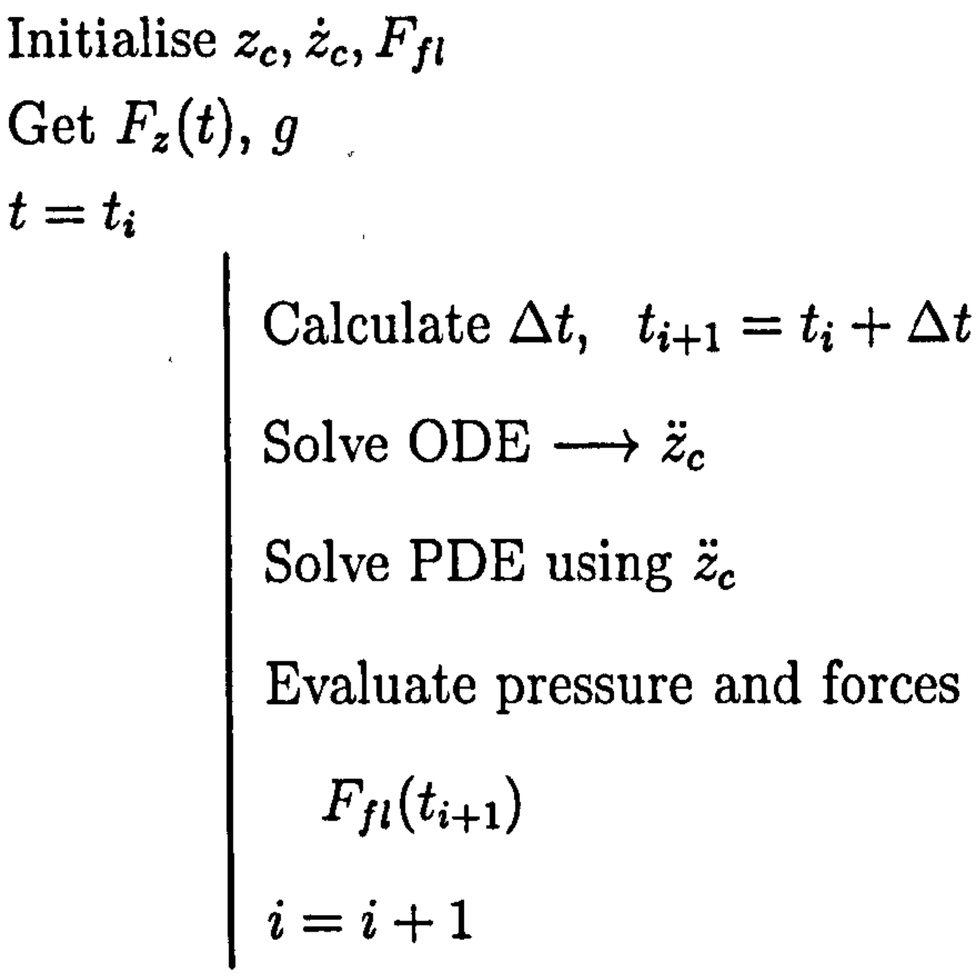


Figure 4.10: Computation sequence.

The computation sequence shown in Figure(4.10) illustrates the implemented simulation of the overall system. Here the calculation of the solution to the ordinary differential equations and the layout of the fluid motion are performed simultaneously. It can be seen that they are influencing each other within each time-step of the simulation of the fluid motion and the values of the various forces are updated from one time-step to the next.

The dynamic model shown in Figure(4.2) illustrates the previously used forces and definitions in one spatial direction. The excitation of the mechanism is represented with the force F_z. The movement of the fluid results in a secondary force F_{fl}. There is additional friction $F_{r,h}$ due to the weight of the mechanism and the fluid, and the fluid dynamics. The movement of the container is represented with z_c, $\dot{z}_c$, $\ddot{z}_c$.

The procedure was programmed in the programming language C (See Appendix(A.1.1.3) for a printout of the code). This procedure was used within the calculation of the mechanical system (ordinary differential equations) as part of the fluid motion solver NaSt2D.

4.3.3 Performance study of model integration schemes

This section focuses on the performance of the two different combination approaches of the calculation of fluid and mechanical system. The two approaches are:

1. Weakly coupled approach which consists of alternate calculation of the two models, hence a sequential exchange of the influence of the properties of the two models.
2. Closed coupled approach of integrated calculation of the two models, thus a parallel exchange of the influence of the properties of the two models.

These two approaches have been compared to distinguish differences and preferences.

It has been intended to use the practical application of the strategies within an optimisation algorithm as a base for the comparison. Therefore not only the accuracy and the computation time of a single component calculation has been compared, but also the use within the iterative process of the optimal control calculations. This has been done due to the possibility of the re-use of information already gained and the possibilities of parallel computing within an optimisation procedure.

Based on these definitions and goals, some properties have been specified to compare the two approaches. All computations are based on the same configuration of the mesh and the mechanical system and they are all performed on the same machine. This is necessary to guarantee comparability of the results achieved. The performance of the strategies is not dependent on these specifications. The closed coupled approach is used as the reference calculation for the weakly coupled calculation due to the greater accuracy of this approach. Its accuracy parameter, the time step of the transient fluid motion, has not been changed within these calculations. The properties summarised in Table (4.3) are as follows:

- Single component calculation.
  One single calculation of the weakly coupled approach ($\boxed{\text{ODE}}\text{–}\boxed{\text{PDE}}$) is compared to one combined closed coupled approach ($\left[\begin{smallmatrix}\boxed{\text{ODE}}\\ \boxed{\text{PDE}}\end{smallmatrix}\right]$) calculation.
- Eight parameters, high accuracy.
  The calculation is based on a sample calculation within the optimisation. In this sample calculation it is said that seven parameters and the overall time of the motion have been specified to interpolate the acceleration of the cart. Hence in each iteration of the optimisation eight values must be varied and their change must be observed. Therefore the single component calculation must be multiplied by eight.
  Additionally in order to get high accuracy, the single component calculation of the weakly coupled approach must be performed twice to enable the fluid to influence the mechanical motion and vice versa. This results in the following calculation procedure

ODE–PDE–ODE–PDE.

- Eight parameters, low accuracy.
  This comparison is similar to the previous one, in terms of the specification of the parameters and the calculation of the closed coupled approach. Within the weakly coupled approach two different calculation procedures are used.

  1. ODE–PDE–ODE–PDE procedure for one acceleration parameter and the overall time of the motion. These calculations are used as an initial guess for the calculation of the other six acceleration parameters.
  2. Single ODE–PDE calculation with the initial guess from the first procedure for the remaining parameters.

  This strategy assumes that the change within the parameter variation is small and therefore the influence of the fluid onto the mechanical system and vice versa is very similar for every varied parameter within one iteration.

- 2/4/8 processors.
  These calculations are based on the use of a parallel processor. The eight calculations within one iteration of the optimisation are distributed to a changing number of processors. Three different types are examined theoretically, they are: the use of two, four or eight processors. The calculation within the weakly coupled approach is based on the strategy described in: Eight parameters, low accuracy.

The values in column two and three are the calculation time of the CPU in [s] of the previously decribed strategies and approaches. Column four illustrates the time correlation of the two main approaches of weakly coupled (ODE–PDE) and closed coupled ($[\frac{\text{ODE}}{\text{PDE}}]$) calculation. The smaller value indicates the faster calculation.

	ODE–PDE	$[\frac{\text{ODE}}{\text{PDE}}]$	Time-rate
Single component calculation	46.00	47.76	0.96:1
Eight parameters, high accuracy	736.00	382.08	1.93:1
Eight parameters, low accuracy	460.00	382.08	1.20:1
2 processors	230.00	191.04	1.20:1
4 processors	138.00	95.52	1.44:1
8 processors	92.00	47.76	1.93:1

Table 4.3: Computation time in CPU [s].

Within Table (4.3) a comparison of weakly and closed coupled calculation of the fluid-structure interaction performed on different calculation platforms is given. The values indicate, that the closed coupled calculation is advantageous in most applications.

Based on the accuracy of the closed coupled approach the author investigated the improvement of accuracy within the weakly coupled approach. For this purpose the single weakly coupled calculation procedure ODE–PDE has been repeated up to four times, using the previously calculated results as an initial guess. The resulting motion has been compared to the motion of the closed coupled approach. A deviation from the results of the closed coupled approach has been calculated and plotted in Figure(4.11).
The weakly coupled calculation procedures can be illustrated as follows:

- 1. Calculation ODE/PDE
  ODE–PDE
- 2. Calculation ODE/PDE
  ODE–PDE–ODE–PDE
- 3. Calculation ODE/PDE
  ODE–PDE–ODE–PDE–ODE–PDE
- 4. Calculation ODE/PDE
  ODE–PDE–ODE–PDE–ODE–PDE–ODE–PDE

In Figure(4.11) a comparison is shown between alternating and integrated simulation. The integrated simulation has been used as a referencing calculation. The deviations of the alternating simulation after one, two, three and four cycles of ODE/PDE calculations compared to the integrated simulation, in percent of the integrated simulation, are shown.

4.4 Selection of the overall modelling strategy

The modelling of the fluid motion and the mechanical system are based upon standard approaches. Since the optimisation is an iterative process, needing possibly an extremely large number of computations the main focus has been on a balance between accuracy

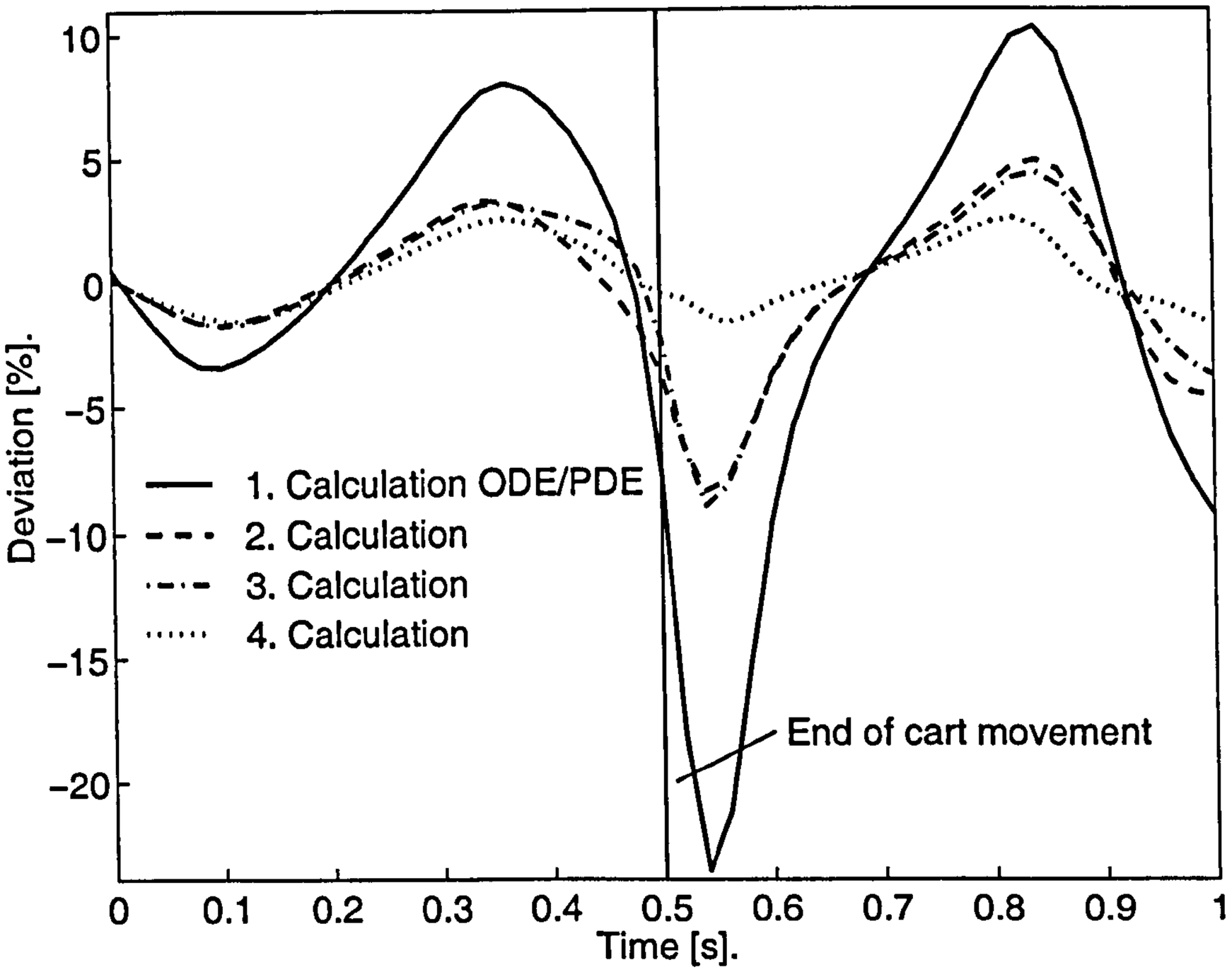


Figure 4.11: Accuracy of alternating compared to integrated calculation.

and simplicity of the models. It is clear that this aim is quite difficult to achieve since free surface modelling is quite demanding in terms of computation time and accuracy.

The data in Table (4.3) and the graphs in Figure(4.11) illustrate quite comprehensively that in each practical computational application the integrated closed coupled simulation is able to perform a more precise result quicker than the alternating weakly coupled simulation. Therefore the author has used this type of closed coupled approach for the optimal control calculations of the minimal time trajectory.

Due to the importance of simplicity and computational speed the author combined the models of fluid flow and mechanical system into one software code. In particular the closed coupled approach is implemented in a way that the transient calculation of one of the models is split into several sub-intervals to be processed within each time step of the other transient model. For reasons of lower data storage, the mechanical model has been split into sub-intervals and integrated within the calculation of the fluid motion.

4.5 Nomenclature

A	digital filter zero
b	coefficient of the damper
B	digital filter pole
k	coefficient of the spring
fun	function vector of the plant
F	general force vector
F_{fl}	fluid force
$F_{r,s}$	friction force of the slider
F_z	external force
g	natural gravity
h	iterative step
k	iteration level
K	gain
m	mass
p, q	order of Runge-Kutta equations
s	domain operator
t	time
T	time constant
$\boldsymbol{x}, \vec{x}(x,y,z)$	position vector with cartesian components x, y, z
x	vector of control or design parameters
z, $\dot{z}$, $\ddot{z}$	position, velocity, acceleration in the horizontal direction of the conveyor system
α_i, $\beta_{i,j}$, γ_i	coefficients of the Runge-Kutta equations
δ	actual error
Δt	time step
μ_s	friction coefficient of the slider
ψ, ϕ	functions, in general
τ	admissible error

Chapter 5

Simulation and Optimisation

This chapter is concerned with the application of the optimisation to the overall model of the fluid-mechanic system investigated in Chapter (4).

The general properties, structure and rudiment to solve the optimisation problem is discussed and illustrated. For this purpose the optimal control problem is transferred to a non-linear programming problem. This involves the introduction of control parameterisation, approximation and solution of the initial value problem.

Additionally the specific optimisation procedure of sequential quadratic programming is illustrated. The essential details are the solution to the quadratic programming sub-problem, the update of the Hesse matrix and the specification of the merit function.

5.1 Simulation of the overall model

Simulation is, first of all, to make experiments with a model; moreover, computer simulation is experimenting with the mathematical model. The reason why computer simulations have become so popular are their advantages for gaining insight into system properties. Such advantages as

- Fast turnaround and low cost investigations,
- No risks or danger as with experiments on the real cart or vehicle.

In particular, simulation models are needed for

- Early stage basic design considerations such as estimation of the approximate dynamic behaviour of alternative concepts based on simple models;
- Design and optimisation of system performance based on more involved but still low order design models;
- Evaluation of the performance of the final design, e.g. running stability, critical velocities, curving ability, force levels;
- Prediction of experimental and field test results to assist in the design of test schedules.

Up to now, the possibilities of mathematically representing the physical properties and interactions of the cart and the fluid have been detailed. This led to the formulation of a model for these systems.

Within this project, simulation is the link between the model and the optimisation. The simulation gathers the information about the plant determined and fixed within the model and the control. This information is processed to calculate specified characteristic values of the plant. This new information can be further processed by the optimisation to generate and calculate optimal values for the control.

5.2 Structure of the optimisation

The general structure of the simulation and optimisation procedure is shown in Figure(5.1) and can be described by the following steps:

Step 1. Get an initial parameter set.

This initial parameter set contains several value pairs of time and amount of acceleration at the specific time. The minimum number of pairs is four and the maximum number should not be greater then twenty, to avoid extensive calculations within the iterative optimisation procedure.

This initial parameter set must be within the boundaries and limitations of the optimisation.

Step 2. Change the parameter set due to the optimisation scheme and the optimality criteria. Due to the actual optimisation scheme one or several different sloshing activities have to be calculated.

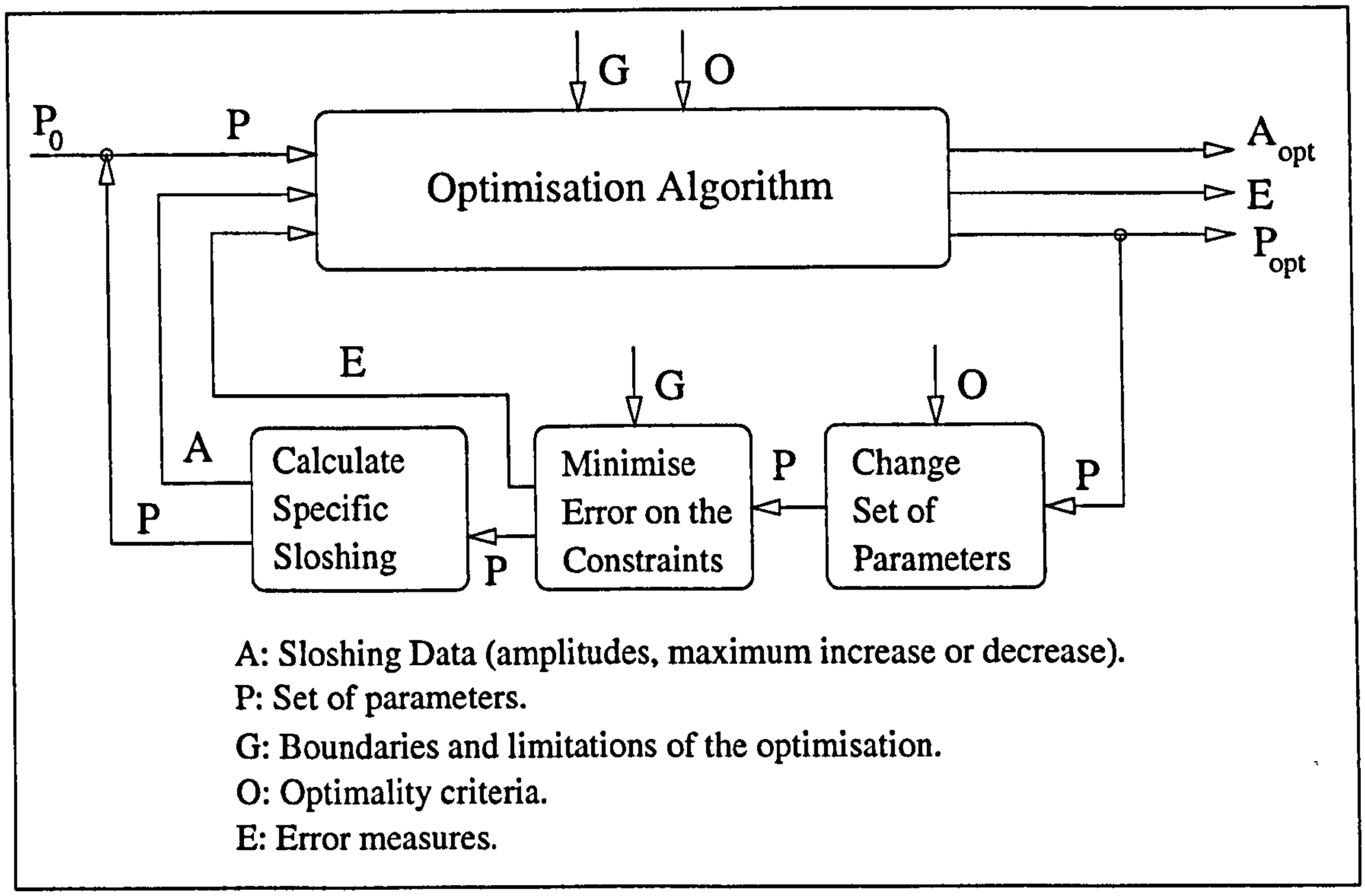


Figure 5.1: General optimisation structure.

Step 3. Optimise the error measures on the constraints. There are limitations on the velocity, the acceleration and the distance of the container and the fluid within the container. The overall time of the transport can be limited as well.
To reach a predefined point within the automized warehouse certain values based on the acceleration profile have to be kept to. The achievement of these values, defines the optimisation of the error measures within this context.
The first limitation defines the resulting distance, based on the change of the velocity in time. The second limitation assures that the container is without any motion after the predefined point has been reached.

Step 4. Calculate the sloshing response due to different sets of parameters.

Step 5. Based on the previous outcome of the sloshing calculations choose next general optimisation direction and start again at Step (2). If a predefined error measure is achieved, terminate the optimisation and return the optimised values of the parameters and the sloshing data, together with the values of different error measures to the user interface.

The alteration of the parameter set and the minimisation of the error measure are part of some optimisation routine. A finite difference code, named NaSt2D[45] has been chosen Leonpacher[68] to calculate and simulate the specific sloshing. The code has been altered by the author to embed the physics of the problem and to create an interface for the exchange of sloshing and parameter data.

5.3 Optimal control problem (OCP)

The optimisation problem needs to be formulated mathematically, allowing computation of cost function and constraints via numerical algorithms. The model for the motion of the cart and the fluid flow, and their interaction has been investigated in the previous chapters.

The formulation of the modelling problem illustrates the separation of two different models. The influences in the specific case of trajectory path planning for open fluid filled containers are separated into the effects within the fluid described by the velocity vector $\boldsymbol{u} = (u, v)$ and those in the mechanical system providing the movement in the coordinates $\boldsymbol{x} = (x, y)$.

The general optimal control problem is formulated as the minimisation of a real-valued objective function subject to the differential constraints[1], appropriate boundary conditions and algebraic state or control constraints by proper choice of control function $\mathbf{u}(t)$ and design parameters, e.g. the final time $b = t_b$.

In the case of optimal control of sloshing liquids, the optimal control problem (OCP) will be formulated as follows:

[1] Ordinary differential equations (ODEs) and partial differential equations (PDEs)

OCP

$$\min_{\mathbf{u}\in U} \int_a^b \psi(t, \boldsymbol{z}(t), \boldsymbol{u}(\boldsymbol{x}, t), \mathbf{u}(t))\mathrm{d}t + \phi(\boldsymbol{z}(a), \boldsymbol{z}(b), \boldsymbol{u}(\boldsymbol{x}, a), \boldsymbol{u}(\boldsymbol{x}, b))$$

subject to

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}(t) - \mathbf{fun}(t, \boldsymbol{z}(t), \mathbf{u}(t)) = \mathbf{0} \qquad t \in [a, b], \tag{5.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p - \frac{1}{Re}\Delta\boldsymbol{u} = \mathbf{F} \qquad t \in [a, b],\ \Omega_i \subset \Omega \tag{5.2}$$

$$\mathrm{div}(\boldsymbol{u}) = \mathbf{0} \qquad t \in [a, b],\ \Omega_i \subset \Omega \tag{5.3}$$

$$\mathbf{g}_{\mathrm{inequality}}(\boldsymbol{z}(t), \mathbf{u}(t), t) \geq \mathbf{0} \qquad t \in [a, b], \tag{5.4}$$

$$\mathbf{g}_{\mathrm{equality}}(\boldsymbol{z}(a), \boldsymbol{z}(b), \boldsymbol{u}(\boldsymbol{x}, a), \boldsymbol{x}(a)) = \mathbf{0} \qquad \boldsymbol{x} \in \Omega,\ \Omega \subset \mathcal{R}^2. \tag{5.5}$$

where equations (5.1-5.3) describe the differential constraints on the system. In particular, (5.1) represents the mechanical system using ordinary differential equations and (5.2, 5.3) represent the Navier–Stokes equations modelling the fluid flow where Re is the Reynolds number.

The algebraic state or control constraints are given in equation (5.4). There are limitations on the control due to the physical limitations of the motors driving the cart:

$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max}, \qquad \forall\, t \in [a, b].$$

The size of the warehouse or the small scale model defines the boundaries for the state constraints. Additionally the limitation for the liquid not to slosh over is specified:

$$\begin{aligned} \boldsymbol{z}_{min} &\leq \boldsymbol{z}(t) \leq \boldsymbol{z}_{max}, \\ y(t)|_{x=0} &\leq h, \quad y(t)|_{x=\mathbf{a}} \leq h, \qquad \forall\, t \in [a, b]. \end{aligned}$$

The layout of the surface at $a = t_a$ and the velocity and location of the cart at (a, b) need to be specified. Figure(1.4) illustrates the free surface within the container at some time-step $(t_i,\ t_i > a)$; the initial distribution of the fluid is as a horizontal surface. This is specified within the boundary conditions:

$$\begin{aligned} y(a) &= y_a, \quad y(b) = y_b, \\ \boldsymbol{u}(\boldsymbol{x}, a) &= \boldsymbol{u}_{\boldsymbol{x},a}, \quad \boldsymbol{x}(a) = \boldsymbol{x}_a, \qquad \forall\, \boldsymbol{x} \in \Omega,\ \Omega \subset \mathcal{R}^2. \end{aligned}$$

In general Ω denotes a feasible domain for the fluid and Ω_i a sub-volume of the fluid filled region in which the Navier–Stokes equations are solved.

Equations (5.1-5.3) describe the continuous optimal control problem (OPC) of the minimal time transport of an open topped fluid filled container. Only for special cases of the governing equations a solution can be found to OCP. The next section focuses on the development of a representation of OCP to solve the problem iteratively.

5.4 Solving (OCP) with nonlinear programming (NLP)

The model for the motion of the cart and the fluid flow, and their interaction has been investigated. If these governing equations had been linear, an algebraic single iteration solution could have been found. However, the model-equations for the mechanical system and the fluid flow are highly non-linear, hence an numerical iterative algorithms must be used to calculate an optimal solution. This section investigates and develops a strategy to represent the OCP in an iteratively solvable problem formulation. The optimal control problem is generally a continuous problem and must be transformed into an iterative procedure to be solved.

In Chapter (2) some optimisation strategies have been discussed to find an optimal solution for the given problem. Two major differences between the optimisation schemes can be found.

1. Optimisation schemes, which do not use information of the curvature of the surface of parameters to be optimised (e.g. genetic algorithms)

2. Optimisation schemes, which do use information of the curvature of the surface of parameters to be optimised (e.g. hill climbing techniques)

In general random initial parameter settings are used within an algorithm of the first category. Within the feasible domain of the parameters several settings will be used to calculate an intermediate solution. This very large amount of intermediate solutions is used to extract improved parameter settings to be worked on within the next iteration. This system provides solutions from within a huge solution domain, hence such algorithms do usually not get caught within local optimal solutions. However the calculation of a global optimal solution is very cost intensive. The experience of the author leads to the following conclusion: In general it is more likely that an algorithm of the first category reaches finally global optimised values due to their search characteristics, compared to an algorithm of the second category.

The main difference between an algorithm which does not use information of the curvature of the surface of parameters to be optimised and one which does use information of the curvature of the surface of parameters to be optimised shall be illustrated using Rosenbrock's[96] banana function.

$$\phi(\boldsymbol{x}) = 100(y - x^2)^2 + (1 - x)^2$$

Using a genetic algorithm by Goldberg[42] it was shown that the known minimal value $\phi(\boldsymbol{x}) = 0$ at $(1, 1)$ can only be reached with a very high error tolerance of order $(\mathcal{O}^{-1})$ when using the same amount of function evaluations when compared with an algorithm of category two, which uses curvature information.

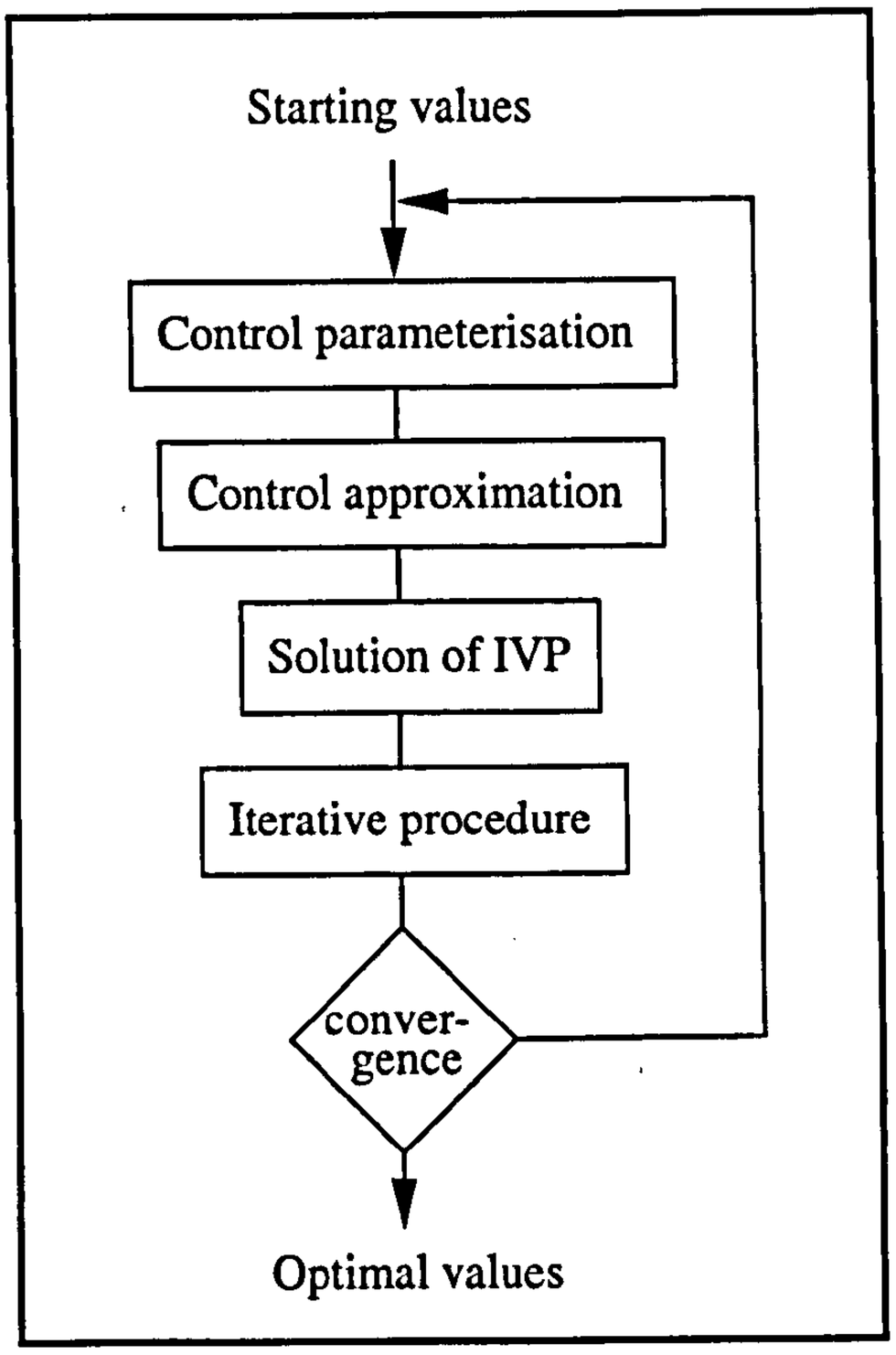
The comparison calculation was performed in MATLAB using the following command:

```
x=fmins('100*(x(2)-x(1)^2)^2+(1-x(1))^2',[-1.2 1],[1,1.e-8]);
```

which required 225 function evaluations. The result has an error tolerance of order $(\mathcal{O}^{-8})$. The search method *fmins* is a direct search method that does not use numerical or analytical gradients.

Therefore the first group of mainly evolutionary algorithms must be neglected due to their extensive use of function evaluations. In the illustrated problem of sloshing water these function evaluations are very cost intensive in terms of calculation time.

Due to the complexity of the equations a direct shooting method will be used which is more robust in comparison to collocation or multiple shooting methods. Robustness is an essential requirement for the illustrated aim of minimal time movement of a cube with sloshing fluid, because of the highly non-linear structure of the Navier–Stokes equations. In addition, due to the complexity of the problem, direct shooting is easier to apply within MATLAB. The structure and implementation of the direct shooting method is shown in Figure (5.2).



The optimal control problem is solved iteratively including the following steps:

1. control parameterisation
2. control approximation
3. solution of initial value problem (IVP)
4. iterative procedure

This will result in a replacement of the optimal control problem (OCP) by a non-linear programming problem (NLP).

Figure 5.2: Applied optimisation structure.

5.4.1 Control parameterisation

The infinite optimal control problem must be converted to a finite dimensional problem by introducing a finite set of control parameters ($\mathbf{x}$) representing the infinite control function $\mathbf{u}(t)$, to be able to solve the OCP.

The resulting non-linear programming problem NLP will be solved using commercial numerical software for finite dimensional optimisation like sequential quadratic programming (SQP) (also known as iterative quadratic programming, recursive quadratic programming and constrained variable metric methods). The following equation describes the NLP formulation:

NLP

$$\min_{\mathbf{x}\in\mathcal{R}^n} \quad \phi(\mathbf{x})$$

$$\text{subject to} \quad \mathbf{lb} \leq \begin{pmatrix} \mathbf{x} \\ \mathbf{g(x)} \\ \mathbf{A}\cdot\mathbf{x} \end{pmatrix} \leq \mathbf{ub}$$

The continuous control function $\mathbf{u}(t)$ is approximated by a vector of control parameters $\mathbf{x}$. The differential constraints represented by $\mathbf{A}$ and bounded with $\mathbf{lb}$ and $\mathbf{ub}$ are solved with initial values $\boldsymbol{\zeta}(a), \boldsymbol{\nu}(\mathbf{x}, a)$, where $\boldsymbol{\zeta}(t)$, $\boldsymbol{\nu}(\mathbf{x}, t)$ are the approximation to $\boldsymbol{z}(t)$, $\boldsymbol{u}(\boldsymbol{x}, t)$ respectively. Additionally the state and control constraints $\mathbf{g}(\mathbf{x})$ must be fulfilled.

5.4.2 Control approximation

The control function $\mathbf{u}(t)$ will be defined differently for each iteration k and has to be approximated via $\tilde{\mathbf{u}}^k(t)$. The different steps are illustrated in Figure(5.3).

1. Interval partitioning: $t_i \in [a, b]$.
   Split the time interval $[a, b]$ in several sub-intervals, not necessarily equally spaced (dashed lines).

2. Choice of basis functions ppf.
   Choose an interpolation scheme e.g. piece-wise polynomial functions (ppf), or other linear or spline functions. In the given example four points are necessary to build a third order polynomial function (dotted line).

3. $\tilde{\mathbf{u}}^k(t) \cong \mathbf{u}^k(t) : \{\text{ppf} \,|\, x_i^k = u^k(t_i)\}$
   Discretise the control using the points x_i^k which represent the value of $\mathbf{u}^k$ at time t_i and interpolate with the basis functions in the intervals in between.

The result is a continuous function $\tilde{\mathbf{u}}^k(t)$ as an approximation to $\mathbf{u}^k$.

5.4.3 Initial value problem

The centre of the direct shooting method is the solution of the initial value problem. Boundary value problems can be formulated as an iterative optimisation, using and solving initial value problems. In this case the system of equations to be solved consists of a sub-system of first order differential equations, representing the mechanical system and the

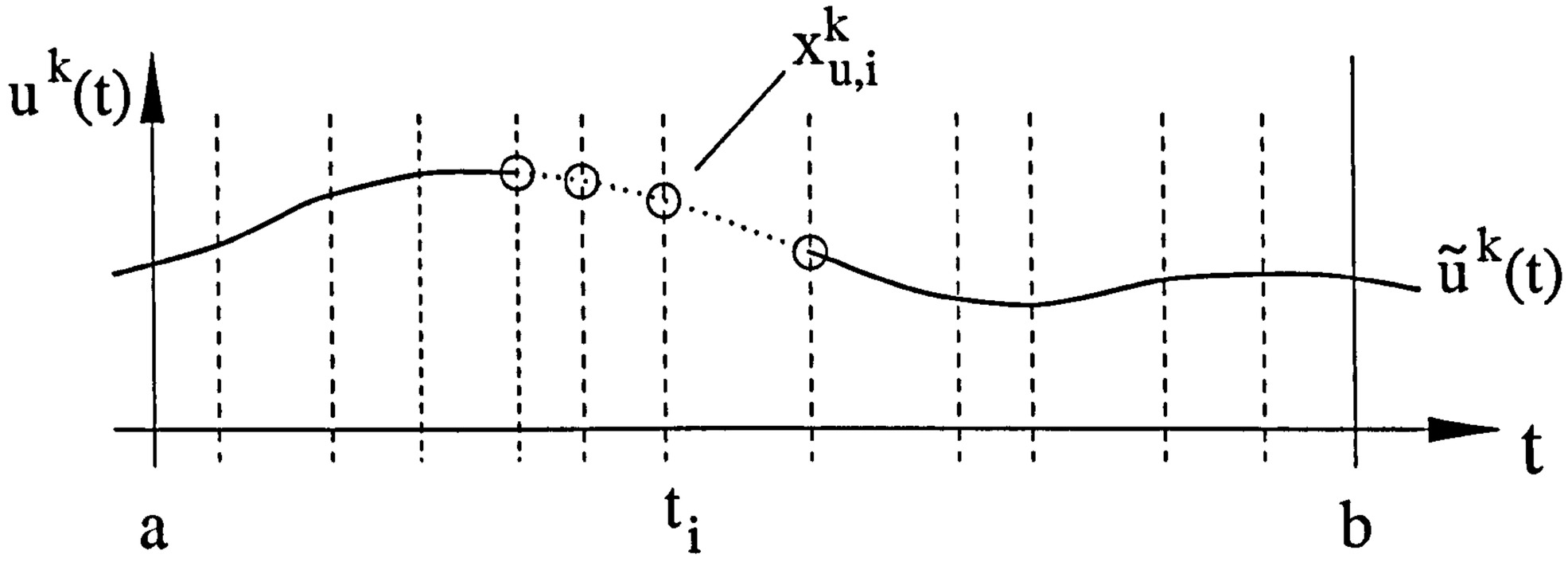


Figure 5.3: Approximation of $\mathbf{u}^k(t)$.

Navier–Stokes equations as representation for the fluid motion. The procedure of solving an initial value problem can be observed in Figure(5.4) and will be performed as follows:

1. Start the calculation with an initial value $\zeta_0 = z_0$.
   This value has been predefined within the boundary conditions.

2. Evaluation of the solution of the IVP using the control function $\tilde{\mathbf{u}}^k(t)$,

$$\zeta_b^k = f(\zeta_0, \tilde{\mathbf{u}}^k(t)),$$

   where k is the number of iteration.

3. The solution deviation is generally $d^k = \zeta_b^* - \zeta_b^k \neq 0$.
   Depending on the value of the state $\zeta_b^k = \zeta^k(b)$ a solution deviation d^k can be calculated which in the optimised case will be $d^* = 0$.

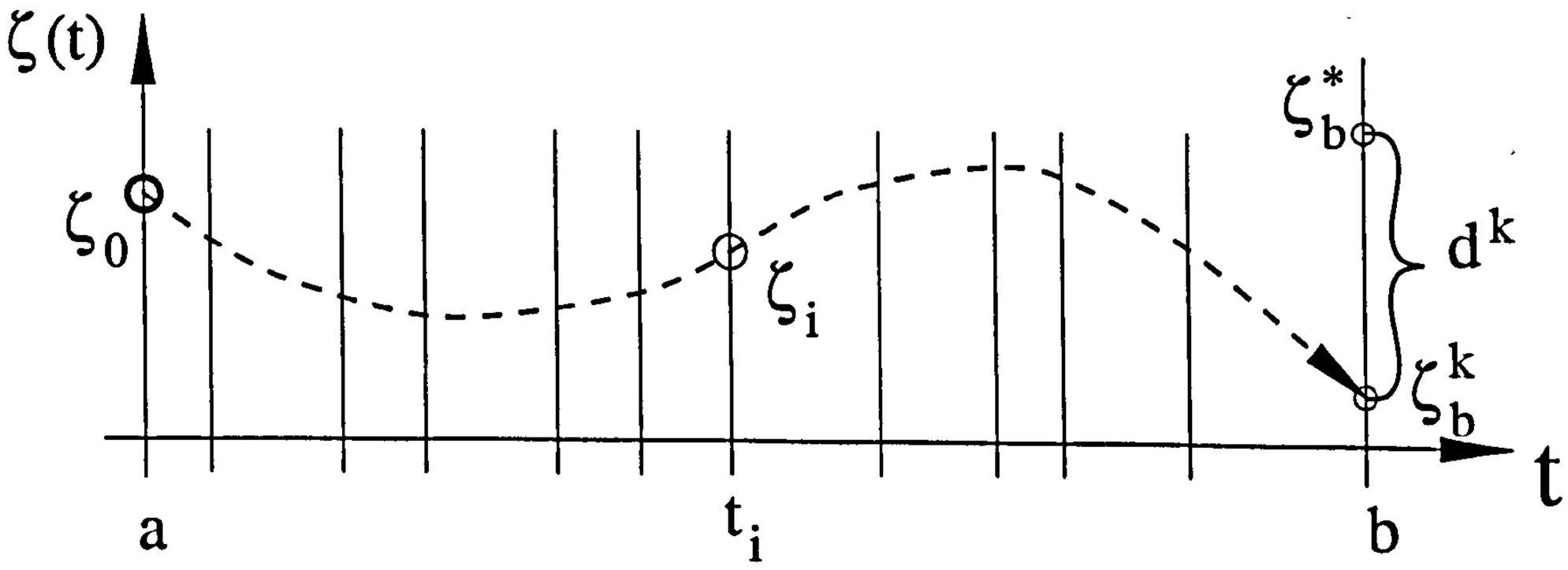


Figure 5.4: Initial value problem.

The constraints must be applied on the control function and the state function. It is very easy to limit the control during its evaluation. Constraint violation of the state will occur while proceeding towards an optimal solution within the iterative process. Point-wise verification as illustrated in Figure(5.5) will be used to measure the constraint violation as follows:

1. Choose $\zeta_{min}(t), \zeta_{max}(t)$.
   The upper and lower bounds on the state are indicated with a grey shade. Generally they are time dependent. In Figure(5.5) these bounds are constant.

2. Set communication grid: $t_i \in [a, b],\ i = 0 \ldots n$.
   The communication grid is illustrated with dotted lines. This grid can be different to the one used for the control discretisation.

3. Analyse function $e_{ub,i} \overset{!}{\geq} 0, \quad \forall\, i = 0 \ldots n$.
   The function will be analysed at all t_i. Figure(5.5) illustrates only the verification on the upper bound. The value of e_i is calculated by $e_{ub,i} = \zeta_{max}(t_i) - \zeta(t_i)$.

A similar procedure can be formulated for the violation of the lower bound:

$$e_{lb}(t_i) \overset{!}{\geq} 0, \qquad e_{lb}(t_i) = \zeta(t_i) - \zeta_{min}(t_i), \qquad t_i \in [a, b],\ i = 0 \ldots n.$$

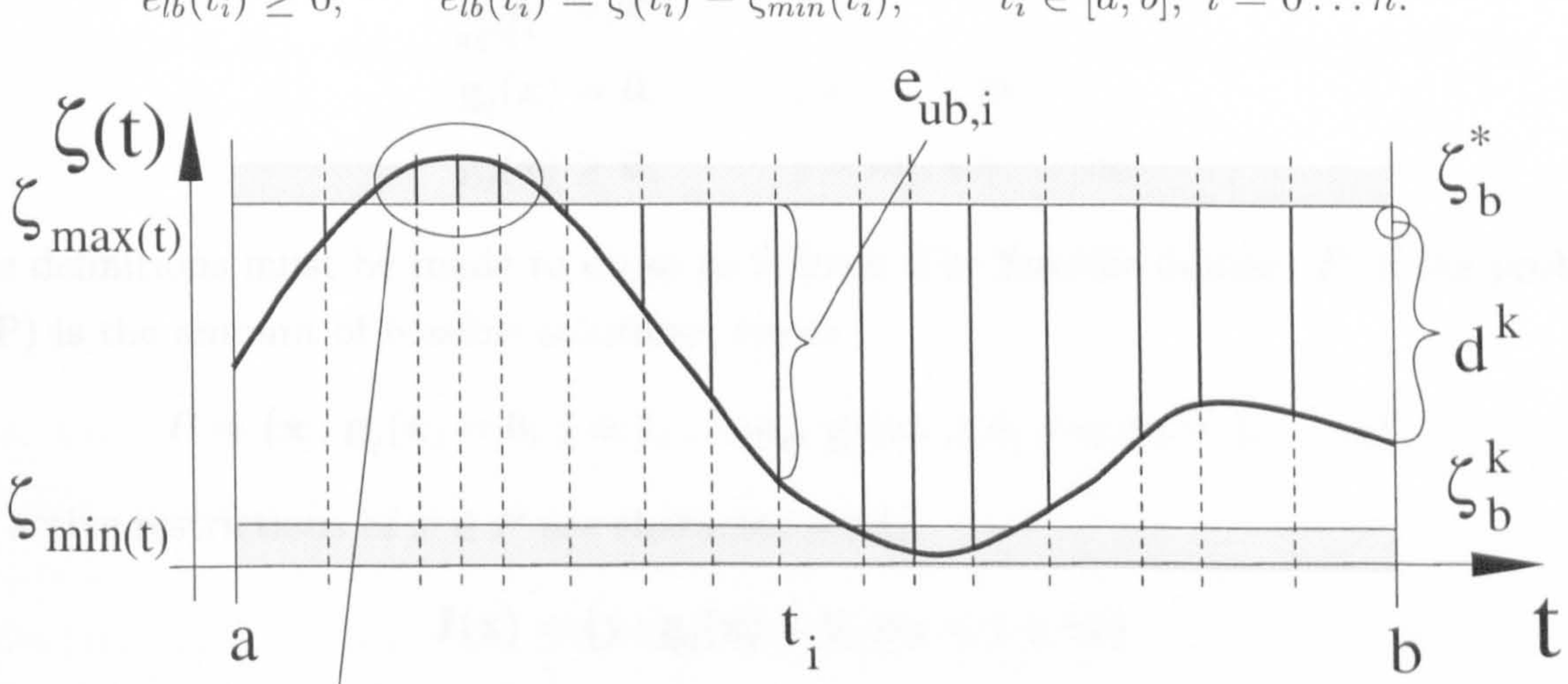


Figure 5.5: Pointwise verification of ζ.

By using the direct shooting procedure it is possible to calculate unique solutions based on the model system. A specific set of control parameters creates a specific solution deviation, constraint violations and a value for the objective function. The control parameters for the

calculation of the fluid motion (proportional to F) are a function of the control parameters x for the mechanical system. Therefore it is not necessary to introduce a separate set of parameters.

5.5 Solving (NLP) with MATLAB sequential quadratic programming (SQP)

In the following section, the nonlinear programming problem (see Section (5.4)) will be solved using sequential quadratic programming. In particular using the mathematical programming language MATLAB. Initially an introduction to SQP algorithms will be given to lead to the special modules implemented in MATLAB.

5.5.1 Introduction to sequential quadratic programming (SQP)

The aim is to solve a NLP problem of the form

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{R}^n} \quad & \phi(\mathbf{x}) \\ \mathrm{g}_j(\mathbf{x}) &= 0, \qquad j = 1, \ldots, m_e \\ \mathrm{g}_j(\mathbf{x}) &\geq 0, \qquad j = m_e + 1, \ldots, m. \end{aligned}$$

Some definitions must be made to do so as follows: The feasible domain P of the problem (NLP) is the amount of feasible solutions, hence

$$P = \{\mathbf{x} : \mathrm{g}_j(\mathbf{x}) = 0,\ j = 1, \ldots, m_e,\ \mathrm{g}_j(\mathbf{x}) \geq 0,\ j = m_e + 1, \ldots, m\}$$

The active restrictions of $x \in P$ are characterised by

$$I(\mathbf{x}) = \{j : \mathrm{g}_j(\mathbf{x}) = 0,\ m_e < j < m\}$$

The Lagrange function of (NLP) is defined for every $\mathbf{x} \in \mathcal{R}^n$ and $\mathbf{l} = (l_1, \ldots, l_m)^T \in \mathcal{R}^m$ with

$$\mathrm{L}(\mathbf{x}, \mathbf{l}) = \phi(\mathbf{x}) - \sum_{j=1}^{m} l_j \mathrm{g}_j(\mathbf{x}).$$

The variables l_j are the multiplier of the Lagrange function.

The following conditions must hold to solve the NLP problem:

- The problem is smooth, hence the problem functions ϕ and $\mathbf{g}$ are continuously differentiable on the domain $\mathcal{R}^n$.
- The problem is small, hence the number of variables and constraints is not too big (e.g. $n, m < 100 - 200$). Otherwise special algorithms like SNOPT[7] would need to be used.
- The problem is well scaled, hence changes of the variables result in changes of the problem functions in the same magnitude.
- The problem is well defined, hence the feasible domain is not empty and an optimal solution does exist.

The previously defined conditions can be verified for the problem of sloshing fluid motion. For example,

- the ordinary and partial differential equations are not analytically but numerically continuously differentiable.
- the problem is small, there is a maximum of 10 variables and less than 15 constraints.
- it has been observed (in a sensitivity analysis) that the problem is well scaled, hence changes of the variables do result in changes of the problem functions in the same magnitude.
- in early optimisation runs it could be observed that the feasible domain is not empty. See Equations (5.6) for the necessary conditions for the existence of an optimal solution.

To picture the Kuhn–Tucker conditions necessary for the existence of an optimal solution the following must be stated.
The problem functions ϕ and $\mathbf{g}_j \; \forall \, j = 1, \ldots, m$ are twice continuously differentiable, $\mathbf{x}^*$ is a local minimum of (NLP) and the constraint qualification[2] holds in $\mathbf{x}^*$. Then there is an

[2]Constraint qualification: linear independence of the vectors $\nabla \mathbf{g}_j(\mathbf{x}^*)$ for all $j \in \{1, \ldots, m_e\} \cup I(\mathbf{x}^*)$.

optimal control $\mathbf{x}^* \in \mathcal{R}^m$, with the following fulfilled conditions:

$$\begin{aligned} l_j^* &\geq 0, \; j = m_e + 1, \ldots, m \\ \mathbf{g}_j(\mathbf{x}^*) &= 0, \; j = 1, \ldots, m_e \\ \mathbf{g}_j(\mathbf{x}^*) &\geq 0, \; j = m_e + 1, \ldots, m \\ \nabla_x \mathrm{L}(\mathbf{x}^*, \mathbf{l}^*) &= 0 \\ l_j^* \mathbf{g}_j(\mathbf{x}^*) &= 0, \; j = m_e + 1, \ldots, m \end{aligned} \tag{5.6}$$

This means that in a minimum the gradient of the objective function is a positive linear combination of the gradients of the active constraints.

Having satisfied the previous conditions a mathematical optimisation algorithm can be applied. Due to their efficiency and robustness, particularly for general problems, where no specialised codes can be developed, a sequential quadratic programming (SQP) algorithm will be illustrated.

SQP algorithms are based on the continuous solution of quadratic sub problems. To solve quadratic problems with objective functions of the form

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{d}^T \mathbf{x}$$

special algorithms can be used which utilise the special structure of such problems to increase convergence speed. The quadratic sub problems within an SQP algorithm are generated by a quadratic approximation of the Lagrange function and a linearisation of the restrictions.

$$\begin{aligned} \min_{\mathbf{s} \in \mathcal{R}^n} \quad & \tfrac{1}{2}\mathbf{s}^T \mathbf{B}_k \mathbf{s} + \nabla \phi(\tilde{\mathbf{x}}_k)^T \mathbf{s} \\ & \nabla \mathbf{g}_j(\tilde{\mathbf{x}}_k)^T \mathbf{s} + \mathbf{g}_j(\tilde{\mathbf{x}}_k) = 0, \qquad j = 1, \ldots, m_e \\ & \nabla \mathbf{g}_j(\tilde{\mathbf{x}}_k)^T \mathbf{s} + \mathbf{g}_j(\tilde{\mathbf{x}}_k) \geq 0, \qquad j = m_e + 1, \ldots, m. \end{aligned}$$

This sub problem, where $\tilde{\mathbf{x}}_k \in \mathcal{R}$ is an approximation to the optimal solution and $\mathbf{B}_k$ is an approximation to the Hesse matrix of the Lagrange function in the optimal solution, must be solved in every iteration of the SQP algorithm. The quadratic sub problem is called (QP). If $\mathbf{s}_k$ is the search direction, $\tilde{\mathbf{l}}_k \in \mathcal{R}^m$ are the approximation to the multiplier of the Lagrange function and $\mathbf{l}_k$ the corresponding multiplier of (QP) the values of the next iteration are:

$$\begin{pmatrix} \tilde{\mathbf{x}}_{k+1} \\ \tilde{\mathbf{l}}_{k+1} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{x}}_k \\ \tilde{\mathbf{l}}_k \end{pmatrix} + \alpha_k \begin{pmatrix} \mathbf{s}_k \\ \mathbf{l}_k - \tilde{\mathbf{l}}_k \end{pmatrix},$$

where α_k is a suitable step length. For $\alpha_k = 1$ it follows that $\tilde{\mathbf{l}}_{k+1} = \mathbf{l}_k$.
A general SQP method can be formulated with the following algorithm.

00) Initially the following values must be set. The generally used start values are given.

$$\begin{aligned} \mathbf{x}_0 &\in \mathcal{R}^n \quad \text{(Given by the user)} \\ \mathbf{l}_0 &\in \mathcal{R}^m \quad (= (0, \dots, 0)^T) \\ \mathbf{B}_0 &\in \mathcal{R}^{n \times n} \quad (= \mathrm{I}, \text{ identity matrix}) \end{aligned}$$

Additionally some values like error tolerance or a reduction multiplier for the step size calculation must be set.

0) Calculate $\phi(\mathbf{x}_0)$, $\nabla\phi(\mathbf{x}_0)$, $\mathrm{g}_j(\mathbf{x}_0)$, $\nabla\mathrm{g}_j(\mathbf{x}_0)$, $j = 1, \dots, m$.

1) Solve the quadratic subproblem (QP).

2) Compute the step length. This calculation is in general based on a predefined merit function with penalty parameters for constraint violations.

3) Set:

$$\begin{aligned} \tilde{\mathbf{x}}_{k+1} &= \tilde{\mathbf{x}}_k + \alpha_k \mathbf{s}_k \\ \tilde{\mathbf{l}}_{k+1} &= \tilde{\mathbf{l}}_k + \alpha_k (\mathbf{l}_k - \tilde{\mathbf{l}}_k) \end{aligned}$$

and calculate $\phi(\tilde{\mathbf{x}}_{k+1})$, $\mathrm{g}_j(\tilde{\mathbf{x}}_{k+1})$, $j = 1, \dots, m$, $\nabla\phi(\tilde{\mathbf{x}}_{k+1})$, $\nabla\mathrm{g}_j(\tilde{\mathbf{x}}_{k+1})$, $j = 1, \dots, m$.

4) Specify the approximation to the Hesse matrix of the Lagrange function $\mathbf{B}_k$ with an appropriate method. This is possible with a Newton algorithm[3] or one of the Quasi-Newton algorithms. The most often used algorithms are those of Davidon-Fletcher-Powell (DFP update) and Broyden-Fletcher-Goldfarb-Shanno (BFGS update).

5) Calculate and evaluate the objective function and the constraint violations, either to stop the algorithm if a local optimal solution has been found, or restart from point one until a solution is found within the predefined error tolerance.

This algorithm to be used is found in the MATLAB OPTIMIZATION TOOLBOX. The special functions for constraint optimal control calculations will be described next.

[3]With the drawback of huge calculation time and possible difficulties in convergence and the layout of the Hesse matrix (nonsingularities and bad conditioning).

5.5.2 Realisation of (SQP) within MATLAB

This section describes the SQP algorithm used in the MATLAB OPTIMIZATION TOOLBOX to solve the constraint NLP problem. A characteristic of a large class of early methods is the translation of the constraint problem to a basic unconstrained problem by using a penalty function for constraints, which are near or beyond the constraint boundary. These methods are now considered relatively inefficient and have been replaced by methods that have focused on the solution of the Kuhn-Tucker (KT) (see Equations (5.6)) equations. If the problem is a so-called convex programming problem, that is, $\phi(\mathbf{x})$ and $\mathbf{g}_j(\mathbf{x})$, $j = 1, \ldots, m$, are convex functions, then the KT equations are both necessary and sufficient for a global solution point. Otherwise the solution might only be a local optimal solution.

The solution of the KT equations forms the basis to many non-linear programming algorithms. These algorithms attempt to compute directly the Lagrange multipliers. Constrained quasi-Newton methods guarantee super-linear[4] convergence by accumulating second order information regarding the KT equations. These methods are commonly referred to as SQP.

A nonlinearly constrained problem can often be solved in fewer iterations than an unconstrained problem using SQP. One of the reasons for this is that, because of limits on the feasible area, the optimiser can make well informed decisions regarding directions of search and step length.

The function available for constraint non-linear minimisation within the MATLAB OPTIMIZATION TOOLBOX is *'constr'*. The general notation for the problems to be solved with this function is

$$\min_{\mathbf{x}} \phi(\mathbf{x}) \text{ such that } \mathbf{g}(\mathbf{x}) \leq 0.$$

The syntax of *constr.m* is

```
[x, OPTIONS,lambda, HESS]
    = constr('FUN',x0,OPTIONS,VLB,VUB,GRADFUN,P1,...,P15);
```

with `x0` as starting point on the problem function `FUN`, applying lower and upper bounds `VLB, VUB`. If mathematical gradients are possible they can be integrated in the function `GRADFUN`. `P1,...,P15` are parameters to be directly transferred to the problem function.

[4]Linear convergence < super-linear convergence < quadratic convergence.

Within this function a quadratic programming sub-problem is built and solved using a modified update for the estimate of the Hessian of the Lagrangian. The QP sub-problem is solved with the function *'qp'* of MATLAB. The general notation for the problems to be solved with this function is

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{d}^T\mathbf{x} \text{ such that } \mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

The syntax of *qp.m* is

```
[X,lambda,how] = qp(H,d,A,b,VLB,VUB,x0,NEC,display_flag);
```

with `x0` as starting point on the estimate of the Hessian matrix `H` and the vector `d` as the set of coefficients of the quadratic objective function. The matrix `A` and vector `b` are the coefficients of the linear constraints. Lower and upper bounds are given by `VLB`, `VUB`. `NEC` is the number of equality constraints.

'qp' of MATLAB uses an active set method, which is also a projection method. Corresponding to the Kuhn-Tucker optimality conditions only constraints active at the solution are significant. Methods where non-active constraints[5] are deleted are called active set methods. The algorithm used in *'qp'* finds an initial feasible solution by first solving a linear programming problem of the form

$$\min_{\mathbf{x}} \mathbf{d}^T\mathbf{x} \text{ such that } \mathbf{A}\mathbf{x} \leq \mathbf{b}.$$

The quadratic programming sub-problem is built upon a quasi-Newton method. Quasi-Newton methods renounce to calculate second order derivatives. The search direction is derived by a multiplication of the gradient of the objective function with a quasi-Newton matrix H_k, therefore

$$\mathbf{s}_k = -H_k \nabla\phi(\mathbf{x}_k)$$

with H_k being an appropriate approximation of $\nabla^2\phi(\mathbf{x}_k)^{-1}$. The following conditions on H_k must hold:

1) H_k must be positive definite,

2) H_k must fulfil the quasi-Newton condition:

$$H_{k+1}(\nabla\phi(\mathbf{x}_{k+1}) - \nabla\phi(\mathbf{x}_k)) = \mathbf{x}_{k+1} - \mathbf{x}_k,$$

[5] Constraints where the Lagrange multiplier for the constraint is negative.

3) H_{k+1} is calculated from H_k, $\mathbf{x}_{k+1}$, $\mathbf{x}_k$ $\nabla\phi(\mathbf{x}_{k+1})$ and $\nabla f(\mathbf{x}_k)$ using a rank-2-correction, hence

$$H_{k+1} = H_k + \mathbf{l}_k\mathbf{l}_k^T - \tilde{\mathbf{l}}_k\tilde{\mathbf{l}}_k^T,$$

with $\mathbf{l}_k$, $\tilde{\mathbf{l}}_k \in \mathcal{R}^n$.

5.5.2.1 Updating the approximation to the Hesse matrix

The most known and used methods for updating the approximation to the Hesse matrix of the Lagrange function $\mathbf{H}_k$, introduced in the previous section (DFP and BFGS algorithm) must be given to illustrate the method used within MATLAB.
Hessian update, Davidon, Fletcher[33], Powell[90] (DFP):

$$\begin{aligned} H_{k+1} &= H_k + \frac{p_k p_k^T}{p_k^T q_k} - \frac{H_k q_k q_k^T H_k}{q_k^T H_k q_k} \\ \text{where} \qquad p_k &= \mathbf{x}_{k+1} - \mathbf{x}_k \\ q_k &= \nabla\phi(\mathbf{x}_{k+1}) - \nabla\phi(\mathbf{x}_k) \end{aligned} \tag{5.7}$$

Hessian update, Broyden[12], Fletcher[33](, Goldfarb[43], Shanno[102] (BFGS):

$$\begin{aligned} H_{k+1} &= H_k + \left(1 + \frac{q_k^T H_k q_k}{p_k^T q_k}\right)\frac{p_k p_k^T}{p_k^T q_k} - \frac{1}{p_k^T q_k}(p_k q_k^T H_k + H_k q_k p_k^T) \\ \text{where} \qquad p_k &= \mathbf{x}_{k+1} - \mathbf{x}_k \\ q_k &= \nabla\phi(\mathbf{x}_{k+1}) - \nabla\phi(\mathbf{x}_k) \end{aligned} \tag{5.8}$$

As a starting point, H_0 can be set to any symmetric positive definite matrix, for example, the identity matrix I. The gradient information is either supplied through analytically calculated gradients, or derived by partial derivatives using numerical differentiation methods, e.g. finite differences. This involves perturbing each of the design variables, $\mathbf{x}$, and subsequent calculation of the rate of change in the objective function.

MATLAB uses a modified Hesse matrix updating algorithm within *'qp'*. This modified algorithm is called dual or complementary algorithm of the DFP updating method due to its close relationship to DFP.

To avoid that H_k within the BFGS update is not positive definite the MATLAB update uses the inverse matrix of H_k for the iterations. When using $B_k = H_k^{-1}$ for all k, the following

rank-2-correction for B_k can be developed:

$$\begin{aligned} B_{k+1} &= B_k + \frac{q_k q_k^T}{q_k^T p_k} - \frac{B_k p_k p_k^T B_k^T}{p_k^T B_k p_k} \\ \text{where} \qquad p_k &= \mathbf{x}_{k+1} - \mathbf{x}_k \\ q_k &= \nabla\phi(\mathbf{x}_{k+1}) - \nabla\phi(\mathbf{x}_k) \end{aligned} \tag{5.9}$$

Additionally, a BFGS update using Cholesky factorisation of Gill et al.[36] can be used. This update has been used by Gill to prove positive definiteness of the BFGS formula. The calculation of the updated Hesse matrix uses the upper triangular non singular matrix R such that $H_k = R^T R$. The BFGS formula may be written as

$$H_{k+1} = R^T W R$$

where the matrix W is given by

$$W = \mathrm{I} - \frac{\bar{p}\bar{p}^T}{\bar{p}^T\bar{p}} + \frac{\bar{q}\bar{q}^T}{\bar{q}^T\bar{p}},$$

with $\bar{p} = Rp_k$ and $\bar{q} = (R^T)^{-1}q_k$. This formula is disabled for use because it is less robust than the above method and slower.

5.5.2.2 Quadratic programming solution

If the Hesse matrix has been updated within the major iteration of the SQP method, a sequential quadratic programming problem is solved.
QP

$$\begin{aligned} \min_{\mathbf{s}\in\mathcal{R}^n} \quad qp(\mathbf{s}) = \quad & \tfrac{1}{2}\mathbf{s}^T H_k \mathbf{s} + \mathbf{d}^T\mathbf{s} \\ & \mathbf{A}_j\mathbf{s} = \mathbf{b}_j, \qquad j = 1, \ldots, m_e \\ & \mathbf{A}_j\mathbf{s} \le \mathbf{b}_j, \qquad j = m_e+1, \ldots, m. \end{aligned} \tag{5.10}$$

The solution procedure for this problem involves two phases. In the first phase a feasible point (if one exists) is calculated, in the second phase an iterative sequence of feasible points are generated that converge to the solution. In this method an active set is maintained, $\bar{I}_k$, which is an estimate of the active constraints at the solution point. Equality constraints always remain in the active set, $\bar{I}_k$.

To form a basis for the search direction $\hat{\mathbf{s}}_k$, $\bar{I}_k$ is updated at each iteration k. The special notation $\hat{\mathbf{s}}_k$ is used to distinguish the search direction in QP from $\mathbf{s}_k$ in the major iterations of the SQP method. The search direction $\hat{\mathbf{s}}_k$ minimises the quadratic objective function while remaining on any active constraint boundaries. The feasible subspace for $\hat{\mathbf{s}}_k$ is formed from a basis, Z_k whose columns are orthogonal to the estimate of the active set $\bar{I}_k$, hence

$$\bar{I}_k \times Z_k = 0.$$

The matrix Z_k is formed from the last $m - l$ columns of the QP factorisation of the matrix $\bar{I}_k$, where m is the number of constraints, l is the number of active constraints and $l < m$. Therefore with $\mathcal{H}$ being the Housholder transformation of $\bar{I}_k$ and R being an upper triangular matrix:

$$Z_k \subset Q = \mathcal{H}_n \cdots \mathcal{H}_2 \mathcal{H}_1$$

where

$$Q\bar{I}_k = \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Having calculated Z_k, a new search direction of the QP problem is sought that minimises the quadratic object function $qp(\mathbf{s})$. $\hat{\mathbf{s}}_k$ is in the null space, the orthogonal complement, of the active constraints, that is, $\hat{\mathbf{s}}_k$ is a linear combination of the columns of Z_k; $\hat{\mathbf{s}}_k = Z_k\mathbf{p}$ for some vector $\mathbf{p}$.

By substituting $\hat{\mathbf{s}}_k$ in the quadratic objective function $qp(\mathbf{s})$ becomes

$$qp(\mathbf{p}) = \frac{1}{2}\mathbf{p}^T Z_k^T H Z_k \mathbf{p} + \mathbf{d}^T Z_k \mathbf{p}$$

Assuming the Hesse matrix H is positive definite (which is the case in the illustrated implementation of SQP), then the minimum of the quadratic objective function $qp(\mathbf{p}) \subset Z_k$ occurs when $\nabla qp(\mathbf{p}) = 0$, hence differentiation with respect to $\mathbf{p}$ yields:

$$\nabla qp(\mathbf{p}) = Z_k^T H Z_k \mathbf{p} + Z_k^T \mathbf{d},$$

$\nabla qp(\mathbf{p})$ is referred to as the projected gradient of the quadratic objective function because it is the gradient projected in the subspace defined by Z_k. The Term $Z_k^T H Z_k$ is called the projected Hessian. To derive a step size $\hat{\mathbf{s}}_k = Z_k^T\mathbf{p}$ the solution of the system of linear equations must be calculated

$$Z_k^T H Z_k \mathbf{p} = -Z_k^T \mathbf{d}.$$

A step is then taken of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha Z_k^T \mathbf{p} = \mathbf{x}_k + \alpha \hat{\mathbf{s}}_k$$

Due to the quadratic nature of $qp(\mathbf{s})$ there are only two choices of step length α at each iteration. If no violation of constraints occurs a step of unity along $\hat{\mathbf{s}}_k$ is the exact step to the minimum of the QP problem. Otherwise, the step along $\hat{\mathbf{s}}_k$ is smaller than unity, hence a new constraint is included in the active set $\bar{I}_k$ for the next iteration.

When n independent constraints are included in the active set $\bar{I}_k$, and no location of the minimum can be found, Lagrange multipliers $\mathbf{l}_k$ are calculated that satisfy the non-singular set of linear equations

$$\bar{I}_k^T \mathbf{l}_k = \mathbf{d}.$$

$\mathbf{x}_k$ is the optimal solution of the QP problem if all elements of $\mathbf{l}_k$ are positive. However, if any component of $\mathbf{l}_k$ is negative, and it does not correspond to an equality constraint of $\bar{I}_k$, then the corresponding element is deleted from $\bar{I}_k$ and a new iterate is sought.

5.5.2.3 Merit function for the SQP step length

When even one constraint function in non-linear, it is not straightforward (and may even be impossible) to generate a sequence of iterates that exactly satisfy a specified subset of the constraints. If feasibility is not maintained, then in order to decide whether $\mathbf{x}_{k+1}$ is a 'better' point than $\mathbf{x}_k$, it is necessary to define a merit function that somehow balances the usually conflicting aims of reducing the objective function and satisfying the constraints.

The solution of the QP sub problem (Equation 5.10) is a vector $\mathbf{s}_k$, which is used within the SQP algorithm to form a new iterate

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k.$$

The step length parameter α_k is determined for each iteration k of the SQP algorithm in order to produce a sufficient decrease in a merit function. MATLAB implemented a merit function by Powell[90] of the form

$$\mathcal{M} = \phi(\mathbf{x}) + \sum_{j=1}^{m_e} r_j \cdot \mathbf{g}_j(\mathbf{x}) + \sum_{j=m_e+1}^{m} r_j \cdot \max\{0, \mathbf{g}_j(\mathbf{x})\}$$

with the recommended settings for the penalty parameter r_j

$$r_j = (r_{k+1})_j = \max_j \left\{ \mathbf{l}_j, \frac{1}{2}((r_k)_j + \mathbf{l}_j) \right\}, \qquad j = 1, \ldots, m,$$

allowing positive contribution from constraints that are inactive in the QP solution but were recently active. The initial penalty parameter r_{j_0} is represented by the Euclidean norm of

$$r_{j_0} = \frac{|| \nabla \phi(\mathbf{x}) ||}{|| \nabla \mathbf{g}_j(\mathbf{x}) ||}.$$

With this initial representation it is ensured that constraints with smaller gradients do have larger contribution to the penalty parameter then constraints with larger gradients. This is the case for active constraints at the solution point.

5.6 Conclusions on simulation and optimisation

The simulation of the warehouse plant has been introduced and discussed. It has been stated that the closed coupled combination of the various models is more precise and accurate then the weakly coupled system.

In Chapter (2) several different optimisation strategies have been considered and a brief selection of the optimisation strategies due to their capability and applicability has been given.

This has been further promoted in this chapter to focus on the proposed strategy for control of the sloshing motion with an optimisation algorithm. Therefore an optimal control problem with the objective of motion in minimal time has been formulated. Firstly in continuous form and secondly in discretised form, to be solved within an iterative procedure.

A direct solution method has been proposed to solve the initial value problem. This method is a practical approach type, using engineering knowledge and experience to establish realistic estimates for initial parameters and to finding the solutions. Solutions are less accurate than using multiple shooting or collocation methods, but easier to find since the convergence radius is larger.

The summarised aspects of the direct shooting method in combination with the MATLAB SQP algorithm are:

- The gradients are calculated numerically.
- Complicated model functions can be used.
- The optimisation problem can be implemented very quickly.

- The QP solver within the SQP can be exchanged.

However, the following drawbacks can be distinguished:

- The numerical calculations are very cost[6] intensive.
- MATLAB does not have parallel processing capabilities.

The general advantage of MATLAB and its OPTIMIZATION TOOLBOX is the uncomplicated pre- and post-processing. If the model is set up correctly, hence having the desired output, the optimisation is able to calculate results based on given initial values. The output of the optimisation can be studied and visualised with the graphical tools of MATLAB.

[6] Computational time.

5.7 Nomenclature

a, b	boundary values of the time for the control
$\mathbf{A}$	general matrix of values
b, d	general vectors of values
$\mathbf{B}$	inverse matrix of H
d	solution deviation
e	error value
fun	function vector of the plant
$\mathbf{F}$	general force vector
g	natural gravity
g	constraint function vector
h	iterative step
H	Hesse matrix (approximation to Hesse matrix)
$\mathcal{H}$	Householder transformation, $\mathcal{H} = \mathrm{I} - \frac{2}{\|\mathbf{b}\|_2^2}\mathbf{b}\mathbf{b}^T$
I	identity matrix
$\boldsymbol{I}$	matrix of active restrictions
$\bar{\boldsymbol{I}}$	estimate of active constraints at the solution
k	iteration level
k	coefficient of the spring
lb	lower bounds
l, l_j	vector of Lagrange multipliers, single multiplier
$\tilde{\mathbf{l}}$	vector of approximation to the Lagrange multipliers
L	Lagrange function
m	mass
m, m_e	constraint function dimension, number of equality constraints
$\mathcal{M}$	merit function
p	pressure
p, q	order of Runge-Kutta equations
p	projected vector of the search direction
P	feasible domain of the optimisation
Q	orthogonal matrix, product of $\mathcal{H}_n \cdots \mathcal{H}_1$
R	upper triangular matrix

r	penalty parameter
$\mathcal{R}$	domain of real numbers
s	domain operator
$\mathbf{s}$	vector of search directions in SQP
$\hat{\mathbf{s}}$	vector of search directions in QP
t	time
T	time constant
$\mathbf{ub}$	upper bounds
$u, \vec{u}(u,v,w)$	velocity vector with cartesian components u, v, w
$\mathbf{u}$	continuous control or design function
$\tilde{\mathbf{u}}$	continuous approximated control or design function
$x, \vec{x}(x,y,z)$	position vector with cartesian components x, y, z
$\mathbf{x}$	vector of control or design parameters
$\tilde{\mathbf{x}}$	approximation to the control or design parameters
z, $\dot{z}$, $\ddot{z}$	position, velocity, acceleration in the horizontal direction of the conveyor system
Z	matrix, with orthogonal columns of $\bar{I}$
α	step length control
α_i, $\beta_{i,j}$, γ_i	coefficients of the Runge-Kutta equations
δ	actual error
Δt	time step
μ_s	friction coefficient of the slider
ψ, ϕ	functions, in general
τ	admissible error
Ω	domain, volume
ν	approximation of the velocity vector
ζ	approximation of the state

Chapter 6

Theoretical results, practical verification

In this chapter final findings from the optimisation and preliminary results on the small scale model rig, are evaluated. These preliminary results were necessary to test the suitability of the rig for application on the given problem, and in particular to evaluate the given constraints for the optimisation code.

Therefore in the first part of this chapter, constraints, applied by the small scale servo rig are evaluated. These constraints influence the height of the fluid within the container and the velocity and acceleration specifications.

The second part of the evaluation consists of the theoretical optimal control calculations of the fluid and cart motion, and the subsequent experimental verification of these results.

Within the theoretical optimal control calculations several different local and global parameters of the container motion are varied to investigate their influence on the local optimal motions of the cart and the fluid.

Some particular interesting motions and configurations have been used to be simulated with the small scale model rig. These motions have been monitored and recorded with a video system.

Finally the results of the optimal control calculations and the corresponding experimental verifications are summarised and linked within the verification of the optimisation results.

6.1 Limitations applied by the servo rig

In this section the two basic limitations applied by the servo rig are investigated. The first limitation concerns the application of the experimental servo rig itself. The control of the servo motor applies limitations on the motion profile to be used within the experiment. It has only been possible to use a step function input for the acceleration of the cart. It is well known that such acceleration profiles can not reflect reality, thus they can only apply sub-optimality. Additionally the bandwidth and the specification of the velocity and the acceleration are limited.

The second limitation affects the modelling and simulation process of the fluid. The rig has been used to evaluate a limit experimentally and mathematically for the applicability of the fluid flow solver on the given sloshing problem. The applicability is limited by the height of the fluid within the container which is modelled within the code NaSt2D.

6.1.1 The control mode of the experimental rig

The basic idea of the small scale model rig is to execute and verify results obtained with the optimisation procedure. For this purpose a large number of optimal control calculations have been specified particularly to be transferable to the experimental rig, knowing that the involved straight line interpolation of the acceleration profile is not optimal. The optimisation of the interpolation mode has been performed in Chapter (6.2.4). It is intended to use a very simple approach for the control of the servo motor, because of the limited capabilities of the servo rig control and the simplicity of the command programme to be written. Additionally it is easier to observe, control and verify the actual motion of the rig if the control is specified in this way.

In Figure(6.1) a simple trapezoidal and a triangular mode of the velocity profile are illustrated. In general the motor is accelerated with a step input until the maximum velocity is reached; for deceleration the same velocity slope will be used. If the maximum velocity can not be reached within half the time of the motion, a triangular velocity motion profile is executed instead.

Additional investigations have been performed, concerning the specification of the velocities and accelerations within the control of the servo motor.

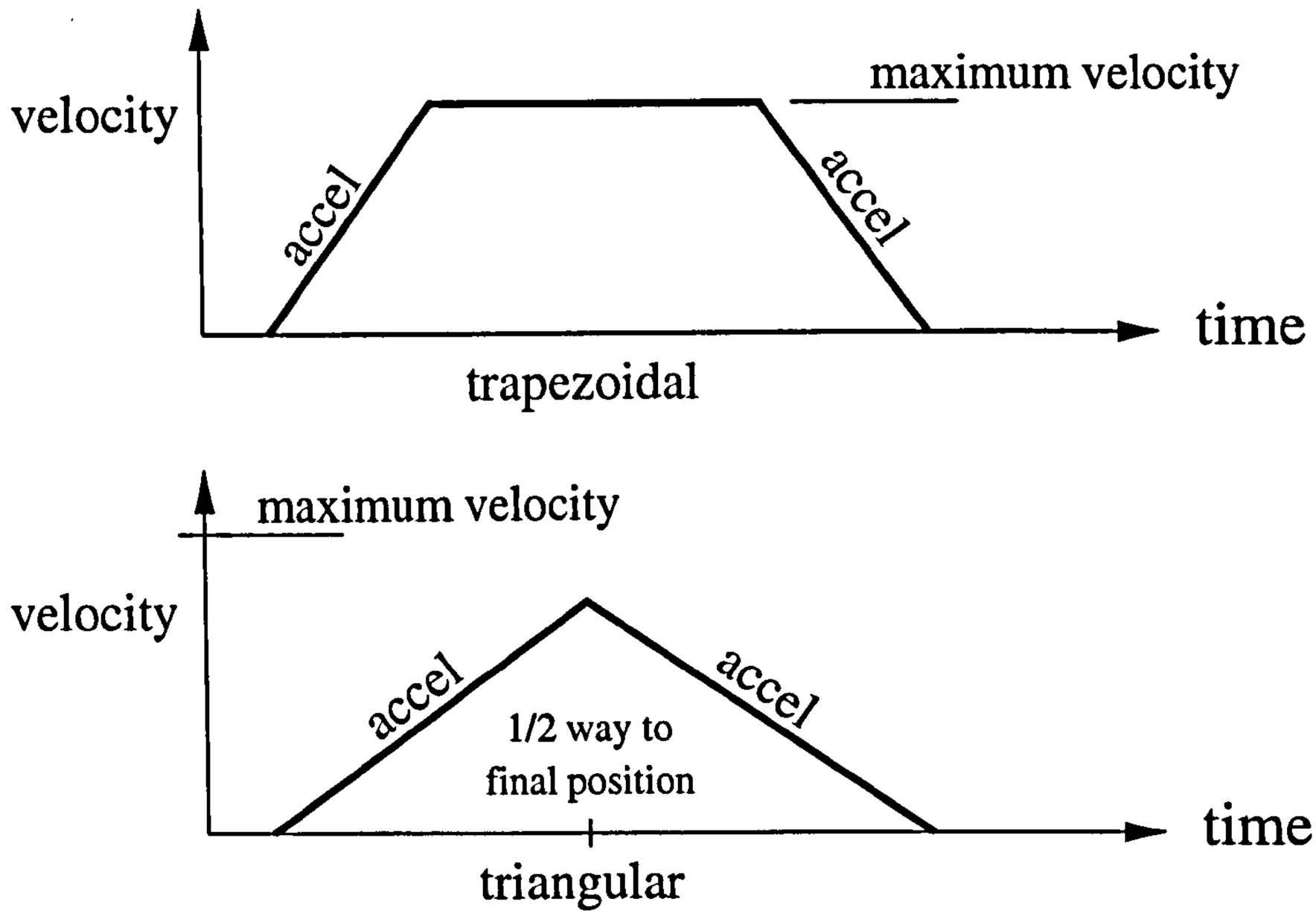


Figure 6.1: Trapezoidal profile control mode.

It has been observed that the behaviour of the servo motors in terms of velocity do not correspond to the command velocity of the motion controller. Experiments have been undertaken to evaluate the response of the servo motors due to specific command velocities of the motion controller in the low velocity mode of the amplifier.

In several experiments, the time has been measured for a high number of revolutions in order to calculate the mean velocity of the servo. To reduce the influence of acceleration and deceleration during the motion the value of the acceleration has been set to its maximum value. Additionally a large number of revolutions has been undertaken to keep the measured time always great enough to be able to reduce the influence of measuring on the overall time.

From the number of revolutions and the measured time a mean velocity has been calculated for every experiment. This velocity has been compared with the velocity calculated with the formula given by the reference manual of the HCTL-1100 motion controller[1]:

$$v_r = \frac{v_q \cdot 60}{Nt} \cdot (rpm \ \ s) \tag{6.1}$$

[1]Note, that the units in brackets must be multiplied with the equation. They are not the unit of the resulting variable.

where v_r is the velocity of the shaft in [rpm], v_q is the programmed velocity in [quadrature counts per sample time](0..255)[2], $1/4N$ is the number of slots in the code-wheel (1000), $t = \frac{16(T+1)}{2000000[1/s]}$ is the sample time[3] (128 [μsec]..2048 [μsec]) and T is the value for the sample time register (15..255).

Finally the quotient of the two velocities has been calculated and plotted in Figure(6.2). The different velocities to be programmed into the HTCL-1100 register are indicated with (v = ..) within the figure. As a further illustration the actual performed velocities have

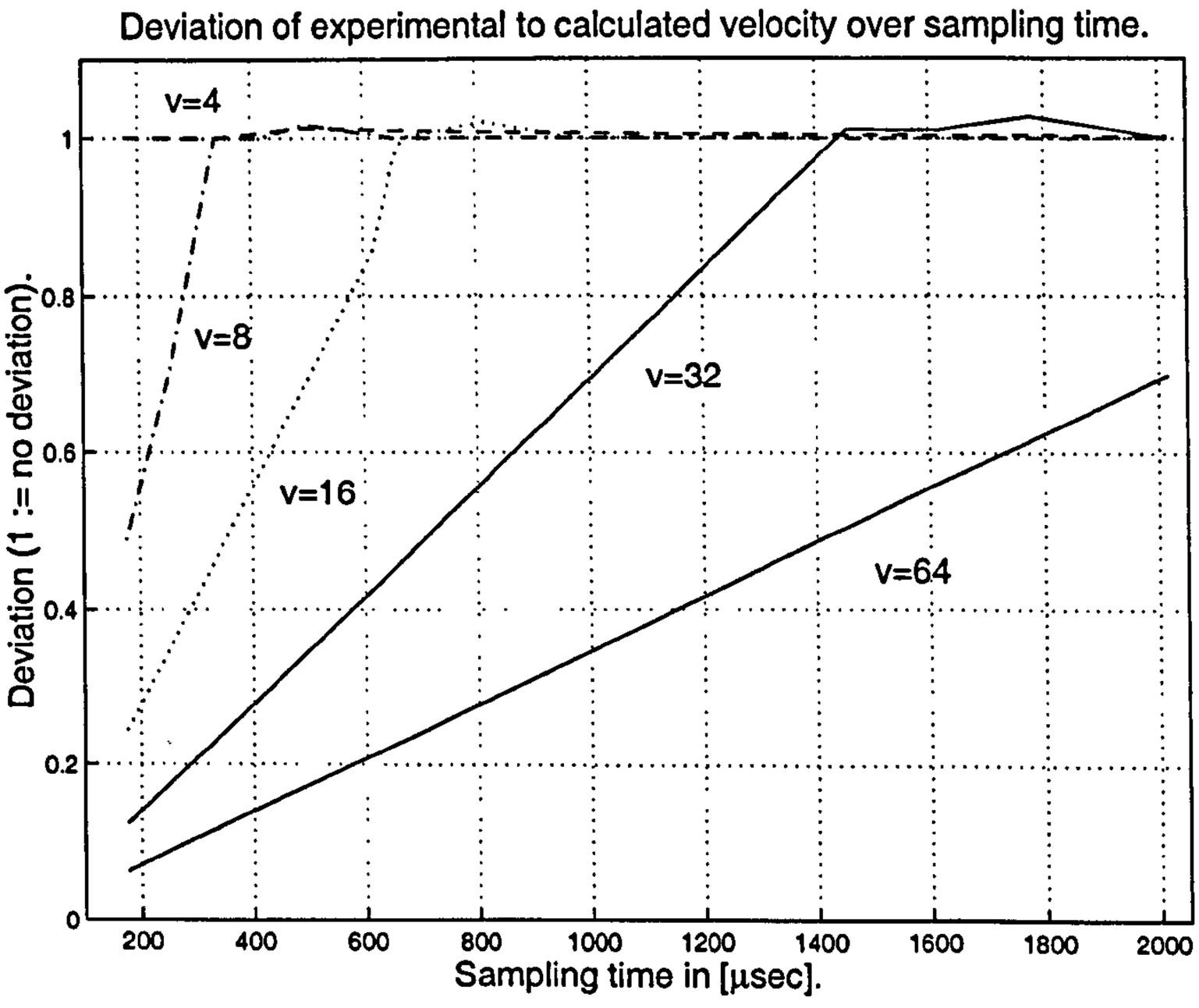


Figure 6.2: Velocity deviation.

been plotted in Figure(6.3) over the sampling time used. It can be observed that there exists a maximum velocity of 333.33 [rpm] which can not be exceeded. It can also be observed that there exists a limit velocity from which no further velocity change will occur, when increasing the programming value of the velocity. Therefore the bandwidth of programmable and actual driven velocity on the rig is limited. The lower constraints on the velocity have no effect on the calculation of the optimal results due to the fact that these velocities are too low to allow sloshing within the fluid. Whereas the upper constraints on

[2]Number-range, the value can be specified in.

[3]Signal command update time.

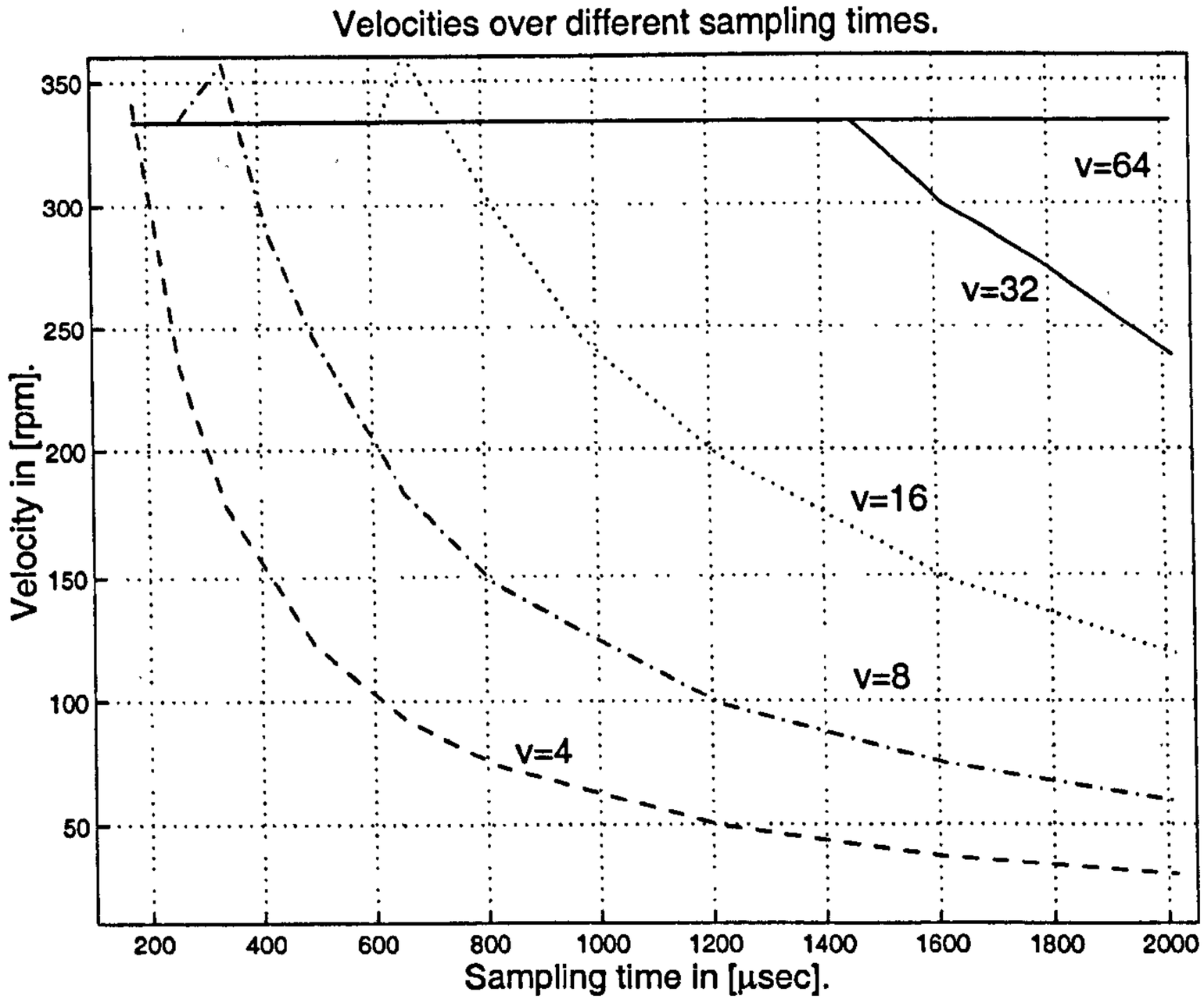


Figure 6.3: Velocities in [rpm].

the velocity introduce the specification of a boundary for the maximum velocity within the calculation of the optimal results.

The values for the velocity and the acceleration of the cart and its command velocity for the motion controller must be converted to the values used within the optimisation and vice versa. Special emphasis must be put on the conversion concerning the non-dimensionalized values within NASt2D. The following equation holds, to convert the velocity used within NASt2D to the command velocity for the motion controller.

$$v_q = \frac{100}{9} v_r^* v_\infty N t \left(\frac{1}{m}\right) \tag{6.2}$$

where v_q is the velocity in [quadrature counts per sample time] (0..255), v_r^* is the non-dimensionalized velocity (within NASt2D) and v_∞ is the characteristic velocity [m/s] (within NASt2D). The variables N, t and T must be specified similar to equation (6.1). The limit velocity of the servo motor can be given in several different states:

velocity of servo motor	333.33	[rpm]
velocity of rig	0.50	[m/s]
nondimensionalized velocity using $v_\infty = 0.1$ [m/s]	5.00	[ ]
command velocity using $T = 22$, $t = 184$ [μsec]	4.08889	[quadrature counts/sample time]

The command velocity must be rounded to the next integer value to be programmed into the command velocity register of the motion controller HTCL-1100[4].

The specification of the acceleration is similar to the specification of the velocities. The experiments, used for the evaluation of the feasible domain for the velocity specification has also been used to observe the acceleration behaviour of the experimental rig.

Special focus has been on the influence of the sampling time on the acceleration of the servo motors. It has been observed that there is no change in the acceleration of the servo motor in the full range of sample time (128 [μsec] to 2048 [μsec]) when using a very low command acceleration. In all other cases, good correlation has been observed between the motion of the rig and the command acceleration.

Therefore, the following equation given in the data sheet of the HTCL-1100 motion controller can not be true for the whole parameter domain. This equation is given to convert the desired acceleration a_r of the servo motor to the command acceleration a_q for the motion controller.

$$a_q = \frac{256}{60} a_r N t^2 \left(\frac{1}{rpm \ \ s} \right) \tag{6.3}$$

The variables N, t and T must be specified similar to equation (6.1).

Sloshing within the container is only possible in cases of medium to severe acceleration of the cart. Therefore the difficulties in specifying low accelerations do not apply in the problem of optimal trajectory planning with open fluid filled containers.

The following equation holds, to convert the acceleration used within NAST2D to the command acceleration for the motion controller.

[4]Note that the corresponding velocities must be recalculated.

$$a_q = \frac{25600}{9} a_r^* \frac{v_\infty^2}{L} N t^2 \left(\frac{1}{m}\right) \tag{6.4}$$

where a_q is the acceleration in [quadrature counts per (sample time)^2] (0..255), a_r^* is the non-dimensionalized acceleration (within NAST2D) and L is the characteristic length [m] (within NAST2D). The variables v_∞, N, t and T must be specified similar to equation (6.2).

As a result, the acceleration of the servo motor can be given in several different states:

acceleration of servo motor	1000	[rpm/s]
acceleration of rig	1.50	[m/s^2]
nondimensionalized acceleration using $v_\infty = 0.1$ [m/s], $L = 0.1$ [m]	15.00	[ ]
command acceleration using $T = 124$, $t = 1000$ [μsec]	17.0667	[quadrature counts/sample time^2]

The command acceleration must be rounded to the next integer value to be programmed into the command acceleration register of the motion controller HTCL-1100[5].

6.1.2 Limitation of fluid height

Based on the standard experimental specifications of the model rig, the fluid height within the cube can be varied. Four different heights have been investigated and a comparison has been calculated between the results of the experiments and the predictions of the NAST2D simulation.

The fluid level has been changed within the cube of the experimental setting. It has been filled with a fluid height of 20, 40, 60 and 80 [mm]. The maximum height of the cube and therefore the sloshing limit is 100 [mm]. While making the maximum velocity available to the servo motors, the acceleration has been increased until the fluid started to slosh over the top level of the cube. This specification results in a triangular velocity profile of the servo motor, because the maximum velocity can not be reached (for illustration see Figure(6.1)). The obtained values of command acceleration and sampling time have been converted to a force distribution on the container to be used to simulate the fluid behaviour with the CFD-code NAST2D. Theoretically this should result in a figure of the amplitudes, of the

[5]Note that the corresponding accelerations must be recalculated

water height on the right and left boundary of the container, where the upper limit of the cube must be reached at least once.

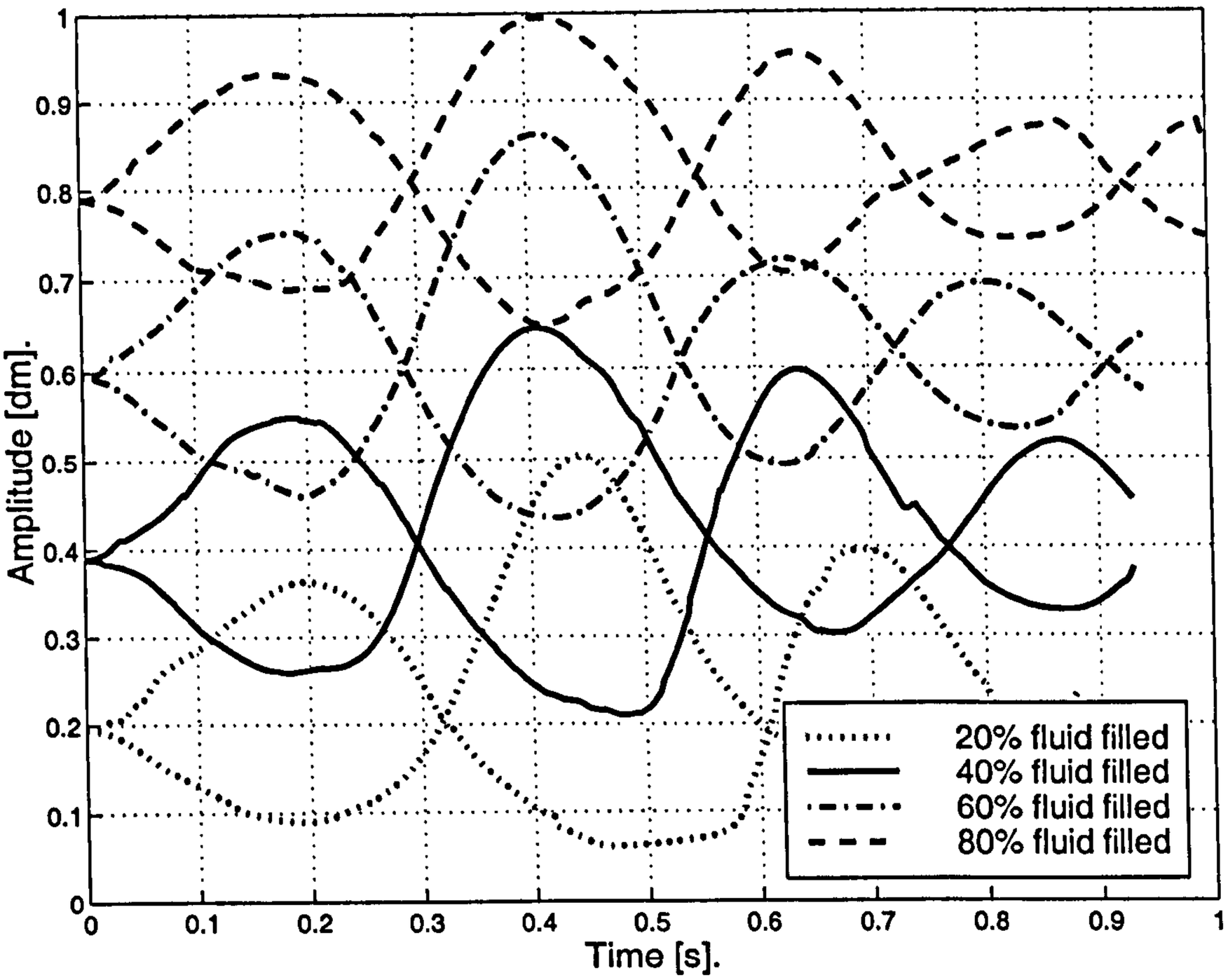


Figure 6.4: Variation of fluid height - Amplitudes of the fluidmotion.

In Figure(6.4) the different amplitudes, calculated mathematically, on the left and right boundary of the cube are shown. It can be seen that only the fluid height of 80 [mm] achieves the goal of reaching the upper limit of the cube. Therefore, in the other cases, the fluid is not modelled and simulated accurately.

This behaviour of the simulation can be explained with the model of the fluid. The fluid is modelled with the CFD-code NAST2D. This code is capable of modelling laminar, viscous, incompressible flow.

In the experiments fluid separation in the form of single fluid droplets were observed for the fluid dimension cases of 20 and 40 [mm] fluid height. This indicates that the fluid is not strictly a laminar flow field. There must be turbulent features in the physical representation of flow field as well. Consequently different physical laws must be implemented in the code to allow computation of all possible fluid levels.

The main problem with the model is not strictly the fluid height within the container. It is more important to limit the allowed sloshing height. Therefore, the following limitation has been heuristically formulated for the fluid flow within the cube:
the fluid model is only accurate if the amplitude motion of the fluid does not exceed 20% of the initial fluid height. Further experiments would be needed to evaluate the influence of the container ratio on this motion limitation.

It has been shown that the fluid flow height is one of the crucial elements in the simulation which must be limited within the model. Only if the fluid height is limited to a certain domain can it be guaranteed that the model of the fluid flow and the fluid flow itself react in a comparable way.

Based on these findings the optimal control calculations have been based on the rectangular cube of the experimental rig with a edge length of 100 [mm] and a filling height of 80 [mm] minimum.

In general the following setting is used for standard experimental verification:

Dimension of the cube	100 [mm] × 100 [mm] × 100 [mm]
Initial surface height	80 – 95 [mm]
Domain decomposition	30 × 26 cells
Fluid properties	Water at 20 °C
Distance to travel	≤ 90 [mm]
Parameter interpolation mode	straight-line
Initial parameter set	Trapezoidal, segmentation 1/3 acceleration, 1/3 steady velocity, 1/3 deceleration final time: 0.55±0.05 [s] acceleration: 1.32/1.58 [m/s^2]

These limitations are due to the specifications of the model rig, the limitations described in Section (6.1) and the findings in the initial experiments.

6.2 Optimisation and simulation results

In order to evaluate knowledge about the characteristic behaviour of the fluid and the cart in motion and the performance of the optimisation procedure using sequential quadratic programming, several simulations of the fluid and the optimisation have been performed.

This has been accompanied by experimental work on the small scale model rig to specify the actual range of use for the mathematical model and simulation. The following sections illustrate some of these theoretical calculations, and where possible, link their result to the experimental model rig.

The general used motion profile, used to be optimised for various parameter settings has been illustrated and explained in Section (6.1). Some limitations have been formulated on the acceleration profile due to the limitations of the experimental servo rig configuration. These limitations concern the fluid height within the container and the specification of the velocity and the acceleration.

The following will focus on the results achieved by the variation of different parameters and properties of the motion as described below. These parameters implement firstly a major impact on the performance of the optimisation code, secondly a variation on the optimal fluid motion or thirdly a change in both characteristics. These parameters are:

- length/height ratio of the container
- distance to travel
- viscosity of the fluid
- influence of interpolation mode

The influence of variations in these parameters on the optimal trajectory will be studied.

The final investigation on the influence of the interpolation mode of the control is based on a study of the necessity of smooth interpolation to guarantee practical applicability of the proposed control strategies for the optimal motion of liquid filled containers.

Finally the results obtained and illustrated within this chapter are summarised.

6.2.1 Variation of length of container

In Section (6.1.2) it has been illustrated that a certain fluid level must be given within the container to guarantee similarity between simulated model and reality. Therefore all the optimal control calculations in this section have been performed with a fluid height of 80 [mm]. Additionally the maximum velocity has been limited to a value of 0.50 [m/s]. This is the maximum velocity applicable on the small scale model rig.

In general the following setting is used for the optimal control calculation:

Dimension of the cube (L×H [×D])	50-200 [mm] × 100 [mm] × 100 [mm]
Initial surface height	80 [mm]
Domain decomposition	30 × 26 cells
Fluid properties	Water at 20 °C
Distance to travel	90 [mm]
Parameter interpolation mode	straight-line (Step function)
Initial parameter set	Trapezoidal, segmentation 1/3 acceleration, 1/3 steady velocity, 1/3 deceleration final time: 0.55±0.05 [s] acceleration: 1.32/1.58 [m/s^2]

Due to the fact that the optimal solutions obtained result only in local minimal solutions, several optimal control computations have been started with different initial parameter settings. This is indicated by the variation of final time and acceleration within the initial parameter set. The obtained results have been used to calculate mean values for the characteristic values indicated in Table (6.1).

Dimension	50[mm]	80[mm]	100[mm]	120[mm]	200[mm]
Number of iterations	12.67	11.33	11.00	14.20	14.17
Number of function evaluations	51.33	47.17	56.33	87.40	69.50
Optimal final acceleration time	0.2806	0.4795	0.4825	0.4882	0.6160

Table 6.1: Calculation results of container length variation.

The low number of iterations in the case of 100 [mm] corresponds to the used initial parameter sets. They have been composed from values achieved with experiments on the small scale model rig. The cube in the experiment has the same dimensions (100[mm] × 100[mm] × 100[mm]).

Figure(6.5) shows the progress of the optimisation from one iteration to the next. Representative optimisation runs of all five variants have been used to analyse their behaviour within the optimisation. A deviation between the value of the objective function f^k at iteration level k and the value of the optimal objective function f^* was calculated. A value for the deviation (f^k/f^*) greater then one indicates that the function value at the iteration time k is larger then the final optimal solution for the specific case. A value for the deviation smaller then one indicates that the function value at the iteration time k is less

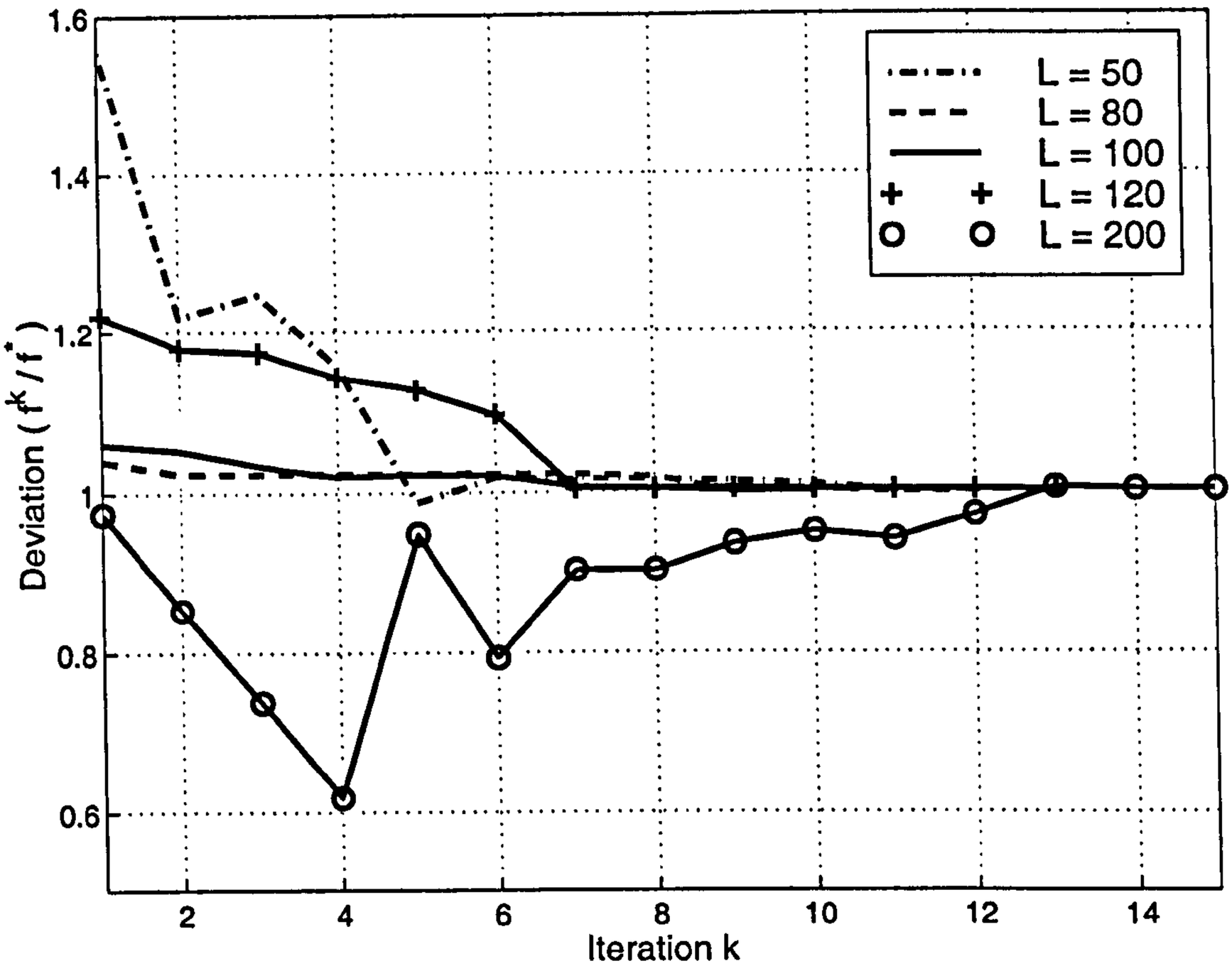


Figure 6.5: Progress of iterations.

then the final optimal solution. In these cases constraints or boundaries are violated.

It can be observed that the iterative procedures for the cases of 80, 100, and 120 [mm] do start quite near to their optimal solution and do iterate quite well towards the optimal solution. All the intermediate results are greater then the final optimal solution. The intermediate solutions within the iterative process are strictly monotonic downwards towards the final solution.

Only the optimisation runs of the case 50 and 200 [mm] container length show extraordinary behaviour.

In the case of cube length 50 [mm] the optimisation starts with an initial set of parameters which can be seen is quite far away from the final optimal solution, but the initial solution provides a fluid motion without sloshing. This leads to a fast iterative process towards the optimal solution, only disturbed by minor back-slopes.

In the case of cube length 200 [mm] the optimisation starts with an initial set of parameters which is very near to the final optimal solution. But in this case the initial solution provides

a fluid motion with sloshing, because the final time in this iteration is smaller then the final time of the optimisation. This leads to a slow iterative process towards the optimal solution with oscillating behaviour within the functions of the iterations. The optimisation code is not able to increase the final time as much as needed[6] to result in a parameter function which allows a fluid motion without sloshing. It is only in the last three iterations the optimisation code is able to specify the control function of the motion in a way that the fluid is not sloshing.

It has been observed in several other calculations, that the optimisation code iterates quicker towards an optimal solution if the time needs only to be reduced (from iteration to iteration). This is the case for dimensions of cube-length 50/80/100 [mm]. In these cases the number of iterations, and the number of function evaluations is smaller than in the cases of dimension 120/200 [mm].

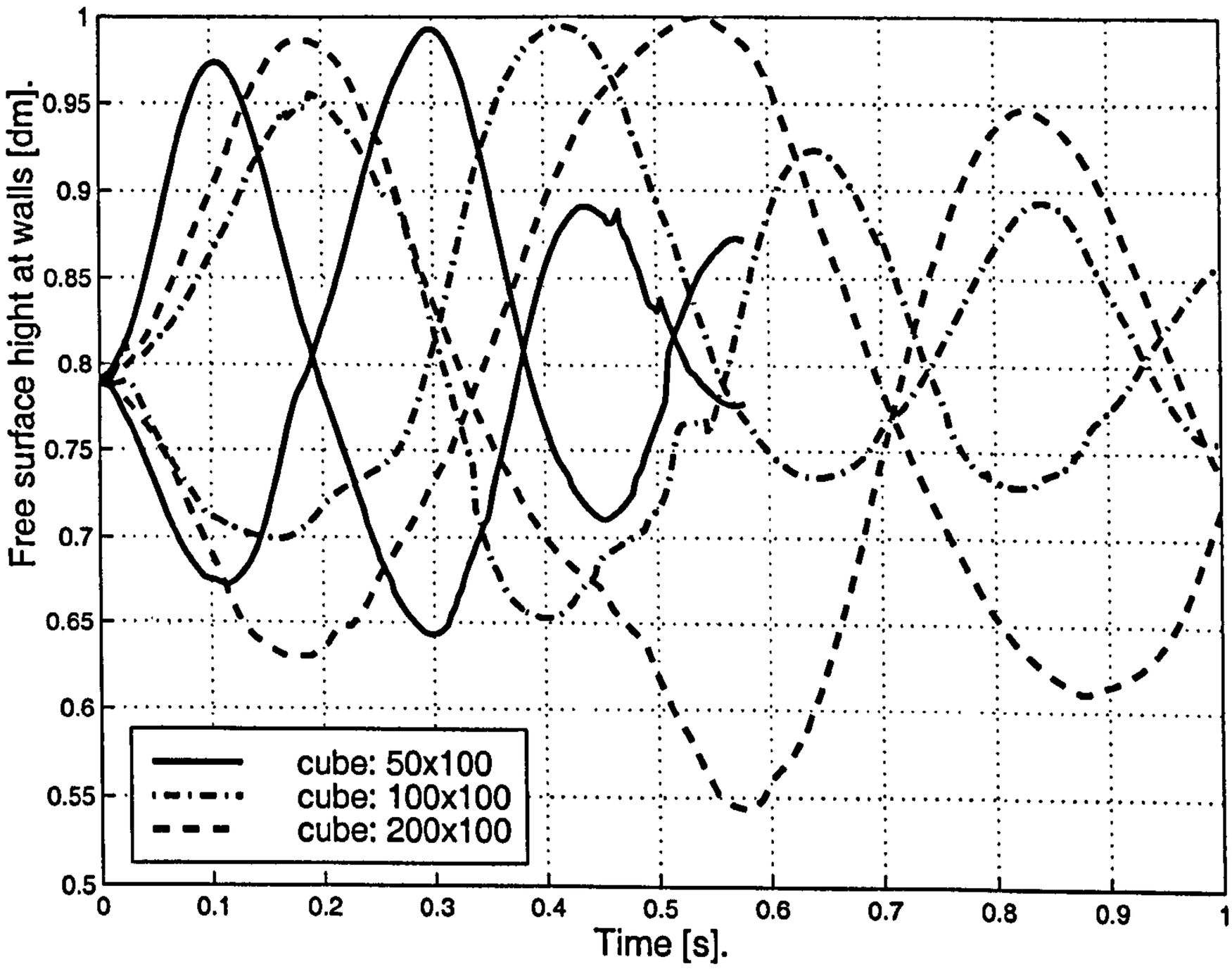


Figure 6.6: Variation of container length - Amplitudes of the fluid.

In Figure(6.6) the amplitudes on the left and right boundary for the time optimal case are given. Please notice that the lower half of the cube is not printed for better illustration.

[6] Quotient of the deviation $f^k/f^* > 1$.

In the case of sloshing, the graphs would go over the top. In this figure one can see that the restriction in terms of sloshing resulted on a limitation of the acceleration (because the upper limit of the cube has been reached and the acceleration has not been restricted initially). It can be observed that the fluid reaction is slow and progressively lethargic, the more the ratio of the container dimensions change from a slim upright cube towards a long horizontal one. Additionally the frequency of the fluid motion is decreasing. This calculated fluid behaviour corresponds to the fluid motion one would measure within a practical experiment by changing this ratio.

6.2.2 Variation of distance to travel

Another geometrical parameter of the container within the warehouse to be studied is the distance to be travelled. Here only motion in horizontal direction has been studied, since this motion applies the necessary forces on the fluid to allow the fluid to slosh. Again, the fluid height has been fixed to 80 [mm] to guarantee similarity between simulated model and reality.

For the study of variation of distance to travel, following settings were used for the optimal control calculations.

Dimension of the cube (L×H [×D])	100 [mm] × 100 [mm] × 100 [mm]
Initial surface height	80 [mm]
Domain decomposition	30 × 26 cells
Fluid properties	Water at 20 °C
Distance to travel	45-720 [mm]
Parameter interpolation mode	straight-line (Step function)
Initial parameter set	Trapezoidal, time of acceleration equal to time of deceleration, steady velocity otherwise, final time and acceleration adapted to the task

Again, several optimal control calculations with different initial parameter settings were performed to heuristically determine the shape of the optimal solution domain. The obtained results do differ in the amount of iterations, in the amount of function evaluations and in the value of the actual optimal solution.

For practical application the best result, hence the calculation with absolute minimal time would be used. For a general overview on the performance of the optimisation code it

Distance to travel	45[mm]	90[mm]	180[mm]	360[mm]	720[mm]
Number of iterations	12.60	11.00	12.33	23.20	29.33
Number of function evaluations	76.80	56.33	55.33	124.0	179.7
Optimal final acceleration time	0.3613	0.4825	0.6527	1.0222	1.720924

Table 6.2: Calculation results of the distance variation.

is desirable to calculate the values indicated in Table (6.2) using mean values. In this calculations, very unsatisfactory results in terms of optimal solution have been eliminated and do not count within these mean values.

Some sample calculations have been chosen to illustrate the calculation behaviour of the SQP-code. The calculation have been based on the coarse acceleration profile which is used within the experimental servo rig.

Two different optimisations have also been studied in more detail. In Figures(6.7-6.10) the results of an optimisation performed with a desired distance of 45 [mm] are presented.

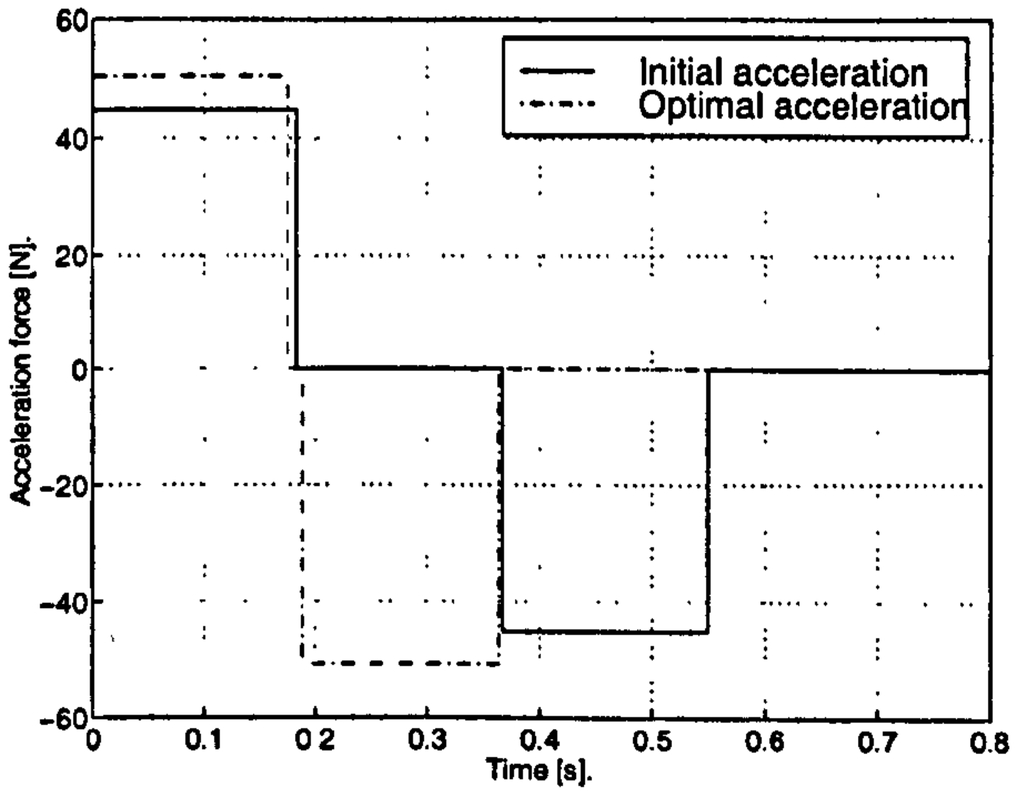


Figure 6.7: Acceleration of the cube.

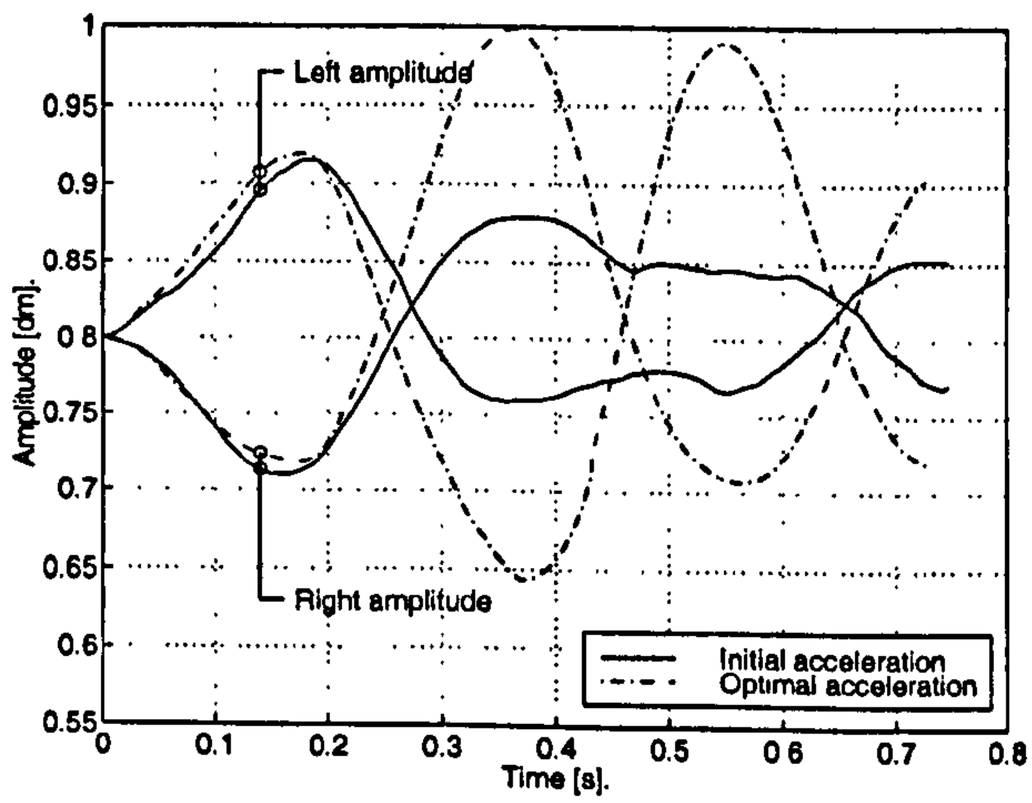


Figure 6.8: Amplitudes of the fluid.

Figure(6.7) shows the initial and optimal acceleration profile. The optimal solution needs only 60 % of the initial time to perform the motion. Figure(6.8) shows the amplitudes of the fluid on the left and the right boundary of the cube. The cart has been accelerated with the optimal acceleration profile shown in Figure(6.7). It is clearly shown in Figure(6.8) that the maximum allowed amplitude has been reached once by the right amplitude of the fluid within the cube.

Figure(6.9) shows the iterative improvement of the solution within the optimisation. Three

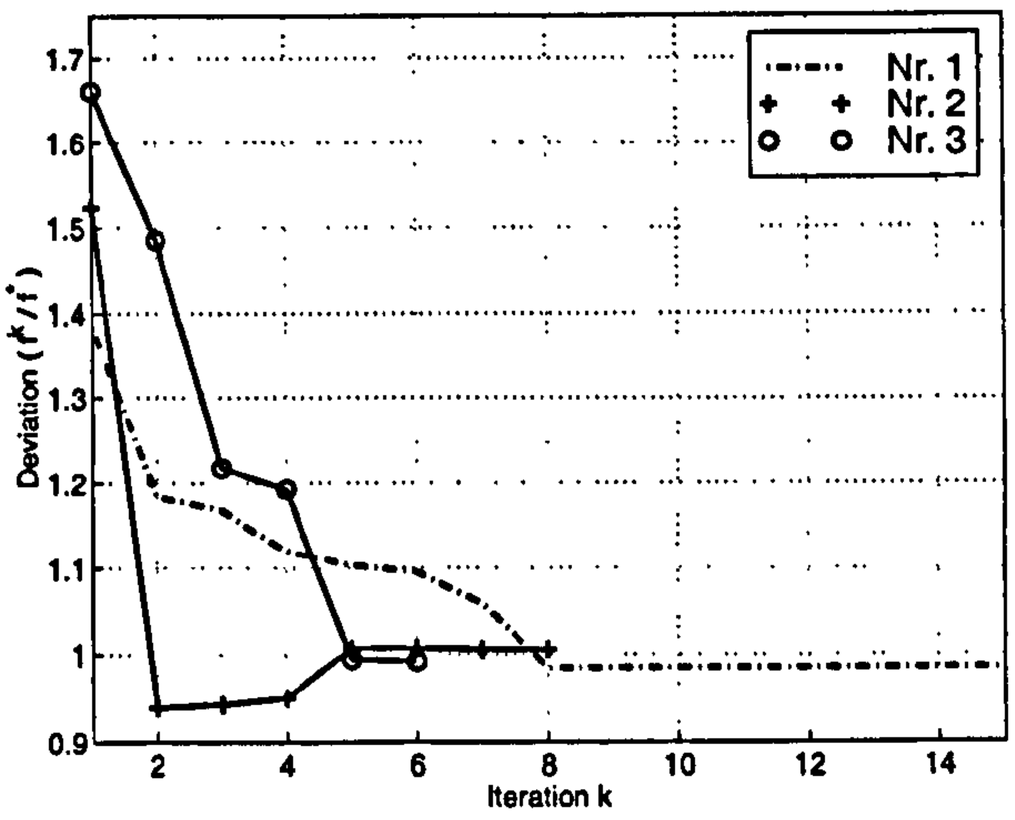


Figure 6.9: Deviation of the solution.

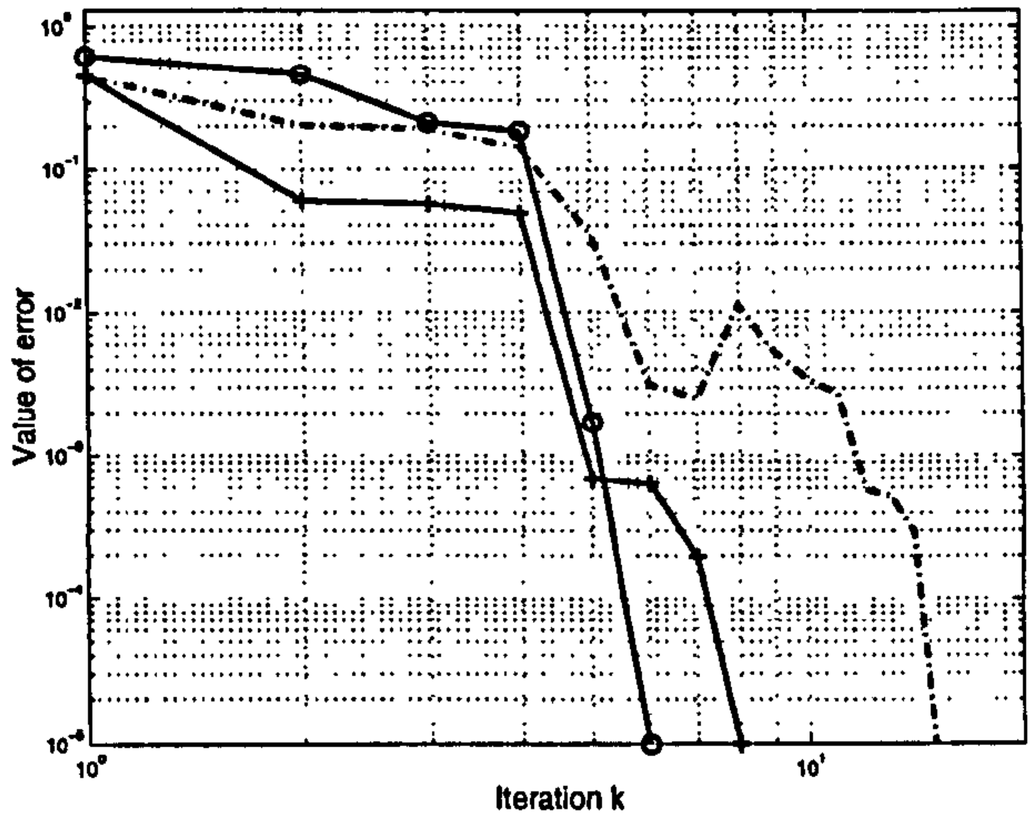


Figure 6.10: Error of the solution.

different optimisations are illustrated within this figure. They differ in their initial parameter setting and their final optimal solution. The unity value of the Y-axes indicates the quotient of specific optimisation solution and the mean value of the final optimal solutions. Hence, if the final solution of one of the shown calculations is less then unity, the specific solution is better then the mean solution. If it is greater then unity, the specific solution is worse then the mean solution.

In all three calculations a steady improvement can be observed from start on, even when the initial time is more than 1.6 times greater then the optimal solution. In all cases the optimal solution has been reached in less than 16 iterations.

Figure(6.10) shows a corresponding distribution of the error within the iterative optimisation process. Note that the line specification is the same as in Figure(6.9). The error function is composed by the maximum of several different error specifications and is given at iteration level k as follows:

$$e^k = e_x + e_v + e_{sl} \tag{6.5}$$

Where e_x is the error on the desired final location (desired final position is 45 [mm]), e_v is the error on the final velocity (desired final velocity is 0 [m/s]) and e_{sl} is the error of sloshing calculated from the maximum sloshing on either right or left boundary.

In this specific optimisation the various errors have the following bandwidth:

- Error 1. Error of 1[mm] deviation between the desired and the calculated final position corresponds to the error value of $e_x = 0.01$. This error is linearly dependent on the

acceleration parameters.

- Error 2. Error of 1[m/s] deviation between the desired and the calculated final velocity corresponds to the error value of $e_v = 1$. This error is also linearly dependent on the acceleration parameters.

- Error 3. Sloshing of 1[mm] over the top of the container corresponds to the error value of $e_{sl} = 0.01$. This error is nonlinearly dependent on the acceleration parameters. In order to increase the influence of this very important and difficult to control error measurement, this value has been multiplied by 100 within the calculations.

It is clearly visible in Figure(6.10), that the maximum error within the optimal control calculations reduces from one iteration to the next. Only the optimisation procedure specified with (Nr. 1) shows a small increase of the error value between iteration seven and iteration eight. This is due to the fact that the maximum error measure within e^k has jumped from one error specification (e_x, e_v or e_{sl}) to another error specification. This is especially conspicuous if at different iteration levels the fluid motion oscillates between sloshing and non sloshing motion.

The second calculation is shown in Figures(6.11-6.14). These are the results of an optimisation performed with a desired distance of 720 [mm]. Note that this is 16 times the distance studied before.

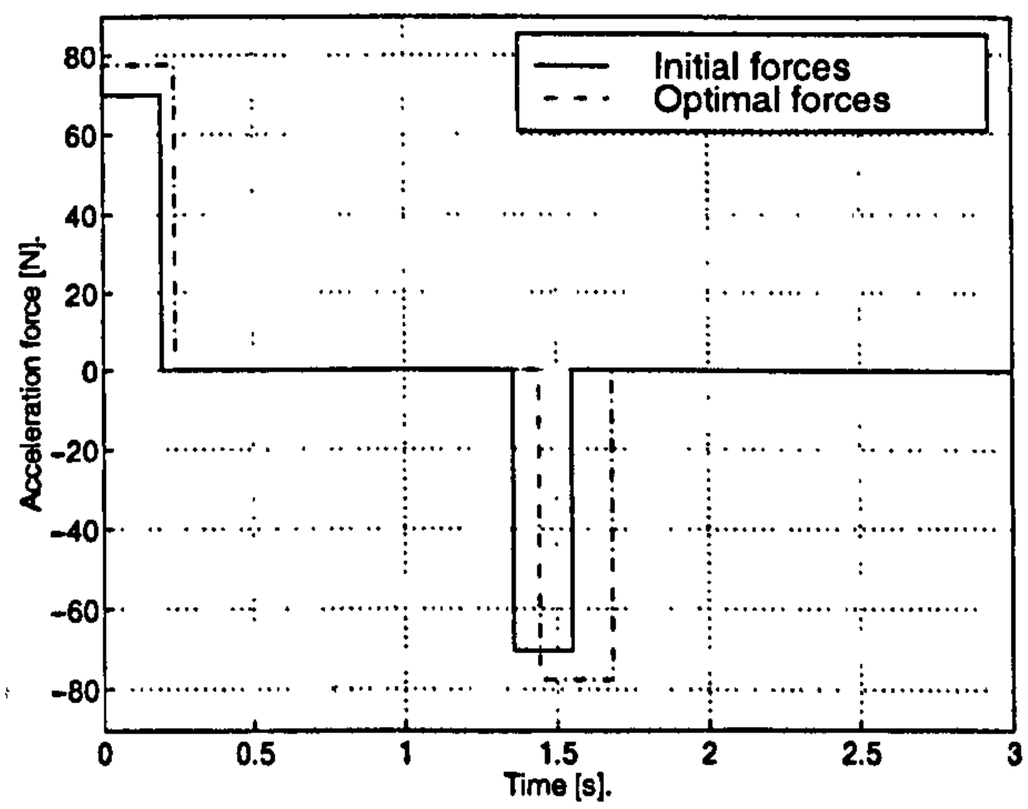


Figure 6.11: Acceleration of the cube.

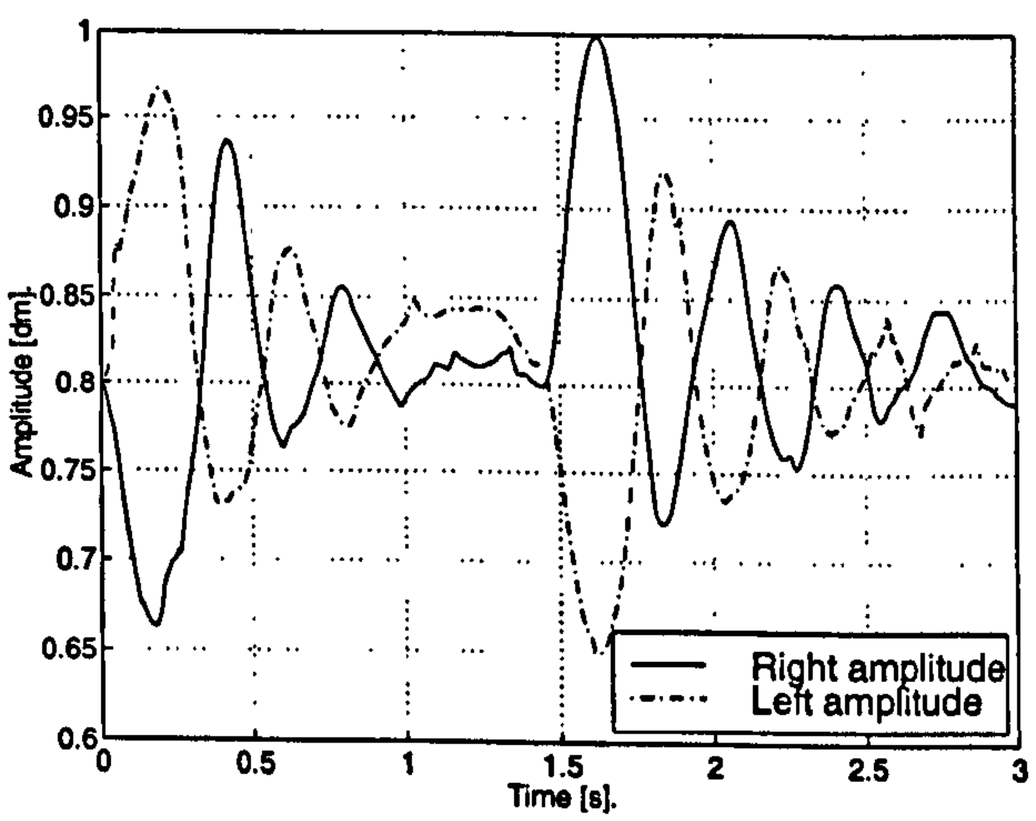


Figure 6.12: Amplitudes of the fluid.

Figure(6.11) shows the initial and optimal acceleration force profile. This time the initial acceleration force profile has been coarsely adapted to a parameter setting which would result in a motion able to move the cart nearly the desired distance. In several other

optimisations the specification of a feasible initial solution has not been necessary. Based on a predefined maximum velocity the whole feasible motion profile has been calculated beforehand based on the desired distance to travel. Due to the fact that this specified velocity has been close to the maximum velocity of the warehouse limitations, the initial parameter setting is very close to the optimal acceleration profile. This time the deviation between initial time and optimal time is only 10%. Figure(6.12) shows the amplitudes of the fluid on the left and the right boundary of the cube. The cart has been accelerated with the optimal acceleration profile shown in Figure(6.11). It is clearly shown that the maximum allowed amplitude has been reached within the deceleration process. A distinct time separation of the fluid motion can be observed, separating the fluid motion during the acceleration from the motion within the deceleration.

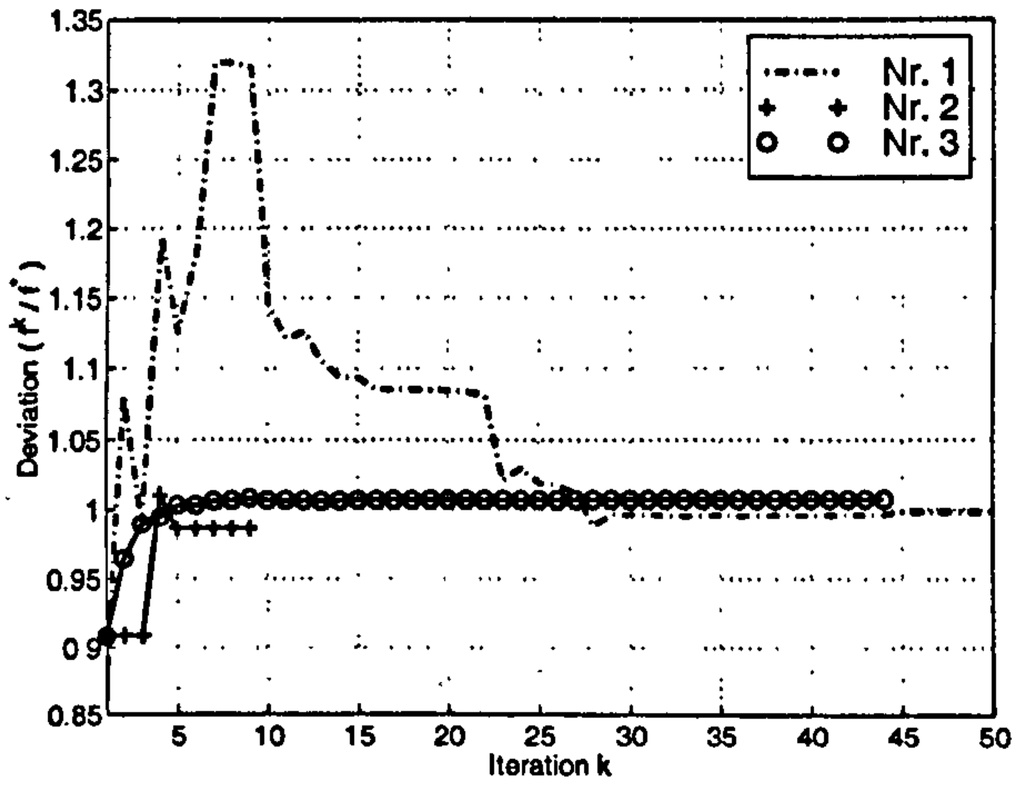


Figure 6.13: Deviation of the solution.

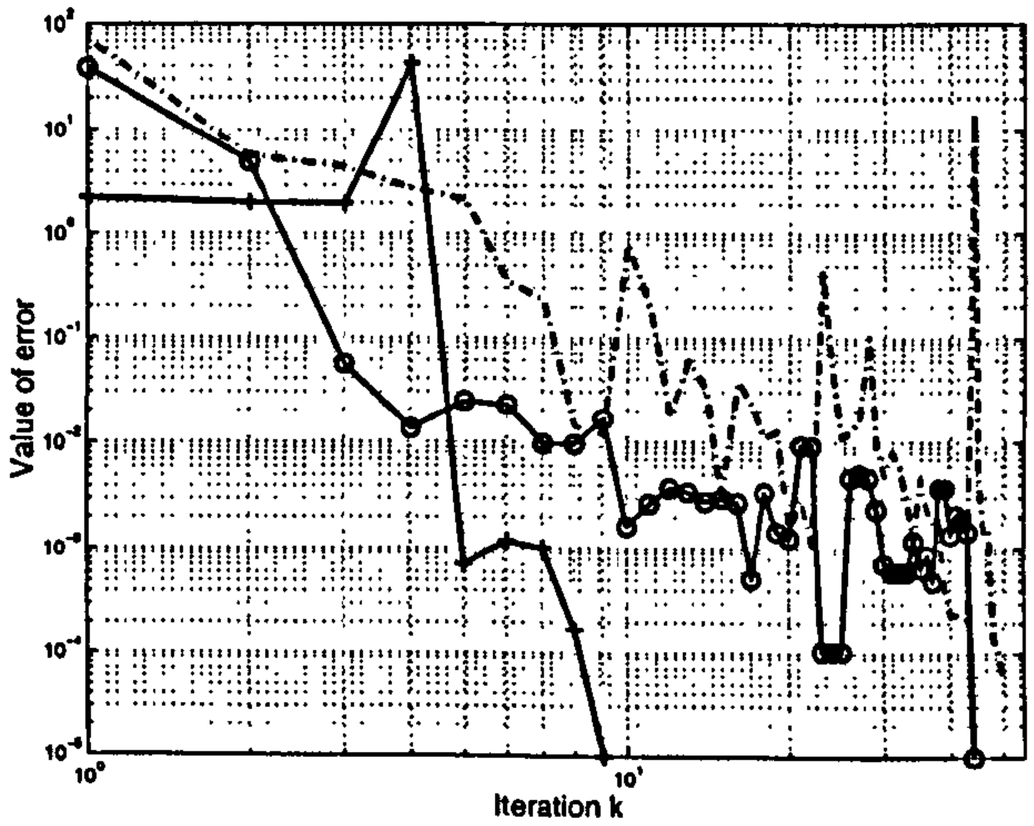


Figure 6.14: Error of the solution.

In Figure(6.13) the time deviation of the solution at iteration level (k) is shown, whereas Figure(6.14) illustrates the error of the solution. The plots numbered 1 and 3 within these Figures indicate that constraints have been active during the optimisation. Within optimisation Nr. 3 the optimisation has started from a point of sloshing and has not been able to move away from this boundary. While following the boundary towards an optimal solution the step size has been reduced dramatically, resulting in a very large number of iterations. Optimisation Nr. 1 has not been able to establish a promising starting point for further iterations until iteration ten. The optimisation pursues a local solution until iteration 22 and has to follow the constraint boundary of sloshing from iteration 30 onwards.

Only the optimisation Nr. 2, which starts also from a point of sloshing, is able to find a feasible point within the solution domain within just four iterations and moves onto the

final optimal solution using just another five iterations.

The comparison of the initial parameter settings does not indicate any significant differences between the optimisations. The optimisations did have the following initial parameter settings:

	Nr. 1	Nr. 2	Nr. 3
Time of acceleration within cart motion	12.5 %	12.5 %	12.5 %
Maximum value of acceleration force	65.00 [N]	70.00 [N]	75.00 [N]
Overall time of forced motion	1.50 [s]	1.55 [s]	1.55 [s]
Number of iterations	50	9	44
Optimal acceleration force	72.08 [N]	77.50 [N]	67.41 [N]
Optimal time	1.70 [s]	1.68 [s]	1.71 [s]

The behaviour of the iterative process seem to be dependent upon the initial parameter setting of the optimisation. This behaviour can be viewed quite precisely within optimisation Nr. 3 within Figure(6.13). This is an example of an optimisation where the constraints of the state and/or the boundaries of the parameters have been reached and the optimisation tries to minimise the violations in order to iterate towards an optimal solution. Within this optimisation the constraint and boundary violations do not change. Here the fluid motion is very close to sloshing from the beginning. This is, due to the fact, that the initial parameter setting of optimisation Nr. 3 has the maximum value of acceleration compared with the other two optimisation calculations. Within the parameter-variation of the optimisation, the error value of the sloshing violation has the greatest impact. This results in very small steps towards the optimal solution, hence the number of iterations increases dramatically compared to optimisation Nr. 2.

In optimisation number 1 some other additional effects have taken place. After iteration 28 the optimisation has reached a parameter set which fulfils the general goals of distance to travel, velocity at final destination and region of optimality, but still suffers of small sloshing within the fluid. This behaviour slowly reduces from iterations 29 to 50. Before that, the optimisation is searching in various directions and is able to find solution regions with smaller error but increased time to travel (see iteration 2, 4 and 7). While the first two regions (iteration 2,3 and 4,5) do not seem to be good starting points for further improvements, the third one (iteration 7 onwards) is able to start a continuous improvement towards the optimal solution. Within the error plot of Figure(6.14), optimisation Nr. 1 performs quite a number of large jumps within the error of the optimisation. These jumps

correspond to the change of the error specification within the optimisation. It is possible that within one iteration, the deviation within the distance to travel is the major error and within the next iteration the velocity at the final time is the major error. These changes can result in jumps within the global error plot of the optimisation.

6.2.3 Variation of fluid viscosity

For the problem of sloshing fluid motion in an open topped container, viscosity is the most important property of the fluid. This property influences the motion of the fluid in terms of sloshing amplitude, sloshing frequency and delay of the fluid.

Again, only motion in horizontal direction has been studied, due to the fact that this motion applies the necessary forces on the fluid to allow the fluid to slosh. Similar to the previous section the fluid height has been fixed to 80 [mm] to guarantee similarity between simulated model and reality. In general the following setting is used for the optimal control calculation:

Dimension of the cube (L×H [×D])	100 [mm] × 100 [mm] × 100 [mm]
Initial surface height	80 [mm]
Domain decomposition	30 × 26 cells
Fluid properties	$\varrho = 998.2\ [kg/m^3]$,
	$\eta = 0.25 - 4 \cdot \eta_{water}\ [kg/(ms)]$
Distance to travel	90 [mm]
Parameter interpolation mode	straight-line (Step function)
Initial parameter set	Trapezoidal, time of acceleration equal to time of deceleration, steady velocity otherwise, segmentation 1/3 acceleration, 1/3 steady velocity, 1/3 deceleration

Again, due to the fact that the calculated parameter sets do, in general, only result in local minimal solutions, several SQP calculations have been started with different initial parameter settings. The obtained results do differ in the amount of iterations, in the amount of function evaluations and in the value of the actual optimal solution.

For practical application the best result, hence the calculation with absolute minimal time would be used. For a general overview on the performance of the optimisation code it is desirable to calculate mean values for those characteristic values. Poor results in terms of

Viscosity $\%of\eta_{water}$	25	50	100	200	400
Number of iterations	13.80	13.75	11.00	12.67	15.25
Number of function evaluations	75.40	85.00	56.33	73.80	81.33
Single motion calculation time	34.30	69.62	169.95	363.41	584.23
Optimal final acceleration time	0.4860	0.4847	0.4825	0.4665	0.4438

Table 6.3: Calculation results of the viscosity variation.

optimal solution have been eliminated and do not count within the mean values presented in Table (6.3).

The mean value of the single motion calculation time is perturbed by influences which alter the accurate result. To generate this value the overall time of a single optimisation run has been stored and divided by the number of single model simulations. A mean value has been calculated based on several different optimisation runs. Hence the following influences can be distinguished:

- The calculation time of the optimisation procedure to generate new parameters is not eliminated.
- Even when the calculations have been performed on one single workstation, the additional calculation load of the CPU is not isolated. Hence it is not assured that the optimisation runs have all been performed on the same computational resources.

The calculation time of the altered parameters within the optimisation routine is very small compared to the calculation time of the fluid motion. Additionally, within every simulation, the same amount of those auxiliary calculations must be added to the simulation time of the fluid motion.

The variation of the computation load on the CPU has been compensated by calculating mean values based on several optimisations. These optimisations have been performed in the middle of the night (no other users on the workstation) as well as in times where students used the workstation for other calculations. Therefore the results have been calculated using a balanced work load distribution of the CPU.

The reduction of the optimal forced motion time (optimal final acceleration time in Table (6.3) with increasing viscosity coheres with the physical understanding that a lethargic fluid can be accelerated more severely than a fluid with lower viscosity.

In addition to the values generated within Table (6.3) the transient amplitudes of the fluid on the left and the right boundary of the cube have been generated. These variations can be seen in Figures(6.15–6.19). In each plot the point of maximum amplitude is clearly visible. Due to the very low viscosity of the fluid, the surface of the fluid in Figure(6.15) reaches the upper limit of the container on both sides of the cube.

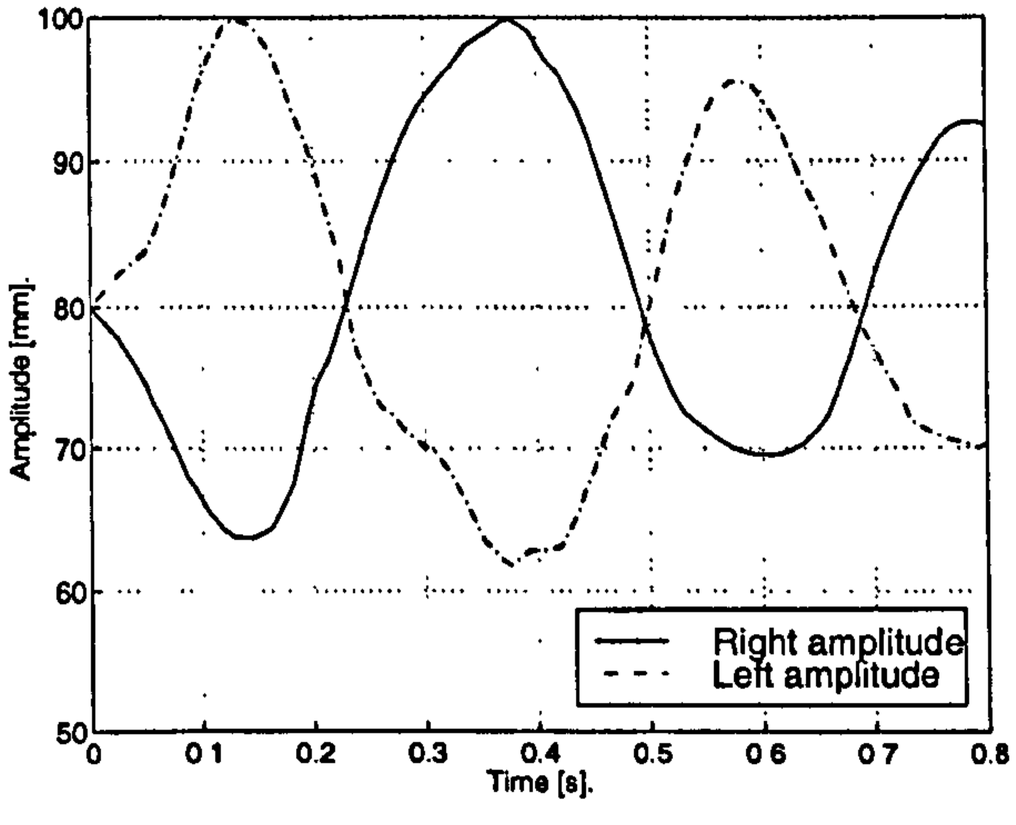


Figure 6.15: Amplitudes, 25% η_{water}.

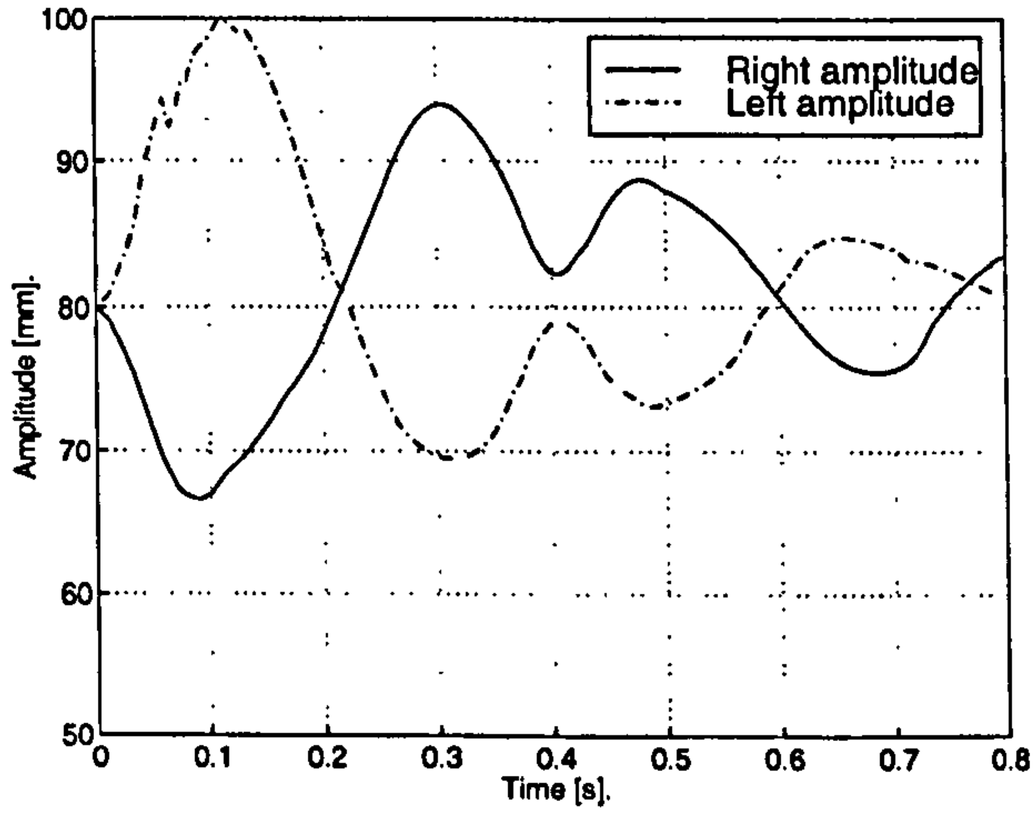


Figure 6.16: Amplitudes, 50% η_{water}.

The motion of the fluid with dynamic viscosities of $\eta = (50/100)\%\eta_{water}$ [kg/(ms)] (Figures(6.16–6.17)) shows that in these cases the deceleration of the cart takes place at a time when the motion of the fluid is not critical. Moreover, while the fluid is decelerated, the fluid moves in opposite direction to the acting forces. Subsequently, the acceleration is the limiting part of the motion, because the first sloshing motion of the left surface amplitude limits the optimal acceleration.

On the other hand the motion of the fluid with dynamic viscosity of $\eta = (200/400)\%\eta_{water}$ [kg/(ms)] (see Figures(6.18–6.19)) show that in these cases the deceleration of the cart takes place at a time when the motion of the fluid is critical. Therefore, at the time of deceleration the fluid moves in the same direction as the acting forces. Subsequently, the deceleration is the limiting part of the motion, because the first sloshing motion at the right surface amplitude limits the optimal acceleration profile.

Figure(6.20) has been generated in order to visualise the increase of frequency of the fluid in motion with increasing dynamic viscosity η. The different plots in this figure are based on optimised acceleration profiles which are very close to the mean values of the various viscosity settings stated in Table (6.3).

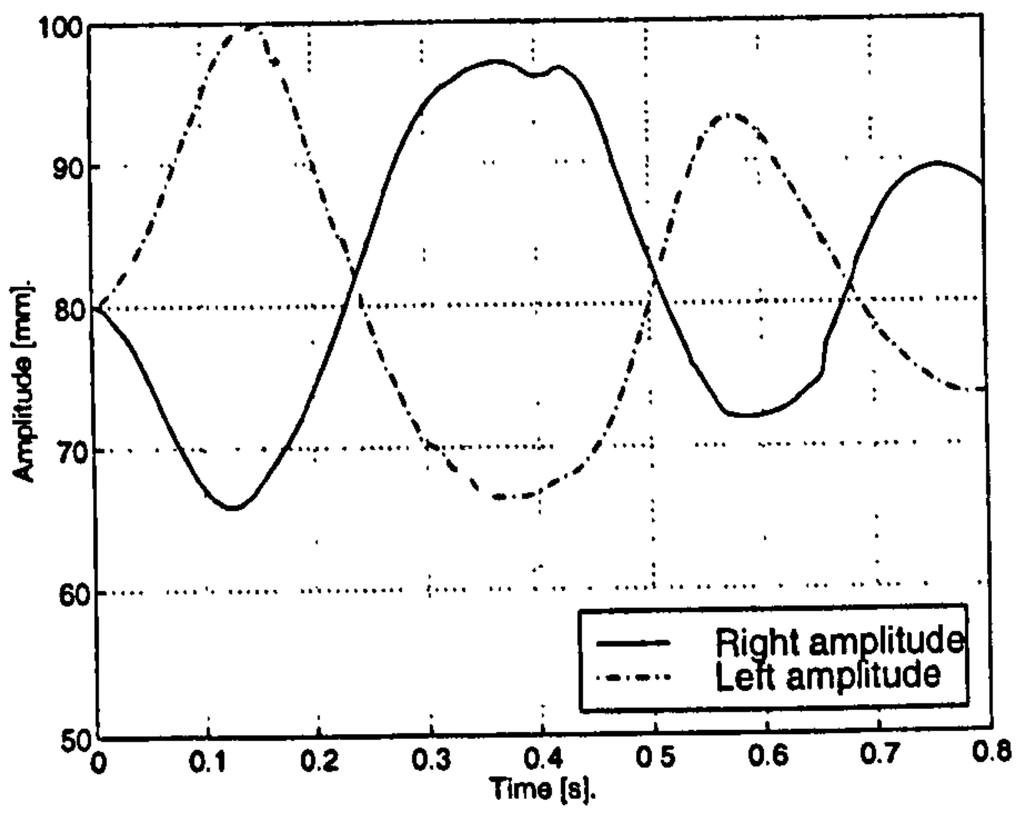


Figure 6.17: Amplitudes, 100% η_{water}.

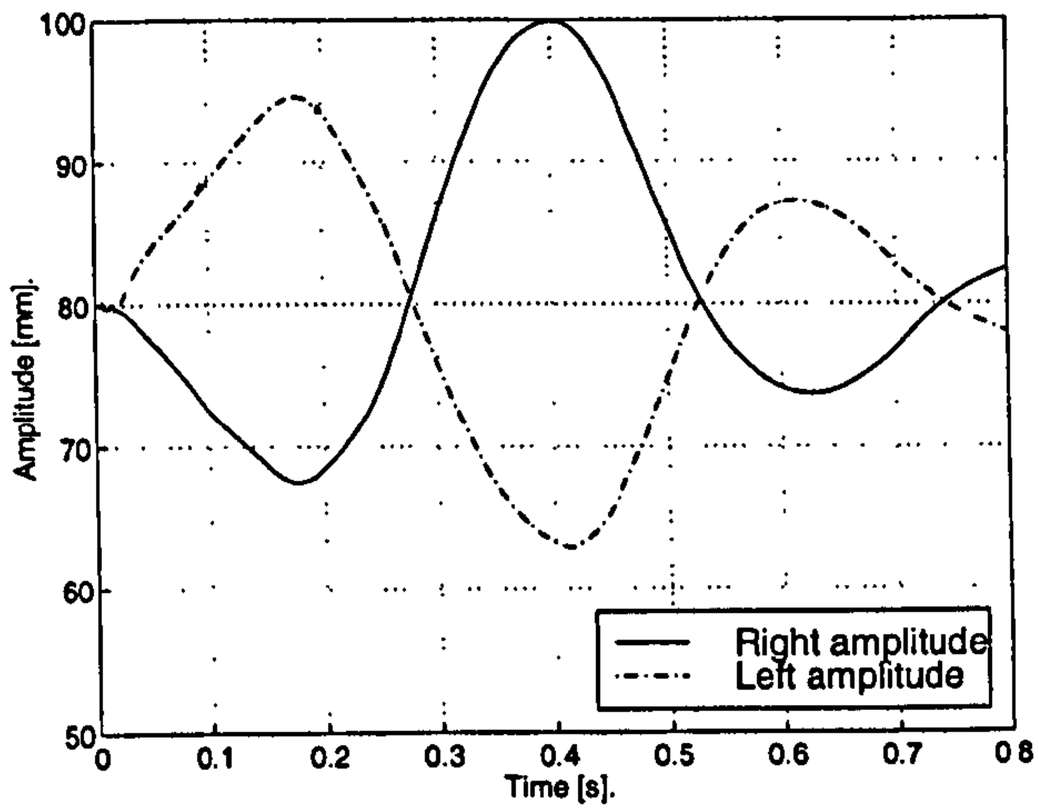


Figure 6.18: Amplitudes, 200% η_{water}.

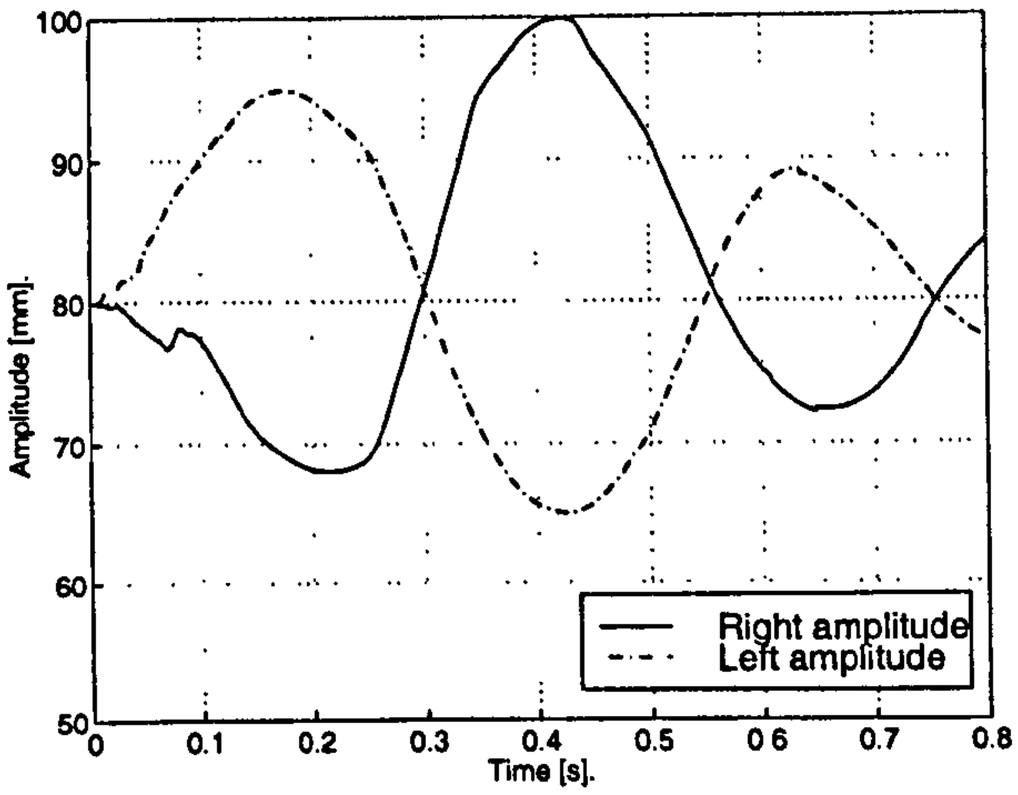


Figure 6.19: Amplitudes, 400% η_{water}.

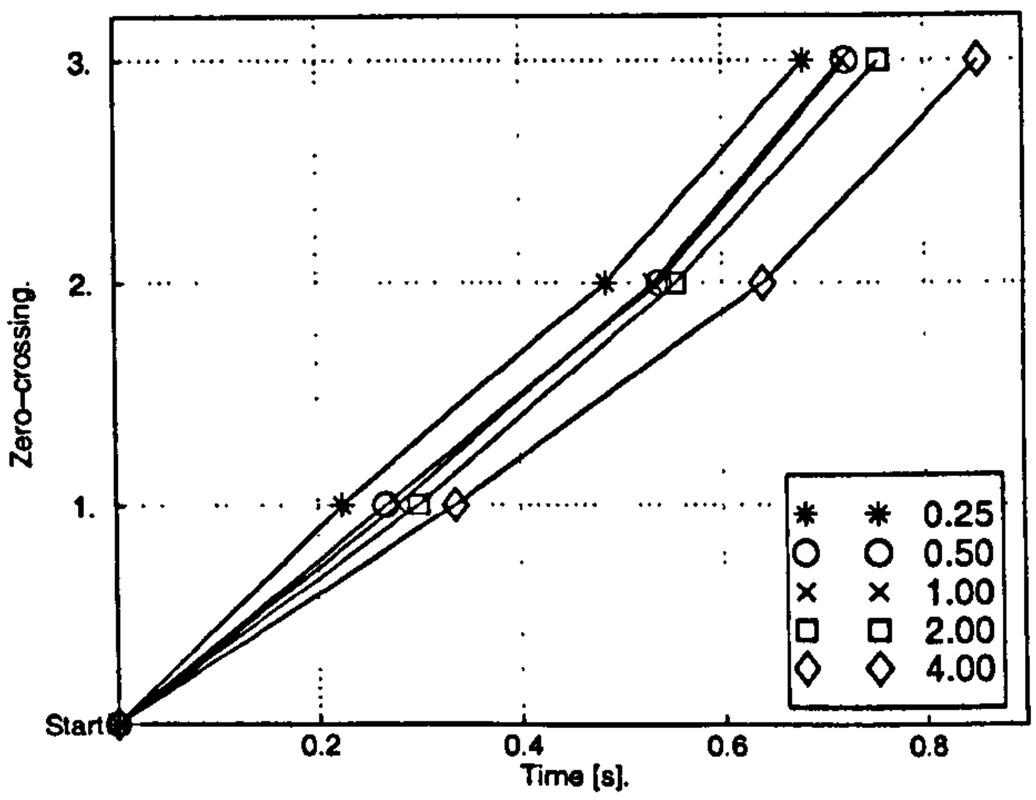


Figure 6.20: Frequency of the motion.

The main feature to be observed is the noticeable increase of frequency with increasing η (it takes longer to cross the zero line for the first, second and third time), even if the optimal acceleration time is decreasing. This also coheres with the physical understanding that a lethargic fluid does not react as quickly as a fluid with lower viscosity. Therefore the frequency of the sloshing motion is reduced.

6.2.4 Shape-variation of optimal acceleration profile

Within this section the focus is on the variation of the shape of the parameter profile, hence the acceleration of the motor of the warehouse plant. In general this profile is defined on

specific breakpoints. The control function is interpolated in the time domain inbetween these breakpoints (see approximation in Figure(5.3)).

Firstly, the focus on this kind of variation will be justified. Secondly, some alternatives will be given, the investigation will be based on. These are:

- Step interpolation.
- Straight line interpolation.
- Polynomial interpolation.
- Spline interpolation.
- Exponential spline interpolation.

Not all of these alternatives have been used for the optimal control calculation. The reason for the selection will be given.

In the third part, the actual achieved results are illustrated to end up the investigation on shape variation with some concluding remarks.

6.2.4.1 Justification for the study of shape variation

Up to now, all the theoretical calculations have been focused on the geometrical and physical properties of the fluid and the container. For this purpose a very simple and harsh acceleration profile has been used. This profile has been used because of its easy applicability to the experimental servo rig and to reduce and simplify the evaluation and analysis of the results. Another reason for the application of this coarse profile is its simple specification. It can be described with only three parameters:

1. Acceleration/deceleration maximum amplitude.
2. Time segment of acceleration/deceleration.
3. Overall time of the forced motion.

As a result, the minimal number of function evaluations per iteration is four. One computation of a basic solution and one variation of each parameter.

In order to incorporate all possibilities of optimality the shape of the acceleration profile must be free to be optimised as well. This must also be done, because the infinite jerk of the

cart, when using a step input, is not acceptable for practical application (see Norton [79]). In Figure(6.21) some general transfer functions of mechanisms, machines or motor controls are illustrated. In order to maximise system performance it is desirable to drive the system in a critically damped way. Especially in the case of transporting an open topped fluid filled container any oscillating behaviour in the input motion of the cart would result in severe sloshing of the fluid. In order to optimise system performance Norton [79] investigated various cam acceleration functions as shown in Figure(6.22). System performance has been classified based on the evaluated maximum velocity, maximum acceleration and especially maximum jerk.

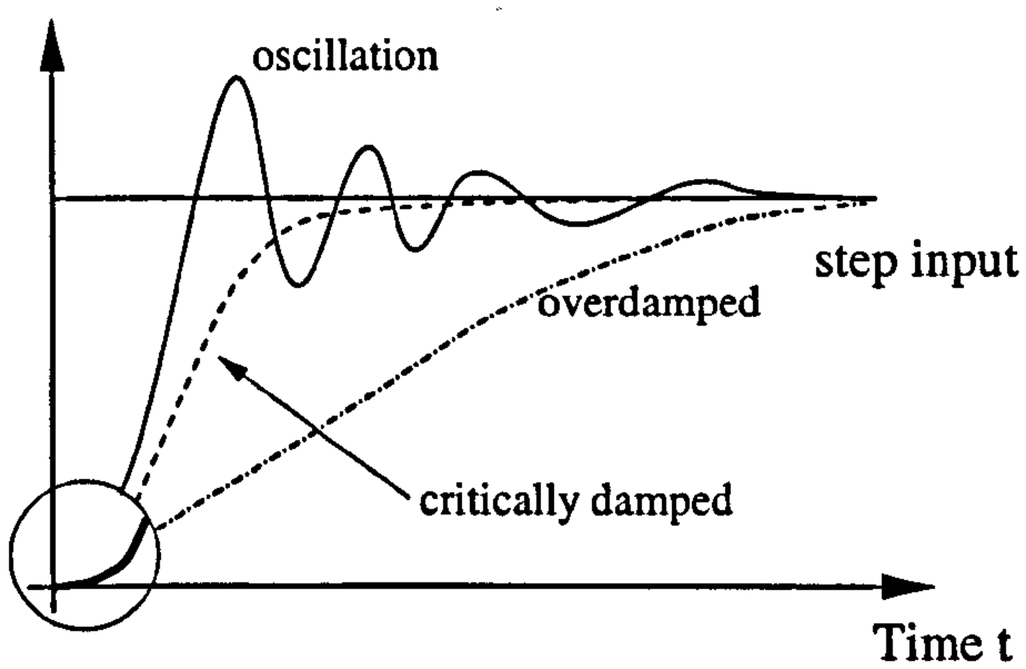


Figure 6.21: General transfer functions.

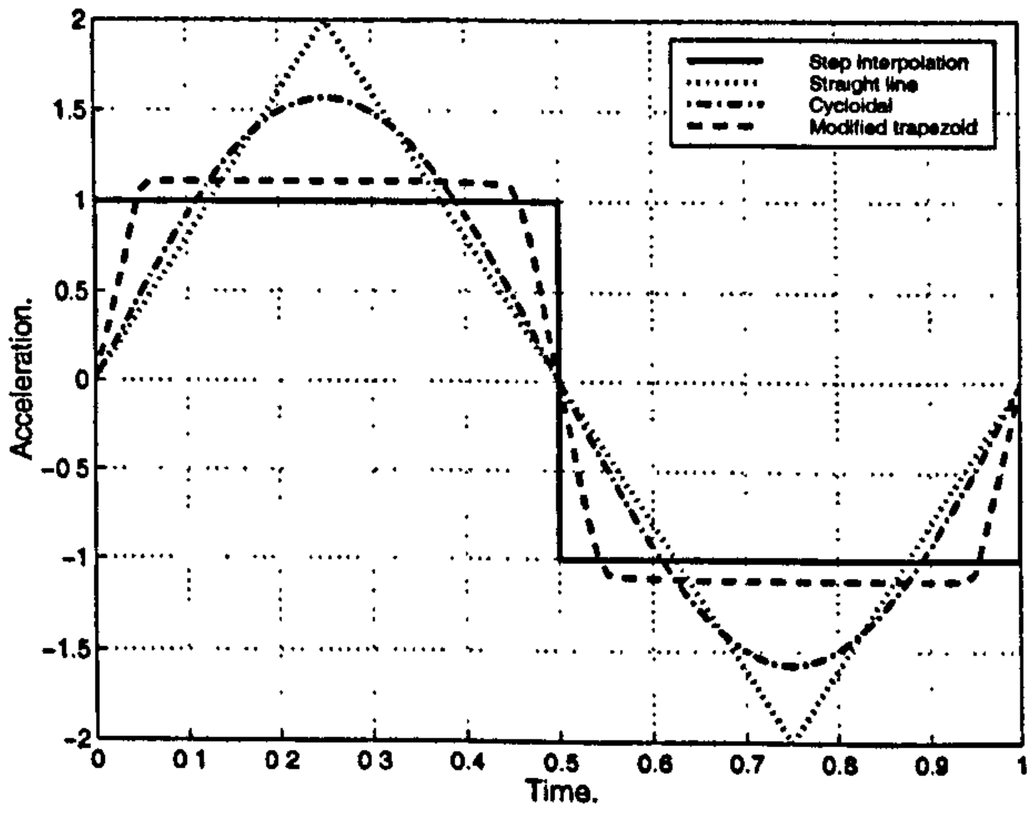


Figure 6.22: Comparison of cam acceleration functions.

This classification outlines that there is no best acceleration procedure. But there is always a most useful and applicable acceleration profile for one specific task. For instance Norton [79] illustrates that a modified sine curve is usually chosen in cam designs in which the follower mass is very large. For the specific problem of fluid motion in an open topped fluid filled container, the modified trapezoidal profile is assumed to be the most appropriate. This is due to the fact that this profile promises in comparison to other profiles, low values of acceleration with a little bit more severe jerk. The low value of acceleration is necessary for the reduction of the liquid sloshing. The slightly rough jerk values are within the limits due to the fact that the cart is only performing low speed motion.

6.2.4.2 Acceleration shapes studied within this research

In particular the following interpolation modes have been investigated.

- Step input (as used previously).
- Linear, straight line interpolation.
- Exponential spline interpolation.

The exponential spline interpolation is assumed to be able to represent an acceleration profile based on the modified trapezoidal function, because this interpolation represents a spline under tension. This spline under tension is able to eliminate oscillating behaviour of a traditional spline or polynomial interpolation.

Within the optimisation, the SQP-algorithm tries to reduce the sloshing motion of the fluid, hence the objective of the code is to reduce the acting acceleration while increasing the speed of the motion. The parameters which are the support-points of the control approximation (see Section(5.4.2)) are tuned within the given control boundaries. When using exponential spline interpolation with fixed end conditions (see comparison in Figures(6.23, 6.24))

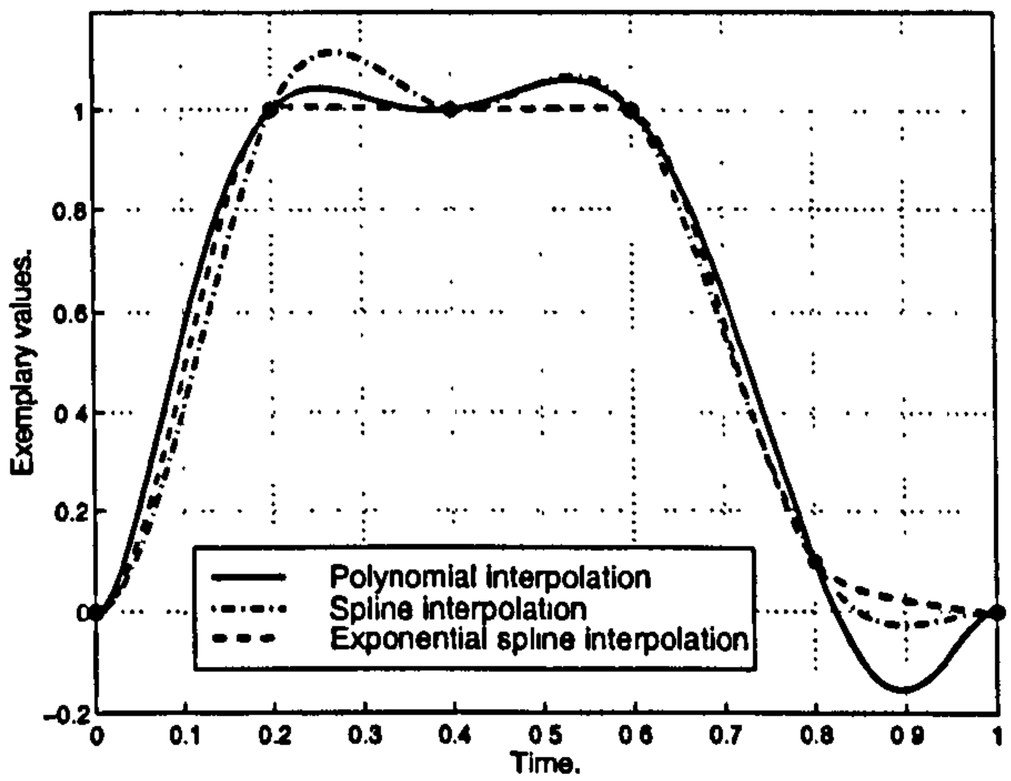


Figure 6.23: Interpolation modes.

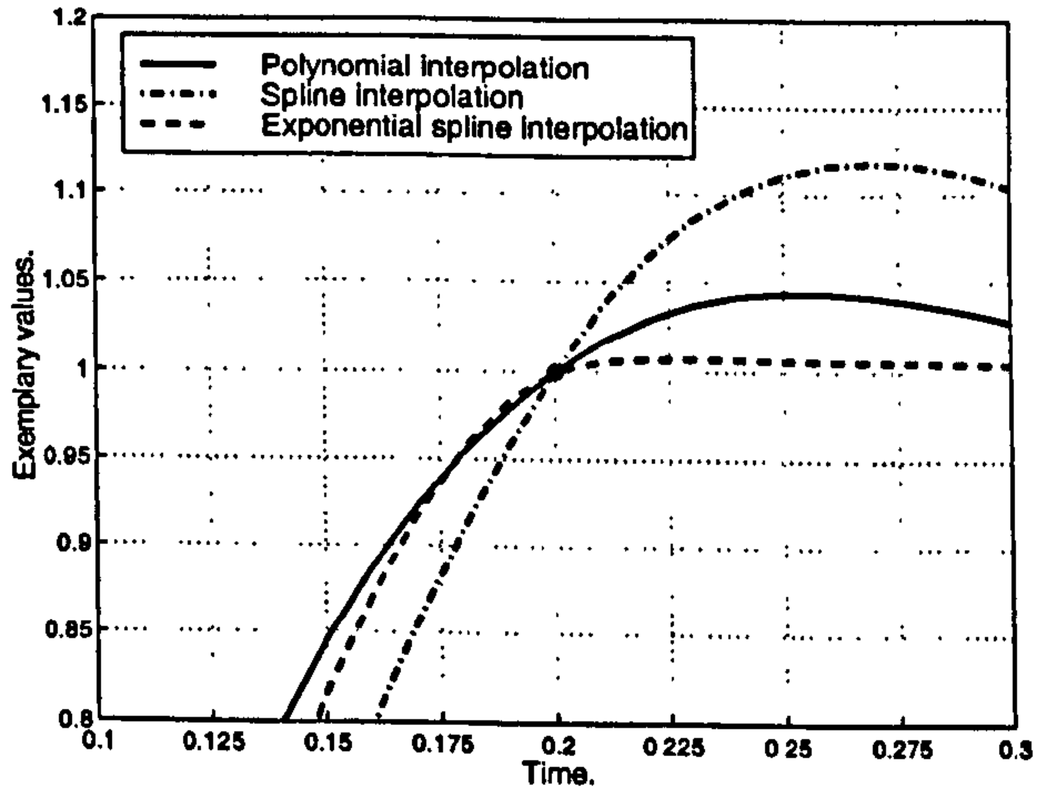


Figure 6.24: Enlargement.

the predefined points will be interpolated in a way that the boundary which has been reached in the points with time $t_i = (0.2/0.4/0.6)$ will only be minor exceeded in the intervals between these points. Additionally, the severe oscillation in the last time interval can be reduced dramatically. These characteristics of an exponential spline or spline under

tension do correspond to the general requirements of the modified trapezoidal acceleration profile proposed in Figure(6.22).

6.2.4.3 Results for the variation of acceleration shapes

The main problem within the comparison of the results of the optimisation of different acceleration profile shapes is based on the practical applicability of these profiles. An acceleration profile in reality is always n-times continuously differentiable. Therefore it is clear that a step input or linear interpolated input, respectively is not able of modelling an acceleration profile of reality.

Within this research program, the plant consist of the motor dynamics and the subsequent mechanical system of the warehouse. The dynamics of the controller, especially the closed loop feedback of the control system has not been modelled. Therefore, the acceleration profile supplied by the controller must be compared.

In order to compare the different acceleration profiles, the input must be pre-computed with a transfer function of the control system. The resulting inputs of the acceleration profiles for the plant are then comparable among each other. In particular a second order system has been used to simulate the dynamics of the controller system.

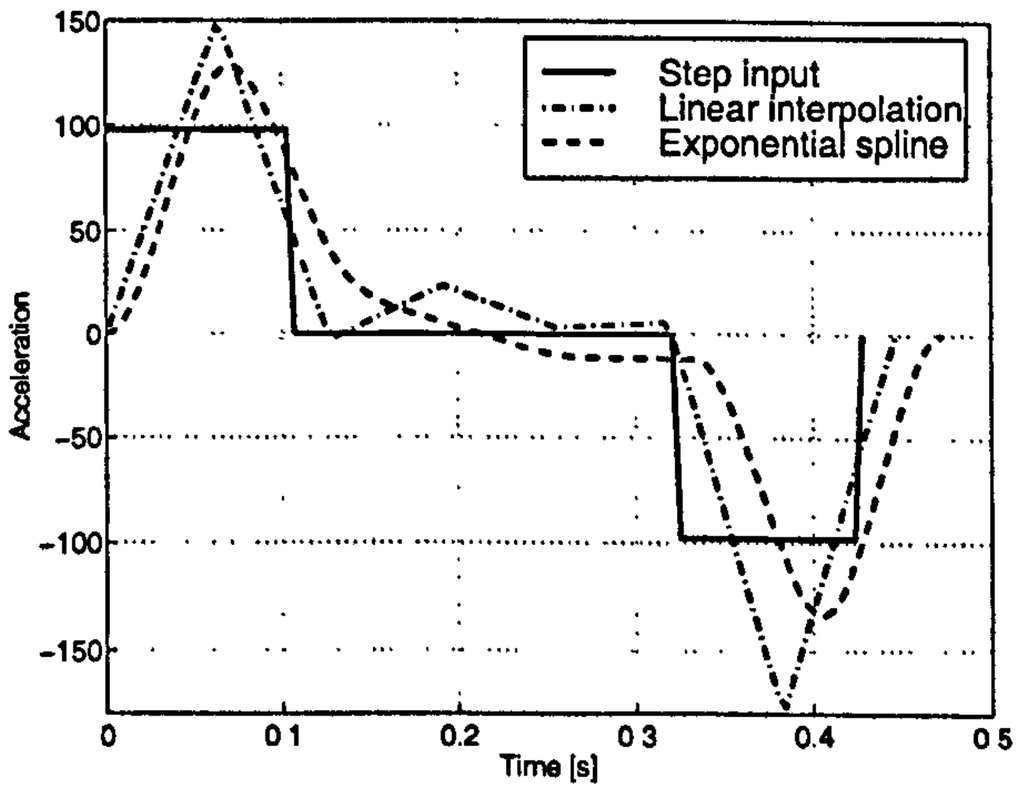


Figure 6.25: Optimal acceleration profile as calculated by SQP-code.

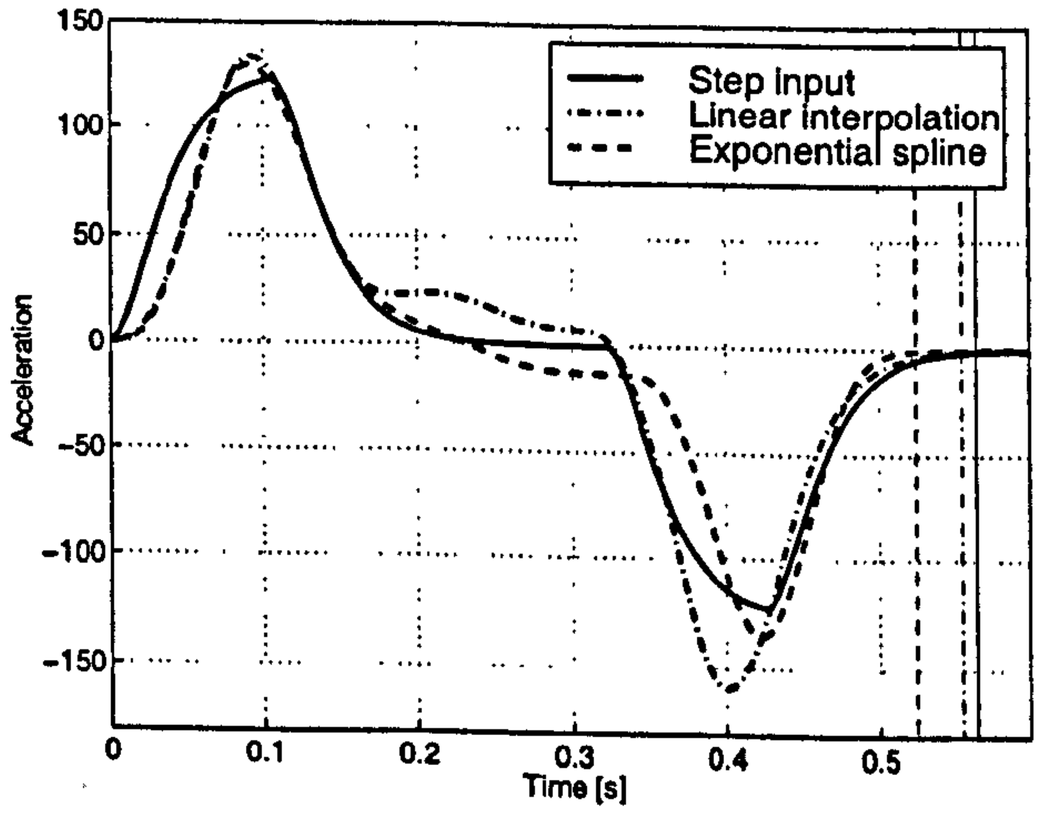


Figure 6.26: Optimal acceleration profile, altered by control system.

Figure(6.25) is based on the results of several computations of the SQP optimal control programme. The following settings for the optimisations have been used:

Dimension of the cube (L×H [×D])	100 [mm] × 100 [mm] × 100 [mm]
Initial surface height	80 [mm]
Domain decomposition	30 × 26 cells
Fluid properties	$\varrho = 998.2\ [kg/m^3],\ \eta = 1.002\ [kg/(ms)]$
Distance to travel	90 [mm]
Parameter interpolation mode	1.) Simplified step function 2.) Linear interpolation 3.) Exponential spline interpolation

Opposed to former calculations on the variation of cube dimensions, distance to travel and variation of viscosity, within the variation of interpolation mode only the best result of all optimisation results have been compared. This has been necessary to be able to compare a specific profile of the different interpolation schemes. A mean acceleration profile of several optimal acceleration functions is assumed to be not representative for the specific interpolation mode.

It is also clear that the comparison is not based on global optimal solutions. But based on the various calculations which have been performed, the three optimal control schemes must be very close to the global optimal solutions.

In Figure(6.25) the results of the optimal control calculations is given. It is clearly visible that the step interpolated case is quicker than the linear interpolated case, which is quicker than the exponential spline interpolated case. This is due to the fact that in the step and linear interpolated cases it is possible to increase the acceleration in the beginning more rapidly then in the exponential spline interpolated parameter profile. The exponential spline interpolated parameter profile has zero gradient of the acceleration boundary end conditions (for illustration see Figure(6.21)).

Figure(6.26) shows the optimal acceleration profiles of Figure(6.25) transformed with the second order transfer function of the control system. In this plot the input functions are all shown in a realistic shape for the motor input. This is clarified by the zero gradient end conditions of the acceleration boundary at all three profiles. In this figure, the final time of motion is shown with the vertical lines on the right side of the plot. These lines specify the time when the value of the acceleration of the corresponding interpolation mode stays smaller then (1). In this figure the expected behaviour of the dynamics of the fluid and the mechanical system according to the acceleration profile of the motor input can be observed. The exponential spline interpolation results in the most realistic interpolation

compared to the step and linear interpolation. Therefore, the actual motion of the cart can be performed the fastest using exponential spline interpolation. Using linear interpolation, the actual motion is slower, and the motion with the step input interpolation is the slowest.

6.3 Practical experiments using the small scale model rig

In this section a specific optimisation will be described for application on the small scale model rig. Motion behaviour of the fluid on the rig has been observed and compared with the mathematically calculated results. Main emphasis is on the sloshing behaviour.

6.3.1 General description of the rig

The rig, illustrated in Section (1.2.1) is capable of two dimensional movements in an upright plane. This motion domain is identical to the motion domain of a real warehouse if the minor motion into the storage place can be neglected.

The servo motors have not been specially chosen or designed for this application, but their load and accuracy capabilities have been investigated by Kassim[59]. The servo motors have the following specifications:

Stall torque	2.10	[Nm]
Peak torque	12.0	[Nm]
Maximum speed	2500	[rpm]
Radial weight limit	18.0	[kg]
Axial weight limit	9.0	[kg]

The load to be placed on the servo is well within these limits. The influences of the control system, which are not modelled within the mechanical system are one reason for the difficulties of predicting the motion of the fluid within the system, because the command acceleration is different to the actual cart acceleration.

To focus especially on the inertia influence of the fluid and its weight, the cube and its fixing have been designed to be stiff but light.

Encoder feedback provides the servo with its accuracy. Encoders used on the servo motors have resolution of one thousand lines per revolution with a quadrature output signal. With an encoder count of four thousand pulses per revolution (hence, a resolution of 360/4000 = 0.09 degree per count), this gives a minimum position resolution in the trajectory of 0.0225[mm], which is adequate for the required assembly accuracy.

6.3.2 Verification of the optimisation results

In order to achieve quick and good results with the optimisation, experiments have been undertaken to generate initial values for the acceleration parameters. These experiments have been undertaken with maximum possible velocity. The acceleration has been increased until the water started to slosh over the edge of the cube.

In general, the sequential quadratic programming optimisation, applied to a non-convex problem can only result in local optimal solutions. This is the reason why every single optimisation, started with small perturbations in the initial parameter set, resulted in a different set of optimal parameters.

The following settings have been used within the optimisation:

Dimension of the cube (L×H [×D])	100 [mm] × 100 [mm] × 100 [mm]
Initial surface height	80 [mm]
Domain decomposition	30 × 26 cells
Fluid properties	$\varrho = 998.2\ [kg/m^3]$, $\eta = 1.002\ [kg/(ms)]$
Distance to travel	90 [mm]
Parameter interpolation mode	straight-line (Step function)
Initial parameter set	Trapezoidal, segmentation 1/3 acceleration, 1/3 steady velocity, 1/3 deceleration final time: 0.55±0.05 [s] acceleration: 1.32/1.58 [m/s²]

With this data, six different local optimal solution have been calculated. MATLAB indicated that all of them reached an optimal solution. The final time for the command acceleration varies in these optimal solutions between 0.460 [s] and 0.552 [s]. The best three of these solutions have been transformed into values for the command velocity and command acceleration of the HTCL-1100 controller card of the model rig.

The motions have been performed on the model rig and showed that in all of these cases the fluid does not slosh over the upper limit of the cube. The fluid only came very close to the sloshing limit. Additionally the time for the motion has been reduced, even compared to the former experiments[7]. This is due to the fact, that in these experiments only the acceleration has been tuned, while the velocity has been set to its maximum value.

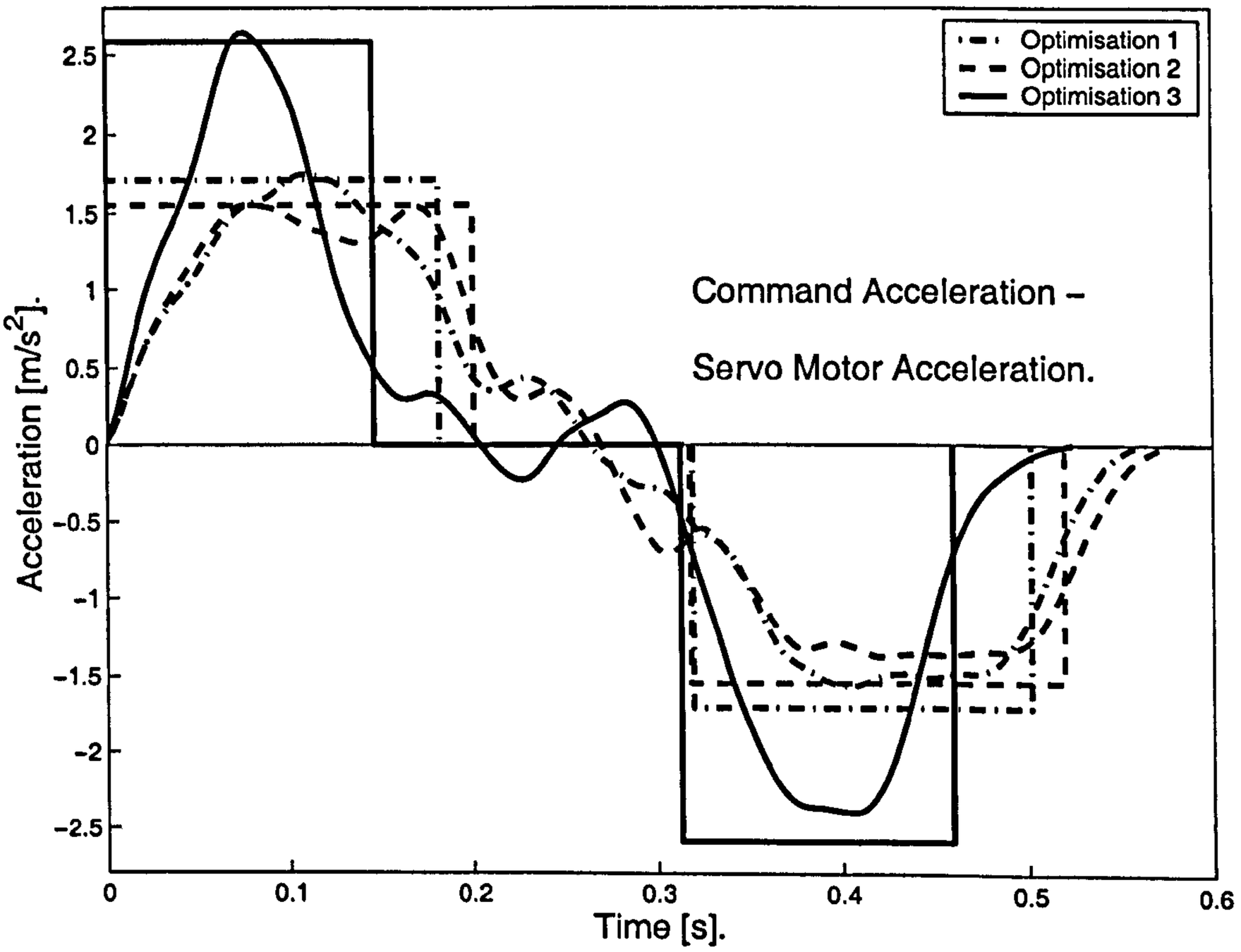


Figure 6.27: Acceleration comparison.

In Figure(6.27) the three profiles are shown. Firstly in terms of the command acceleration and secondly with the time-position information given by the encoder of the servo motor.

It has been shown that the modelling and simulation of the small scale model rig can be used to calculate the fluid and cart motion. These mathematical results are according to the experimental results of the actual servo driven cube. This has been documented with a video record. It has also been documented that a small change within the acceleration of the optimal acceleration profile of the cube does result in a sloshing behaviour of the fluid.

Another experimental verification has been undertaken to illustrate the influence of velocity and acceleration within the motion. Again three different motions have been calculated and

[7]Increasing the acceleration until the fluid starts to slosh.

verified upon the small scale model rig.

1. Firstly, a slow motion has been chosen. This motion can be stated as based on an engineers knowledge and experience. This motion is based on a maximum acceleration of 10% of natural gravity. This approach seems to be safe enough that the fluid will not slosh more than 25% of its height within the cube of 100[mm] × 100[mm] × 100[mm].

2. Secondly, a quick motion has been chosen. This motion uses the same time base as the optimised motion. This motion is based on a maximum acceleration of 20% of natural gravity. This approach will result in a sloshing behaviour of the fluid.

3. Thirdly, the optimised motion has been calculated. In this motion the optimisation resulted in an acceleration of 25% of natural gravity. This approach does not result in a sloshing behaviour of the fluid.

In Figure(6.28) these three acceleration profiles are shown. Firstly in terms of the command acceleration and secondly with the time-position information given by the encoder of the servo motor. This difference is based on the fact, that the closed loop control of the servo has not been modelled within the mechanical system of the warehouse.

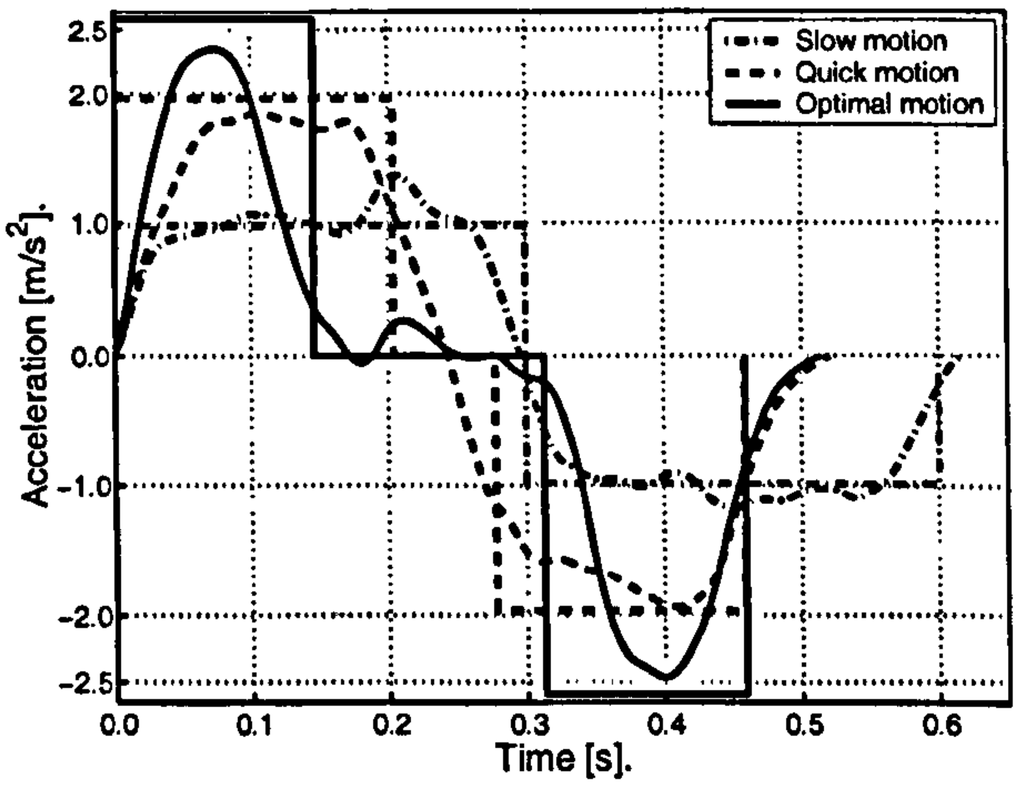


Figure 6.28: Acceleration of the experiment.

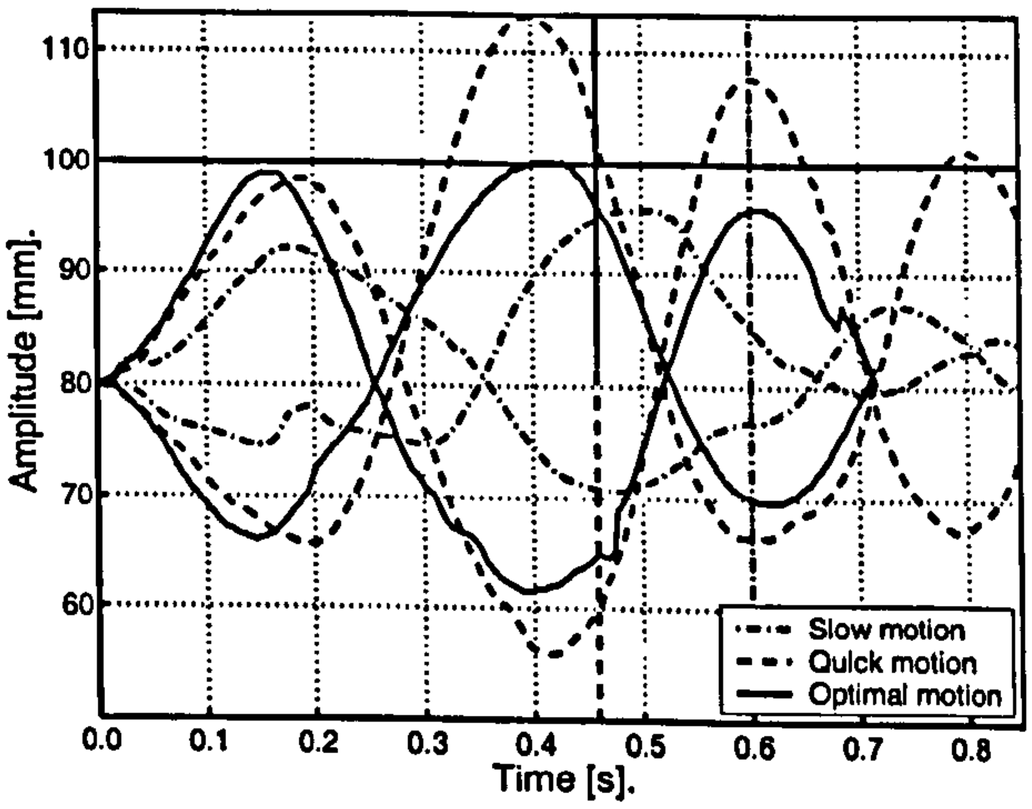


Figure 6.29: Amplitudes of the experiment.

In Figure(6.29) the calculated reaction of the fluid due to the acceleration of the cube are shown.

The obvious features of the different accelerations are:

- Slow motion: only minor sloshing movement, the top of the cube has not been reached,

- Quick motion: major sloshing movement, water is sloshing over the top of the cube on both sides,
- Optimised motion: sloshing movement, but water is only reaching the top of the cube,

These features have been documented with the servo motor encoder and a video record. Using these velocities and accelerations on the small scale model rig it has been possible to document the different motions of the fluid.

6.4 Summary of theoretical results and practical verification

Based on the software implementation of a closed coupled overall model code, incorporating the code NaSt2D and a Runge–Kutta ODE-solver, an optimisation procedure within MATLAB was used to calculate theoretical optimal control for the warehouse problem. Several different parameters like, fluid height, length of container and distance to travel have been varied to simulate the behaviour of the fluid and the mechanical system. Furthermore, the influence of these properties on the shape and characteristics of a corresponding optimal trajectory solution were investigated.

It has been shown that the case of fluid height limits the capabilities of the finite difference code NaSt2D. A prediction with this code is only reliable if the maximum allowable sloshing motion does not exceed 25% of the initial surface height.

Furthermore it has been shown that the optimal trajectory is influenced by the shape of the container. Here the influence of the ratio of length to height of the container can be studied. Results have been obtained for container shapes ranging from slim and tall containers to long and flat ones. In particular the influence on the minimal time and maximum acceleration were investigated.

Within the variation of distance to travel, optimal results have been achieved for distances ranging from 45 [mm] to 720 [mm]. Different characteristic behaviours of the amplitudes of the fluid on the left and right container wall were distinguished for very short and very long distances to travel. Related to this property, a great influence on the performance of the optimisation code has been documented.

The fluid property viscosity has been varied in a bandwidth of 0.25 ... 4.0 times the viscosity of water. Within these calculations an increase in the single motion calculation time has been observed, while the optimal value for the time to travel decreases. In addition to these characteristic values of the performance of the simulation and optimisation, the layout of the amplitudes on the left and the right wall of the container have been observed and compared. As a result, a frequency of the sloshing motion of the fluid has been calculated. This frequency is decreasing with increasing viscosity.

In order to reflect the properties and essential elements of reality more comprehensively, the shape of the interpolation scheme has been optimised. This included the study of step-, linear interpolated- and exponential spline interpolated input. It has been shown that a parameter profile which is more justified by practical applicability does also result in better solutions then the step input which is not usable in practical applications.

Additionally a small scale model rig has been built to simulate a selection of motions calculated with the SQP-algorithm. The rig has been described and its features in terms of control capabilities were shown. Especially some difficulties with the specification of the velocities and accelerations using the HTCL-1100 were illustrated.

Finally, some motions have been selected to be verified on the small scale model rig. It has been shown that the sloshing behaviour of the fluid is dependent not only on the maximum value of the acceleration but also on the time the acceleration acts on the system. The optimal movement, using even $1/4g$ is not sloshing compared to the motion with $1/5g$.

6.5 Nomenclature

a	acceleration
e	error value
f^k	value of the objective function at iteration level k
f^*	value of the optimal objective function
fr	frequency of the sloshing motion
g	natural gravity
L×H [×D]	length × heigth [× depth] of the container
N	accuracy of the encoder
t	time
t	sampling time in [μsec]
T	value of the sample time register
v	velocity
$x, \vec{x}(x, y, z)$	position vector with cartesian components x, y, z
η	dynamic viscosity
ϱ	density

Chapter 7

Discussion of the optimal trajectory results

The aim of this chapter is to discuss, explain and illustrate the results presented in Chapter (6). In particular the findings of the following investigations will be studied and discussed.

- Investigations on the influence of the dimension ratio of the container onto the optimal fluid motion and the optimisation.
- Investigations on the influence of the distance to travel on the optimal motion of the fluid and the performance of the optimisation.
- Investigations on the correlation of change of viscosity of the fluid and change within the optimal fluid motion and the iterative optimisation procedure.
- Investigations on the objectivity of different interpolation shapes for the control and their performance and impact on the optimal fluid motion and the optimisation.

Additionally some findings will be discussed, which spanned over all computation processes within the optimisations. These results are linked with various parameters and properties of the CFD- and the SQP-code. In particular the influences of surface tension and parameters of accuracy like time-step control, number of particles per fluid-cell or penalty function variation are considered.

Where possible, considerations are made in which way the achieved theoretical results can be turned into practical applications.

7.1 Theoretical investigations concerning the simulation

In order to discuss the results presented in Chapter (6) in a rational and objective way, it is important to restate the main targets and intentions of this work. The aim was to develop a novel model for the representation of a warehouse environment in order to be able to optimise the motion of the transport facility in a time optimal way. Due to certain limitations like limited computational resources one of the objectives was the research and development of efficient, accurate but also fast models for the fluid and the mechanical system.

It has been shown in Chapter (3.8) that the necessary reduction of accuracy in order to achieve the goal of faster calculation times does not affect the accuracy of the characteristic values of the system. These characteristic values, the maximum sloshing amplitudes at the boundaries of the container, have not been affected within the reduction of the model accuracy. To illustrate this behaviour of the finite difference code NaSt2D various experiments were undertaken to classify and verify its accuracy.

Furthermore, the accuracy and representation of the mechanical system has been investigated. Its model is based on very standard and generally applied approaches, which were revised several times (see Chapter (2.2)), these codes have only been verified in terms of reality reproduction. Strong correlation to other results[10][72] were found.

Focus was on the model and simulation of the fluid-structure interaction. Based on the known approaches of closed and weakly coupled interaction, computer and accuracy performance features were studied. As a result, the strongly coupled approach has been selected (see Chapter (4.4)) to be used within the overall model of the fluid and the mechanical system. This decision has been based on the advantages of higher accuracy and faster calculation of the selected closed coupled approach.

Another study within the preparation of the optimal control calculations and their experimental verification involved the investigation of the capabilities of the small scale servo rig

and its limitations which must be mapped onto the boundaries of the theoretical calculations. Furthermore the experimental rig was used to set boundaries for the applicability of the CFD-code NASt2D in terms of accurate representation of the physical properties of the fluid.

As a result of these preliminary experiments constraints had to be formulated for the optimisation in order to correlate the results of the optimisation procedure to the practical experiments on the small scale model rig.

It was possible to illustrate that the fluid model is only accurate if the amplitude motion of the fluid is not exceeding 25% of the initial fluid height. This heuristic rule was verified for a quadratic cube of 100 [mm] edge length, according to the one used within the experiments. These findings are based on the absence of modelling additional physical properties, like turbulence, within the finite difference code NASt2D. The experiments on the small scale model rig showed the presence of these properties.

In order to evaluate knowledge about the characteristic behaviour of the fluid in motion, several theoretical calculations of the fluid and the optimisation were performed. First of all some geometrical parameters were varied to study their influence on the shape of the optimal control acceleration profile.

Within the calculation of fluid motion there is an ongoing debate about the influence of the shape of the control volumes on the results of the calculations. In general the most precise results can be achieved if the length of the edges of the control volume all have the same length, hence their ratio is one. It was researched what influence a change in this ratio of the overall cube edge lengths causes within the optimal acceleration shape. For this purpose the length to height ratio of the cube was varied between ($0.5 \leq \text{ratio} \leq 2$). It can be observed that the fluid reacts more slowly and lethargic, the more the ratio of the container dimensions change from a slim upright cube towards a long horizontal one. Additionally the frequency of the fluid motion is decreasing. It has been shown that the acceleration of the cart motion can be increased while making the container slim and tall. On the other hand, the motion must be slowed down if the container is very long and flat.

The small scale model rig limits the absolute distance to travel quite thoroughly (the maximum distance to be driven is 100 [mm]). Therefore some other distances to travel must be investigated to get closer to the dimension properties of a normal large scale warehouse environment. Only the motion perpendicular to the natural gravity was investigated to

study the motion of most severe sloshing motion. Additionally, the shape of the acceleration profile was limited to the simple step input motion used by the servo rig motion controller.

When using the very simple acceleration and deceleration profile the optimisation of the distance to travel can be analysed separated into the acceleration part and the deceleration part. The sloshing motion of the fluid, stimulated by the acceleration does not influence the fluid sloshing within the deceleration. This separation is only possible for large distances to travel. A specific, characteristic distance can be distinguished, starting from which this separation occurs. This specific distance is dependent on the properties of the fluid and the cart (viscosity of the fluid, weight of the cart, etc.). In the experimental cases studied where water is being transported, this travelled distance is approximately 600 [mm].

The degree of freedom in terms of possible sloshing constraint violations is greater the longer the distance to travel becomes. There is a characteristic frequency for the given fluid and container properties in which the fluid moves. This frequency is dependent on the external force impact and has a frequency bandwidth of $0.2\frac{1}{[s]} \leq fr \leq 0.3\frac{1}{[s]}$ using the given fluid and container properties (viscosity of the fluid, dimensions of the cart, etc.). Therefore with a fixed maximum velocity of the cart, a maximum number of sloshing constraint violations can be calculated. This number is greater the longer the distance to travel becomes. It is clear, that an increased number of possible constraint violations results in an increased complexity and level of difficulty of the optimisation procedure. This is an additional reason for the high numbers of iterations within Figure(6.14).

Within the shorter distances to travel, the sloshing motion of the liquid amplitudes is the result of a combined influence of acceleration and deceleration. The separation of the two sloshing motions within the longer distances makes it more difficult for the optimiser to calculate the optimal acceleration profile because the constraint violations are not as closed coupled with the variation of the acceleration profile like in the short distance calculations. That means that a small perturbation within the control profile could result in the reduction of the left amplitude within the acceleration process, and at the same time an increase of the left amplitude at a later time. As a result, the optimiser could not distinguish whether the motion control profile has improved or not.

In order to investigate the most important property of a fluid in an open topped container, the viscosity of the liquid, was varied. This physical property influences rapidly the motion of the fluid in terms of sloshing amplitude, sloshing frequency and delay of the fluid.

The reaction of the fluid was studied for viscosity ranging from 25% to 400% of η_{water}. In general, the calculation of the fluid motion with higher viscosity is more time consuming than with lower viscosity. This is due to difficulties within the pressure correction of the SIMPLE algorithm within NASt2D (see Section (3.5)). Numerical experiments of the simulation showed that these difficulties can not be avoided by variation of the number of mesh-cells, the shape of the cells (ration of the edge lengths) or the time step control. They were overcome by using higher accuracy, especially using more particles in every single fluid cell. It was verified that the calculation time of a single function evaluation is linear coupled with the number of particles per fluid cell. Hence, when the number of particles within one calculation cell is doubled, the calculation time for the overall calculation is doubled as well.

It was confirmed that the frequency of the motion, hence the time intervals between the moments when left and right amplitude do have the same height, is decreasing with increasing viscosity. But it must be drawn to the attention of the reader, that a change of the physical property of viscosity does also possibly change the overall model. Hence it might be possible that higher viscosity requires the implementation of surface tension within the simulation of the fluid. This was not considered due to the low influence of surface tension within the model of liquids like water.

For practical application, fluids with higher dynamic viscosity can be accelerated more severely than fluids with lower viscosity, because the liquid responds slower to the acceleration of the cube. This can be demonstrated easily with water compared to honey. But on the other hand, if the fluid is already sloshing, it is more difficult and costly in terms of energy consumption to reduce the motion of the fluid.

Finally within the investigations of the theoretical calculations, the main input parameters of the simulation and optimisation were studied. These input parameters are the control of the servo motors. They are also called design parameters, because they determine the response and motion of the plant. There are different ways to specify the control parameters. Due to its flexibility, adaptivity, robustness and ability to be tuned easily, an interpolation attempt will be used. In this research work equally spaced base points are specified and interpolated using several different interpolation schemes. This specification is done for the time period ($[0 \ldots 1]$), which will be multiplied with the overall time to be optimised. In order to get an insight into the performance of the mechanical system and the fluid, this interpolation mode was varied and the results studied.

Three different shapes were investigated. They are step interpolation, straight line interpolation and exponential spline interpolation. The results were presented in two stages. Initially the actual results of the simulation based on the optimal trajectories of each case were compared. From these findings the following conclusion can be drawn:

The rougher the interpolation mode is, the better and quicker the motion will be. This would assume that the acceleration can be switched from zero to maximum within an infinite small step in time. Due to the fact, that this motion is not applicable on servo motors in reality a second way was introduced.

Within this second approach, the control system will be modelled as well. This results in the advantage, that the actual motion of the servo motor, based on the three different interpolation modes, can be investigated. Hence the practical implementation will come into focus and not only the theory. For this purpose the controller of the servo motor was modelled with a two degree of freedom dynamical system. The optimal acceleration profiles of the interpolation schemes were transferred from the pre-controller state to the post-controller state. Subsequent the results of the transferred optimal trajectories were compared once more.

This time the results are turned completely. The smoother the interpolation mode is, the better and quicker the motion will be. The final outcome of the study on interpolation shape variation can be summarised as follows:

The optimal motion, hence the motion using minimal time is strongly coupled with the context of practicality. The interpolated shape which is closest to the optimal shape within cam design (see Norton[79]) in terms of inertia, velocity, acceleration and jerk is also the optimal shape in terms of overall speed.

7.2 Theoretical investigations concerning the optimisation

Within the used sequential quadratic programming algorithm of MATLAB some problems occurred which are well known when using numerical optimisation codes.

First of all it became obvious very early that the optimisation gets into difficulty in terms of proceeding towards an optimal solution, if the simulation result of the arbitrary initial parameter set did already result in sloshing of the fluid. Meaning, that the solution of the initial parameter set is not part of the feasible domain. It was necessary to introduce a penalty function which forces the optimisation code to develop a parameter set with feasible solution. This penalty function was varied and several different schemes (fixed value, linear or exponential increase, etc.) were studied. Finally a very simple, linear function was used to increase the value of violation. This approach indicated the best results.

Secondly, the author was able to observe the known critical behaviour of SQP methods to follow the steepest descent towards a boundary and then following the boundary towards an optimal solution. This behaviour is very time consuming in the iterations where the SQP-code follows the boundary, because the step size must be made very small. Additionally even more severe constraint violations of the solution will occur, based on the parameter-variations within the SQP-code.

The plots of iterative improvement showed that the optimal solution can be found more easily if the initial overall acceleration time is longer than the final optimal time, hence the optimisation should be started from a feasible solution. In the case of the overall time, an additional effect comes into play. In the case of longer initial time the main constraint violation will take place preferable in the limitations of final position and velocity and not within the sloshing behaviour of the fluid. In the case of shorter initial acceleration time the limitations of final position and velocity are strongly accompanied by the violation of the sloshing limitation.

It has been shown that the SQP-code is more sensitive on the sloshing error than on the velocity or position error. The additional violations of sloshing aggravate the search for a new, improved set of design parameter.

Additional difficulties within the optimisation were observed while the change in the parameter set leads to jumps within the constraints. Jumps means a switch from one constraint

being the maximum violation to another constraint being the new maximum violation. This results in severe difficulties within the calculation of the new step direction and step size. This behaviour was pictured in large variations within the error plots of the iterative process within the optimisation.

It is clear that the optimal solution of the problem of trajectory optimisation to reduce sloshing in open liquid filled containers lies on the boundary of the feasible domain. This boundary is marked by the sloshing constraint. Therefore the influence of the boundary on the iterative process of the optimisation is very large. Initially, the optimisation tries to reduce the violation of the constraints. In the case that the initial parameter set results in a feasible solution, the only constraints to be reduced are the constraints of final position and final velocity. The reduction of the objective function, in our case the overall time of the movement, is only pursued with minor interest. From the point, when a set of design parameters can be found, which is very close to the fulfilment of these constraints, the main objective of the SQP-algorithm is the optimisation of the objective function. As long as the time of first sloshing can be delayed, the iterative process is proceeding rapidly towards an optimal solution. Hence the main delay within the optimisation is based on the violation of the sloshing constraint.

7.3 Practical verification

It was confirmed that the finite difference code NaSt2D is capable of modelling and simulating free surface motion of water within a cube. This was documented within preliminary investigations using the bed of a milling machine as an accelerator for the cube.

Additionally, based on the small scale model rig, it was shown that the modelling and simulation of this rig can be used to calculate the fluid and cart motion. These mathematical results are according to the experimental results of the actual servo driven cube. A large number of experiments were performed and eventually documented with a video record. It was also documented that a small change within the acceleration of the optimal acceleration profile of the cube does result in a sloshing behaviour of the fluid. For this purpose limit accelerations based on maximum velocities were determined experimental.

Finally a set of acceleration profiles were used to illustrate that the optimal set of initial acceleration parameters in the case of step input for the controller of the servo motor is

a combination of the optimal maximum velocity and maximum acceleration. The optimal motion is not determined strictly by the acceleration of the cube. The author was able to document these calculated findings with a video, when applying the corresponding acceleration profiles to the small scale model rig.

The main limitations within the practical verification were the restricted motion input and the limited distance to travel. Additionally it was not possible to control and drive the servo motors directly, the controller had to be programmed respectively. This controller has a feedback control of the servo motor encoder which was not modelled within the mechanical system of the warehouse.

Chapter 8

Conclusion and recommendation for further work

This chapter is dedicated to a final conclusion on the whole research project. In particular conclusions will be made on the modelling process, the optimisation procedure and the results obtained.

These conclusions will be rounded off with an illustration of the contribution to knowledge elaborated by this research work.

Finally an outlook will be given into suggested improvements and alterations possible for the modelling, simulation and optimisation of the given problem in a more comprehensive way.

8.1 General conclusions of the thesis

In order to draw conclusions on the thesis presented the general goals and aims are illustrated. These aims can be shortly summarised as follows:

1. Development of a simple but accurate model of the fluid motion and the mechanical system.

2. Efficient combination of the two models.

3. Simulation of characteristic values for the warehouse motion and incorporation of the model simulation within an optimisation package.
4. Calculation of optimal trajectories within the optimisation procedure.
5. Verification of the obtained result, using real world application.

The general way in which the proposed goals (1-5) have been fulfilled will be presented.

With the help of the finite difference calculation procedure NaSt2D and additional programming, the free surface fluid motion has been modelled. In order to achieve the goal of accuracy combined with efficiency some simplifications and limitations had to be introduced. Therefore, the results presented in this thesis include laminar, incompressible, free surface motion without modelling turbulence compressibility or surface tension. Due to the module orientated structure of programming there would be no problem to integrate additional modules with these capabilities. The programmed enhancements to this code include a time varying acceleration of the fluid and a reduced calculation time. Furthermore, a sensitivity analysis revealed the particular good applicability for the given problem due to the specific balance of computational time and accuracy of the code.

The mechanical system of the small-scale model rig has been modelled with a standard approach for such systems of ordinary differential equations. The motion of the mechanical system is calculated using a Runge-Kutta method of order 4(5). The accuracy of this solver has been verified using other mathematical calculation platforms such as MATLAB.

A closed-coupled, fluid-structure interaction model for the combination of the fluid motion and mechanical system dynamics was developed. The advantages of such a closed-coupled approach in comparison to a weakly coupled approach were shown. As a result of the combination, the overall model is able to calculate the interaction of the fluid motion and the motion of the mechanical system within every time step of the CFD algorithm. Furthermore, the author was able to show the effects of the assumption, that the motion of the fluid influences the motion of the cart.

With the generated model, the motions of the cart and the fluid were simulated. Calculations were performed especially to determine the influence of the walls of the container on the motion of the fluid. These investigations led to a more exact determination of the free surface at the walls, allowing higher sensitivity in the correlation of the acceleration profile (the input) and the amplitudes of the fluid (the output). This increased accuracy has been very important during the simulation of the overall system.

Based on the need for robustness and efficiency an appropriate optimisation algorithm was selected. The comprehensive overall model was incorporated within the optimisation code. In order to do so, the optimal control problem presented within this thesis was adapted to the sequential quadratic programming algorithm implemented within MATLAB. This selection was justified by the saving of computational time compared to evolutionary algorithms like genetic algorithms and the robustness of the code compared to specialised SQP implementations.

Within a series of optimal control calculations the efficiency of the combined model and the performance of the optimisation algorithm were evaluated. On the one hand, these calculations illustrated thoroughly that the specified SQP algorithm combined with the overall model could be used to simulate the model of a warehouse and calculate optimal trajectories for a fluid filled container within such warehouses. On the other hand, difficulties within the optimisation have been spotted and solution strategies in the sense of appropriate initial value settings have been formulated. Finally, a set of time optimal trajectories were calculated to illustrate dependencies of the motion on different physical properties like cube dimensions, distance to travel or fluid viscosity.

In order to be able to verify results of the optimisation algorithm, constraints have been specified which fulfil the limitations by the small- scale model rig. Using the overall model and the SQP algorithm, a time optimal trajectory for a specific motion has been calculated. Using the small-scale model rig, this optimal trajectory was verified. Furthermore, the author illustrated that the time optimal trajectory is not only based on the maximum acceleration used, but also on the maximum velocity gained during the motion.

8.2 Contribution to knowledge

Several aspects of the tools and models that were developed during the research have innovative significance.

The integration of various different interpolation schemes for the specification of the acceleration of the container and the fluid has opened new simulation capabilities for liquid motion. These capabilities have still not been realised in commercial CFD-codes like FIDAP.

A new combination of fluid motion and mechanical system motion was proposed and used to develop a finite difference CFD-code with Runge- Kutta solver for the mechanical system.

The study on such closed coupled fluid-structure systems is at the moment under intense research. These systems became applicable due to the improvements of computational resources.

The study on such a system was further promoted, applying optimisation. To the knowledge of the author, this is the first study on the numerical optimal calculation of a motion of a coupled system of ordinary and partial differential equations.

The applicability of an optimisation technique on such a system was verified using a practical experiment.

8.3 Future work

Already within this study several improvements of the models and the simulation and calculation of the fluid and mechanical system were thought over. In most cases these improvements had to step back due to their complexity of implementation or their increase of computational costs.

For future, additional research in the illustrated field of optimal control of sloshing liquids several different directions can be distinguished. In general they have all the common aims to improve the calculation speed and the solution domain of the results. This means the application of the calculations to a broader field of science.

Firstly the speed of computation can be further improved. This would allow the calculation of real time motion. With real time calculation the control system of the motion platform would be enabled to perform a closed loop calculation. Hence, the feedback of the actual motion of the fluid could influence the optimal motion of the platform. This would enable the control system to perform self tuning of the optimal motion parameters. The results obtained and illustrated in this thesis were calculated based on a limited access to modern computational resources. The workstation which was bought at the beginning of the research was in terms of computational speed modern at that time and outdated at the end of the research. Together with the modern parallel computational techniques there is an enormous amount of possible improvements in calculation speed.

Secondly several different computational techniques could be improved in order to allow the calculation of a broader area of initial conditions and a broader area of final solutions.

The following list illustrates in the left column the computational proposals and in the right column the possible improvements.

1. The various elements of the modelling of the process could be improved. This would include the calculation of the fluid motion in three dimensions, the integration of surface tension modelling and the modelling of turbulence within the fluid. Additionally a boundary layer model could be included to reflect the stick-slip conditions of the fluid at the container walls. Within the model of the mechanical system the non-linear characteristic curves of the servo motor and the other mechanical equipment could be integrated. A database could be generated to allow the calculation with various different configurations (rubber band-, leadscrew- or gear drive).	With these improvements the potential user of the code would be enabled to use the optimal control code on a wider range of problems. In particular the calculation of real warehouse configurations would be enabled. Nearly all containers in practical use are of a circular shape and the calculation of severe sloshing is only possible with turbulence modelling (see Section(6.1.2)).
2. A multiple shooting method could be integrated. Within this research a direct shooting method has been used in order to calculate the response of the cart and the fluid based on a specified parameter set of the acceleration of the drive.	The application of a multiple shooting method would enable the calculation of more speculative initial parameter sets, due to the fact that these procedures calculate more intermediate precise results.
3. A multiple-grid approach could be used to simulate the fluid motion. This would be very effective when using turbulent motion of the fluid.	Multiple-grid calculations have the advantage that the propagation of errors within the free surface motion of the fluid are highly limited due to smoothing of the solution using 'V' or 'W' multiple-grid cycles.

The common disadvantage of all of these algorithms and modelling enhancements is their additional computational cost. Therefore, it can be summarised that the resources of physical properties to be modelled in order to incorporate all interactions still absolutely exceeds the possibilities of modern computers. Therefore the range of improvements on this field of modelling will be still a science of simplifications for a long time to come.

References and Bibliography

[1] Abbott M. B., Basco D. R.: *Computational Fluid Dynamics An Introduction for Engineers.* Longman Scientific & Technical, Essex, 1989.

[2] Ancona M. G., DeVore C. R.: *Numerical Simulation of High-Field Transport Using a Flux-Corrected Transport Algorithm.* Proceedings of the Third International Workshop on Computational Electronics, Oregon State University, Portland OR, p. 256, 1994.

[3] *Introduction to FLOTRAN.* ANSYS, Inc., Documentation List, Canonsburg, 1996.

[4] *Annual review of fluid mechanics.* Annual Reviews Inc., Palo Alto, Calif., 1969 - 1993.

[5] Ascher U., Christiansen J., Russel R. D.: *Collocation software for boundary-value codes.* ACM Trans. Numer. Softw., Nr. 7, 209–222, 1981.

[6] Ascher U., Mattheij R., Russell R. D.: *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[7] Barcley A., Gill P. E., Rosen J. B.: *SQP methods and their application to numerical optimal control.* In *Variational Calculus, Optimal Control and Applications.* Edited by Schmidt W. H., Heier K., Bittner L., Bulirsch R., ISNM Vol. 124, pp. 207-222, Birkhäuser Verlag, Basel, Berlin, 1998.

[8] Barnard B. J. S., Mahoney J. J, Pritchard, W. G.: *The excitation of surface waves near a cut-off frequency.* Phil. Trans. R. Soc. Lond. A 112, 87-123, 1977.

[9] Behr M., Tezduyar T. E.: *Finite Element Solution Strategies for Large-Scale Flow Simulations.* Comp. Meth. in Appl. Mech. and Eng., 112, 3-24, 1994.

[10] Bishop R. H.: *Modern Control Systems Analysis and Design Using MATLAB.* Addison–Wesley, New York, Bonn, Paris, 1993.

[11] de Boor C.: *A Practical Guide to Splines.* Springer, New York, 1978.

[12] Broyden C. G.: *The Convergence of a Class of Double-rank Minimization Algorithms.* J. Inst. Maths. Applics., Vol. 6, pp. 76-9, 1970.

[13] Bulirsch R., Montrone F., Pesch H. J.: *Abort Landing in the Presence of Windshear as a Minimax Optimal Control Problem, Part 2: Multiple Shooting and Homotopy.* J. of Optimization Theory and Application, Vol 70, No. 2, 223-254, 1991.

[14] Bungartz H.-J., Frank A., Meier F., Neunhoeffer T., Schulte S.: *Fluid structure interaction: 3d numerical simulation and visualization of a micropump.* Institut für Informatik der TU München, SFB-Report 342/06/97 A, 1997.

[15] Bungartz H.-J., Frank A., Meier F., Neunhoeffer T., Schulte S.: *Efficient Treatment of Complicated Geometries and Moving Interfaces for CFD Problems.* International FORTWIHR Conference, High Performance Scientific and Engineering Computing, München, 1998.

[16] Chapmann D. R.: AIAA (American Institute of Aeronautics and Astronautics), Journal 17, pp. 1293-1313, 1979.

[17] Cliff E. M., Heinkenschloss M., Shenoy A. R.: *An Optimal Control Problem for Flows with Discontinuities.*, ICAM (Interdisciplinary Center for Applied Mathematics), Virginia Polytechnic Institute and State University, September 1995 (revised February 1996).

[18] Connor A. M.: *The Synthesis of Hybrid Mechanisms Using Genetic Algorithms.*, Ph.D. Dissertation, Liverpool John Moores University, 1996.

[19] Dehmel A.: *Numerische Simulation dreidimensionaler Strömungen mit freien Randwerten.* Numerical simulation of 3D-flows with free boundaries. Diploma-Thesis, Technical University Munich, Department of Computer Science, 1996.

[20] Diekhoff H.-J., Lory P., Oberle H.-J., Pesch H.-J., Rentrop P., Seydel R.: *Comparing routines for the numerical solution of initial value problems of ordinary differential equations in multiple shooting.* Numer. Math., Nr. 27, 449–469, 1977.

[21] Driels M.: *Linear Control Systems Engineering.*, McGraw–Hill, New York, London, 1996.

[22] Dütsch H., Becker S., Lienhart H.: *Numerical and Experimental Investigations of Flow Induced by Harmonic Motion of a Circular Cylinder.*, Contributions to the 10th AG STAB/DGLR Symposium, Braunschweig, 1996.

[23] Dütsch H., Melling A., Durst F.: *Fluid Damped Oscillation of a Lamina at Large Initial Amplitudes.* International FORTWIHR Conference, High Performance Scientific and Engineering Computing, München, 1998.

[24] Egelja A., Schäfer M.: *Computation of free surface flows.* LSTM Bericht 439/N, Lehrstuhl für Strömungsmechanik, Technische Fakultät Universität Erlangen-Nürnberg, 1995.

[25] Emanuel P., Leff E.: *Introduction to Feedpack Control Systems.* McGraw–Hill, Tokyo, London, Hamburg, 2nd printing, 1984.

[26] Eriksson K., Estep D., Hansbo P., Johnson C.: *Introduction to Adaptive Methods for Differential Equations.* Acta Numerica, pp. 105-158, 1995.

[27] Fehlberg, E.: *Klassische Runge-Kutta Formeln fünfter und siebter Ordnung mit Schrittweiten-Kontrolle.* (Classical Runge-Kutta formula with step size control of order five and seven), Computing Nr. 4, 93–106, 1969.

[28] Feng Z. C., Sethna P. R.: *Symmetry–breaking bifurcations in resonant surface waves.* J. Fluid Mech., Vol. 199, PP. 495–518, 1989.

[29] Ferziger J. H., Perić M.: *Computational Methods for Fluid Dynamics.* Springer Verlag, Berlin, Heidelberg, New York, 1996.

[30] *FIDAP reference guide.* Revision 7, FDI Fluid Dynamics International Inc., 500 Davis Street, Suite 600, Evanston, IL 60201 USA, 1995.

[31] Fletcher C. A. J.: *Computational Techniques for Fluid Dynamics Volume I.* Springer Verlag, Berlin, Heidelberg, New York, 1991.

[32] Fletcher C. A. J.: *Computational Techniques for Fluid Dynamics Volume II.* Springer Verlag, Berlin, Heidelberg, New York, 1991.

[33] Fletcher R.: *A New Approach to Variable Metric Algorithms.* Computer Journal, Vol. 13, pp. 317-322, 1970.

[34] *FLUENT User's Manual.* Fluent Inc., Version 4, Lebanon, 1997.

[35] Fursikov A. V.: *Optimal Control Problems for Navier–Stokes System with Distributed Control Function.* In: Sritharan S. S. (ed.): *Optimal Control of Viscous Flow.*, SIAM, Philadelphia, 1998.

[36] Gill P. E., Murray W., Saunders M. A.: *Practical Optimization.* Academic Press, 1981.

[37] Gill P. E., Murray W., Wright H. H.: *Large-scale SQP Methods and their Application in Trajectory Optimization.* In *Computational Optimal Control.* Edited by Bulirsch R., Kraft D., ISNM Vol. 115, Birkhäuser, Basel, pp. 29-42, 1994.

[38] Glover F.: *Tabu Search–Part I.* ORSA Journal on Computing, Vol.1, Nr.3, pp. 190-206, 1989.

[39] Glover F.: *Tabu Search–Part II.* ORSA Journal on Computing, Vol.2, Nr.1, pp. 4-32, 1990.

[40] Glover F.: *Tabu Search: A Tutorial.* INTERFACES 20: 4, pp. 74-79, 1990.

[41] Glover F.: *Tabu search for nonlinear and parametric optimization.*, Discrete Applied Mathematics, Vol.49, pp. 231-255, 1994.

[42] Goldberg D. E.: *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison Wesley, 1989.

[43] Goldfarb D.: *A Family of Variable Metric Updates Derived by Variational Means.* Mathematics of Computing, Vol. 24, pp. 23-26, 1970.

[44] Gregory J., Lin C.: *Constrained Optimization in the Calculus of Variations and Optimal Control Theory.* Van Nostrand Reinhold, New York, London, 1992.

[45] Griebel M., Dornseifer T., Neunhoeffer T.: *Numerische Simulation in der Strömungsmechanik.* (Numerical Simulation within Fluid Dynamics) Vieweg, Braunschweig, Wiesbaden, 1995.

[46] Harlow F.H., Welch J.E., Shannon J.P., Daly B.J.: *The MAC method.* Report LA-3425, Los Alamos Scientific Laboratory, 1965.

[47] Hawken D.,F., Gottlieb J.J., Hansen J.S.: *Review of Some Adaptive Node – Movement Techniques in Finite – Element and Finite – Difference Solutions of Partial Differential Equations.* J. Comput. Phys., 95, 254-302, 1991.

[48] Heinkenschloss M., Vicente L N.: *Analysis of Inexact Trust-Region Interior-Point SQP Algorithms.* ICAM Report 95-06-01, Blacksburg, VA, 1996.

[49] Hilberg D.: *Akima-Interpolation.* CT Magazin für Computertechnik 206–214, 1989.

[50] Hirsch C.: *Numerical Computation of Internal and External Flows, Volume I., Fundamentals of Numerical Discretization.* John Wiley & Sons, Chichester, 1988.

[51] Hirsch C.: *Numerical Computation of Internal and External Flows, Volume II., Computational Methods for Inviscid and Viscid Flows.* John Wiley & Sons, Chichester, 1988.

[52] Hirt C. W., Nichols B. D.: *Volume of Fluid (VOF) method for the dynamics of free boundaries.* J. Comput. Phys., 39, 201-225, 1981.

[53] Hoffmann K. A., Chiang S. T.: *Computational Fluid Dynamics for Engineers – Volume I.* Engineering Education System, Wichita, Kansas, 1993.

[54] Hoffmann K. A., Chiang S. T.: *Computational Fluid Dynamics for Engineers – Volume II.* Engineering Education System, Wichita, Kansas, 1993.

[55] Huerta A., Liu W. K.: *Viscous Flow with Large Free Surface Motion.* Comp. Meth. in Appl. Mech. and Eng., 69, 277-324, 1988.

[56] Jahn J.: *Introduction to the Theory of Nonlinear Optimization.* Springer-Verlag, Berlin, London, Tokyo, 1994.

[57] Kreiss H.-O., Lorenz J.: *Initial-boundary value problems and the Navier-Stokes equations.* Acad. Press, Pure and applied mathematics Nr. 136, Boston, 1988.

[58] van Kan J.J.I.M., Segal A.: *Numerik partieller Differentialgleichungen für Ingenieure.* (The numerics of partial differential equations for engineers.) B.G. Teubner, Stuttgart, 1995.

[59] Kassim A. B.: *Application of Artificial Intelligence (AI) to Cam-design.* Ph.D. Thesis, John Moores University, Liverpool, to be published 1997.

[60] Kinnebrock W.: *Optimierung mit genetischen und selektiven Algorithmen.* (Optimisation with genetic and selective algorithms) R. Oldenburg Verlag, München, Wien, 1994.

[61] Kirk D. E.: *Optimal Control Theory An Introduction.* Prentice-Hall Inc., New Jersey, 1970.

[62] Kit E., Shemer L., Miloh T.: *Experimental and theoretical investigation of nonlinear sloshing waves in a rectangular channel.* J. Fluid Mech., Vol.181, PP. 265-291, 1987.

[63] Kraft D.: *TOMP – Fortran Modules for Optimal Control Calculations.* ACM Trans. Math. Softw., Vol. 20, No. 3, pp. 262-281, 1994.

[64] Kraft D.: *On converting optimal control problems into nonlinear programming codes.* In: Schittkowski K. (ed.): *Computational Mathematical Programming.*, Kluwer Academic Publishers, Dordrecht, Boston, London, 1995.

[65] Kraft D.: *Optimal Robot Path Planing.* In: Bestle D., Schiehlen W. (eds.): *IUTAM Symposium on Optimization of Mechanical Systems.*, NATO ASI Series 15, Springer, 261-280, 1995.

[66] Kraft D.: *Modeling and Simulation of Robots in* MAPLE. MapleTech, Vol. 1, No. 2, pp. 39-49, 1994.

[67] Kraft D., Leonpacher H., Schaupp M.: *Modellierung, Simulation und Optimierung komplexer Systeme.* (Modelling, simulation and optimisation of complex systems.) FHM-Journal, Nr. 1.98,pp. 69-73, München, 1998.

[68] Leonpacher H.: *The Problem of Sloshing Waves in a Rectangular Container as a Base to Illustrate Numerical Solution Strategies and Programs.* M.Sc. Dissertation, Liverpool, München 1995.

[69] Leonpacher H., Douglas S. S., Woolley N. H., Kraft D.: *Optimal Control of Sloshing Liquids.* In *Variational Calculus, Optimal Control and Applications.* Edited by Schmidt W. H., Heier K., Bittner L., Bulirsch R., ISNM Vol. 124, pp. 301-310, Birkhäuser Verlag, Basel, Berlin, 1998.

[70] Leonpacher H., Douglas S. S., Woolley N. H., Kraft D.: *Simulation and Optimization of Logistic Processes Involving Sloshing Media.* In: Bungartz H.-J., Durst F., Zenger Ch. (Eds.): High Performance Scientific and Engineering Computing, Springer, Berlin, 1999.

[71] LeVeque R. J.: *CLAWPACK – A software package for solving multi-dimensional conservation laws.* Proc. 5th International Conference on Hyperbolic Problems, Stony Brook, June, 1994, (J. Glimm et.al., eds.), World Scientific Press, 1996, pp. 188-197, 1996.

[72] *Using MATLAB Version 5.* The MathWorks, Inc., Natick, 1996.

[73] Branch M. A., Grace A.: *MATLAB Optimization Toolbox* The MathWorks, Inc., Natick, 1996.

[74] Mohammadi B.: *Fluid Dynamics Computation with NSC2KE; An User-Guide; Release 1.0.* INRIA Publications, Nr. RT-0164, Rocquencourt, 1994.

[75] More J. J., Wright S. J.: *Optimization Software Guide.* Siam Publications, Frontiers in Applied Mathematics 14, Philadelphia, 1993.

[76] Murray W., Wright M. H.: *Computation of the search direction in constrained optimization algorithms.* Math. Progr. Study, Nr. 16, 62–83, 1982.

[77] Muto K., Kasai Y., Nakahara M., Ishida Y.: *Experimental Tests on Sloshing Response of a Water Pool with submerged Blocks.* Pressure Vessels & Piping Const., 209-214, 1985.

[78] Nomura T.: *ALE finite element computations of fluid-structure interaction problems.* Comp. Meth. in Appl. Mech. and Eng., 112, 291-308, 1994.

[79] Norton R. L.: *Design of Machinery.* McGraw-Hill series in mechanical engineering, New York, 1992.

[80] Oertel jr. H., Laurien E.: *Numerische Strömungsmechanik.* (Numerical fluid dynamics.) Springer - Verlag, Berlin, Heidelberg, 1995.

[81] Oswatitsch K.: *Physikalische Grundlagen der Strömungslehre in Fluid Dynamics I.* (Physical foundations of flow priciples in Fluid Dynamics I.) Edited by Flügge S., Springer - Verlag, Berlin, 1959.

[82] Osypiw D.: *High Speed Automatic Assembley of Fuselinks.* Ph.D. thesis, Liverpool John Moores University, 1994.

[83] Otter M., Elmquist H., Cellier F. E.: *Modeling of Multibody Systems with the Object-Oriented Modeling Language Dymola.* Proc. NATO/ASI Computer Aided Analysis of Rigid and Flexible Mechanical Systems, Troia, Portugal, 1993.

[84] Patankar S. V., Spalding D. B.: *A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows.* Int. J. Heat and Mass Transfer, Vol. 15, 1787-1806, 1972.

[85] Pope A.: *Wind-tunnel testing.* Wiley, 2. ed., 5. print, New York, 1964.

[86] Popov G., Sankar S., Sankar T. S., Vatistas G. H.: *Liquid Sloshing in Rectangular Road Containers.* Computers Fluids, Vol. 21, No. 4, 551-569, 1992.

[87] Popov G., Sankar S., Sankar T. S., Vatistas G. H.: *Dynamics of liquid sloshing in horizontal cylindrical road containers.* Proc. Instn. Mech. Engrs., Vol. 207, pp. 399-406, 1993.

[88] Popov G., Sankar S., Sankar T. S.: *Optimal shape of a rectangular road container.* J. Fluids Struct., 1993.

[89] Pozrikidis C.: *Introduction to Theoretical and Computational Fluid Dynamics.* Oxford University Press, Oxford, New York, 1997.

[90] Powell M. J. D.: *A Fast Algorithm for Nonlinearly Constrained Optimization Calculations.* Numerical Analysis, G. A. Watson ed., Lecture Notes in Mathematics, Springer Verlag, Vol. 630, 1978.

[91] Prentis J. M.: *Dynamics of Mechanical Systems.* John Wiley & Sons, New York, Chichester, 2nd edition, 1986.

[92] Ravindran S. S.: *Numerical Approximation of Optimal Flow Control Problems by SQP Method.* In: Sritharan S. S. (ed.): *Optimal Control of Viscous Flow.*, SIAM, Philadelphia, 1998.

[93] Rees Jones J.: *Cams and Cam Mechanisms.* Mechanical Engineering Publications for the Institution of Mechanical Engineers, London, 1978.

[94] Rees Jones J.: *Designing for high-speed transfer.* Engineering, The Design Council, London, 1984.

[95] Rentrop P., Wever U.: *Theory and Application of the Exponential Spline.* Report No. 282, Math. Inst. TU München, 1991.

[96] Rosenbrock H. H.: *An Automatic Method for Finding the Greatest or Least Value of a Function.* Computer Journal 3, pp. 175-184, 1960.

[97] Schaupp M.: *Optimal Robot path planing.* Ph.D. Thesis, John Moores University, Liverpool, to be published 1998.

[98] Schäfer M., Schreck E., Wechsler K.: *An Efficient Parallel Solution Technique for the Incompressible Navier–Stokes Equations.* Notes on Numerical Fluid Mechanics, 47, Vieweg, Braunschweig, 1994.

[99] Schäfer M.: *Numerical simulation of coupled fluid-solid problems.* International FORTWIHR Conference, High Performance Scientific and Engineering Computing, München, 1998.

[100] Schittkowski K., Zillober C., Zotemantel R.: *Numerical Comparison of Nonlinear Programming Algorithms for Structural Optimization.* Strutural Optimization, Vol. 7, No. 1, pp. 1-28, 1994.

[101] Sciavicco L., Siciliano B.: *Modelling and Control of Robot Manipulators.* McGraw–Hill, New York, London, Tokyo, 1996.

[102] Shanno D. F.: *Conditioning of Quasi-Newton Methods for Function Minimization.* Mathematics of Computing, Vol. 24, pp. 647-656, 1970.

[103] Shenoy A. R., Cliff E. M., Heinkenschloss M.: *Thermal–Fluid Control via Finite–Dimensional Approximation.* AIAA Paper 96-1910, New Orleans, 1996.

[104] *SIMPACK User Manual Part 1.* Deutsche Forschungsanstalt für Luft und Raumfahrt DLR, Oberpfaffenhofen, 1994.

[105] Smith R. E., Goldberg D. E., Earickson J. A.: *SGA-C: A C-language Implementation of a Simple Genetic Algorithm.* TCGA Report No. 91002, Alabama, 1994.

[106] Smith D. R., Houghton J.: *Fluid mechanics through worked examples.* Cleaver-Hume, London, 1960.

[107] Soulaïmani A., Fortin M., Dhatt G., Ouellet Y.: *Finite Element Simulation of Two– and Three–Dimensional Free Surface Flows.* Comp. Meth. in Appl. Mech. and Eng., 86, 265-296, 1991.

[108] Stoer J., Bulirsch R.: *Numerische Mathematik 2.* (Numerical mathematics 2.) Springer, Berlin, 1990.

[109] von Stryk O.: *Numerical Solution of Optimal Control Problems by Direct Collocation.* In: Bulirsch R., Miele A., Stoer J., Well K. H. (eds.): *Optimal Control and Variational Calculus.*, ISNM Vol. 111 Birkhäuser, Basel, 129-143, 1991.

[110] Tezduyar T. E., Behr M., Liou J.: *A new strategy for Finite Element Computations Involving Moving Boundaries and Interfaces — The Deforming-Spatial-Domain / Space-Time Procedure: I. The Concept and the Preliminary Numerical tests.* Comp. Meth. in Appl. Mech. and Eng., 94, 339-351, 1992.

[111] Tezduyar T. E., Behr M., Liou J.: *A new strategy for Finite Element Computations Involving Moving Boundaries and Interfaces — The Deforming-Spatial-Domain / Space-Time Procedure: II. Computation of Free-Surface Flows, Two-Liquid Flows, and Flows with Drifting Cylinders.* Comp. Meth. in Appl. Mech. and Eng., 94, 353-371, 1992.

[112] Truckenbrodt E.: *Fluidmechanik Band 1.* Dritte Auflage, Springer - Verlag, Berlin, 1989.

[113] Ye Y.: *SOLNP Users' Guide.* Department of Management Sciences, University of Iowa, 1989.

Appendix A

Additional theory

A.1 NaSt2D

A.1.1 NaSt2D additions

In this appendix the files and additions within NaSt2D are given which have been necessary to compute the problem of fluid sloshing in a warehouse environment.

A.1.1.1 The file *ode_fun.c*

This routine is necessary to compute the result of the motion of the mechanical system.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <fcntl.h>
#include <unistd.h>
#include "datadef.h"
#include "init.h"
#include "spline.h"
#include "auxiliary.h"
#define MAX(A, B)  ((A) > (B) ? (A) : (B))
#define MIN(A, B)  ((A) < (B) ? (A) : (B))

/*---------------------------------------------------------- */
/*                  O D E - A L G O R I T H M                */
/*----------------------------------------------------------*/

void ODE45(double *GX,double *GY,double **txvor,double **polypx,double **tyvor,
double **polypy,double t0,double t_end,int n_dz,int numi,
```

```
int spmode,double **zu,double **zu_stop,double **Ffun,double F_G,double **l_end)
{
  double **alpha,**bbeta,**ggamma,exponent,t_i,temppx,temppy;
  double **dzu_m,**dzu,**zu_m,**zu_temp,**zu_temp2,hmax,h_gl,delta_e,
tau_e,toler_e;
  double **ddzu_temp;
  int i,j,k,count_k;

 /*-------------------------------------------------------------*/
 /* Runge-Kutta algorithm with Fehlberg coefficients of order 5(4)      */
 /*                                                             */
 /*   GX : Force of fluid in X-direction                        */
 /*   GY : Force of fluid in Y-direction                        */
 /*   txvor : Time-value pair of initial motion in X-direction  */
 /*   tyvor : Time-value pair of initial motion in Y-direction  */
 /*   polypx : Spline parameters for txvor                      */
 /*   polypy : Spline parameters for tyvor                      */
 /*   t0 : Initial time for ODE-solver                          */
 /*   t_end : Final time for ODE-solver                         */
 /*   n_dz : Number of ODEs, length of state vektors            */
 /*   numi : length of txvor,tyvor (0 ... numi)                 */
 /*-------------------------------------------------------------*/
 /*-------------------------------------------------------------*/
 /*       Allocation of space for data arrays                   */
 /*-------------------------------------------------------------*/
 toler_e = 0.0000001;          /*   Tolerance fuer RK45         */

 alpha = RMATRIX(0,0,0,4);
 bbeta = RMATRIX(0,4,0,5);
 ggamma = RMATRIX(0,1,0,5);

 ddzu_temp = RMATRIX(0,1,0,499);
 dzu_m = RMATRIX(0,5,0,n_dz-1);
 dzu = RMATRIX(0,0,0,n_dz-1);
 zu_m = RMATRIX(0,n_dz,0,499);
 zu_temp = RMATRIX(0,0,0,n_dz-1);
 zu_temp2 = RMATRIX(0,0,0,n_dz-1);

 alpha[0][0] = 1.0/4.0;
 alpha[0][1] = 3.0/8.0;
 alpha[0][2] = 12.0/13.0;
   alpha[0][3] = 1.0;
   alpha[0][4] = 1.0/2.0;

   bbeta[0][0]=1.0/4.0; bbeta[1][0]=3.0/32.0; bbeta[2][0]=1932.0/2197.0;
bbeta[3][0]=439.0/216.0;   bbeta[4][0]=-8.0/27.0;
   bbeta[0][1]=0.0; bbeta[1][1]=9.0/32.0; bbeta[2][1]=-7200.0/2197.0;
bbeta[3][1]=-8; bbeta[4][1]=2;
   bbeta[0][2]=0.0; bbeta[1][2]=0.0;  bbeta[2][2]=7296.0/2197.0;
bbeta[3][2]=3680.0/513.0;  bbeta[4][2]=-3544.0/2565.0;
   bbeta[0][3]=0.0; bbeta[1][3]=0.0;  bbeta[2][3]=0.0;  bbeta[3][3]=-845.0/4104.0;
```

```
bbeta[4][3]=1859.0/4104.0;
  bbeta[0][4]=0.0; bbeta[1][4]=0.0;  bbeta[2][4]=0.0;  bbeta[3][4]=0.0;
bbeta[4][4]=-11.0/40.0;
  bbeta[0][5]=0.0; bbeta[1][5]=0.0;  bbeta[2][5]=0.0;  bbeta[3][5]=0.0;
bbeta[4][5]=0.0;

  ggamma[0][0]=16.0/135.0;       ggamma[1][0]=-1.0/360.0;
  ggamma[0][1]=0.0;              ggamma[1][1]=0.0;
  ggamma[0][2]=6656.0/12825.0;   ggamma[1][2]=128.0/4275.0;
  ggamma[0][3]=28561.0/56430.0;  ggamma[1][3]=2197.0/75240.0;
  ggamma[0][4]=-9.0/50.0;        ggamma[1][4]=-1.0/50.0;
  ggamma[0][5]=2.0/55.0;         ggamma[1][5]=-2.0/55.0;

  exponent = 0.2;

  /*-----------------------------------------*/
  /* Initialisation                          */
  /*-----------------------------------------*/

for (i=0;i<=5;i++)
     for (j=0;j<=n_dz-1;j++)
dzu_m[i][j] = 0.0;

hmax = (t_end - t0)/16;
h_gl = hmax/8;

count_k = 0;
  /*-----------------------------------------*/
  /*          States                         */
  /*-----------------------------------------*/
  t_i   = t0;

  zu_m[0][count_k] = t_i;
  for (i=1;i<=(n_dz);i++)
zu_m[i][count_k] = zu[0][i-1];

  /*-----------------while ?-----------------*/
while ((t_i < t_end) && ((t_i + h_gl) > t_i))
{
  if (t_i + h_gl > t_end)
h_gl = t_end - t_i;

  /*-----------------------------------------*/
  /*------------  Compute the Slope  --------*/
  /*-----------------------------------------*/

  for (j=0;j<=n_dz-1;j++)
zu_temp[0][j] = zu[0][j];

  temppx = SPLINT(spmode, numi, txvor, polypx, t_i);
  temppy = SPLINT(spmode, numi, tyvor, polypy, t_i);
```

```
  ODE_FUN(dzu,zu_temp,temppx,temppy,Ffun,t0,txvor[0][numi],l_end);

  for (i=0;i<=(n_dz-1);i++)
{
dzu_m[0][i] = dzu[0][i];
}

  /*--------------------------*/
  /*   MATRIXMULTIPLIKATION   */
  /*--------------------------*/
  for (i=1;i<=5;i++)
{
  /*-----------------------------------------*/
  /*------------  Clear up          ---------*/
  /*-----------------------------------------*/
for (j=0;j<=n_dz-1;j++)
   zu_temp2[0][j]= 0.0;

for (j=0;j<=n_dz-1;j++)
   {
   for (k=0;k<=5;k++)
{
zu_temp2[0][j] = zu_temp2[0][j] +(dzu_m[k][j]*bbeta[i-1][k]);
}
   }

for (j=0;j<=n_dz-1;j++)
   {
   zu_temp2[0][j] *= h_gl;
   zu_temp[0][j] = zu_temp2[0][j]+ zu[0][j];
   }

        temppx = SPLINT(spmode, numi, txvor, polypx, (t_i+alpha[0][i-1]*h_gl));
        temppy =SPLINT(spmode,numi,tyvor,polypy,(t_i+alpha[0][i-1]*h_gl));

        ODE_FUN(dzu,zu_temp,temppx,temppy,Ffun,t0,txvor[0][numi],l_end);

for (j=0;j<=(n_dz-1);j++)
   {
   dzu_m[i][j] = dzu[0][j];
   /* printf("dzu[0][%d]= %.4e ",j,dzu[0][j]); */
   }
}
  /*-----------------------------------------------*/
  /*  Estimate the error and the acceptable error  */
  /*-----------------------------------------------*/

   delta_e = 0.0;
   tau_e = 1.0;

   for (j=0;j<=n_dz-1;j++)
```

```
   zu_temp2[0][j]= 0.0;

   for (j=0;j<=n_dz-1;j++)
{
for (k=0;k<=5;k++)
   zu_temp2[0][j] += dzu_m[k][j]*ggamma[1][k];

zu_temp2[0][j] *= h_gl;
zu_temp2[0][j] = fabs(zu_temp2[0][j]);

delta_e = MAX(delta_e,zu_temp2[0][j]);
tau_e = MAX(tau_e,fabs(zu[0][j]));
}

   tau_e *= toler_e;

   /*-----------------------------------------------*/
   /*  Update only if error acceptable              */
   /*-----------------------------------------------*/

   for (j=0;j<=n_dz-1;j++)
   zu_temp2[0][j]= 0.0;

   if (delta_e <= tau_e)
{
t_i += h_gl;
for (j=0;j<=n_dz-1;j++)
   {
   for (k=0;k<=5;k++)
zu_temp2[0][j] += dzu_m[k][j]*ggamma[0][k];

   zu_temp2[0][j] *= h_gl;
   zu[0][j] += zu_temp2[0][j];
   if( t0<=txvor[0][numi])
{
zu_stop[0][j]=zu[0][j];
}
   }
count_k += 1;

zu_m[0][count_k] = t_i;
for (i=1;i<=(n_dz);i++)
   {
   zu_m[i][count_k] = zu[0][i-1];
   }
}

  /*-----------------------------------------------*/
  /*  Update the step size                         */
  /*-----------------------------------------------*/

   if (delta_e != 0.0)
```

```
      h_gl = MIN(hmax, 0.8*h_gl*pow((tau_e/delta_e),exponent));


}

  /*----------------------------------------*/
  /*  Differentiation                       */
  /*----------------------------------------*/

 for (i=0;i<=count_k-1;i++)
{
ddzu_temp[0][i] = (zu_m[2][i]-zu_m[2][i+1])/(zu_m[0][i]-zu_m[0][i+1]);
ddzu_temp[1][i] = (zu_m[4][i]-zu_m[4][i+1])/(zu_m[0][i]-zu_m[0][i+1]);
}

  if (t0<=txvor[0][numi])
{
l_end[0][0]=zu[0][0];
l_end[0][1]=zu[0][2];
}

  *GX = ddzu_temp[0][count_k-1];  /* Wert von x_2 zur Zeit t_end */
  *GY = ddzu_temp[1][count_k-1] + F_G;


  /*----------------------------------------*/
  /* Free allocated space          */
  /*----------------------------------------*/
  FREE_RMATRIX(alpha,0,0,0,4);
  FREE_RMATRIX(bbeta,0,4,0,5);
  FREE_RMATRIX(ggamma,0,1,0,5);

  FREE_RMATRIX(ddzu_temp,0,1,0,499);
  FREE_RMATRIX(dzu_m,0,5,0,n_dz-1);
  FREE_RMATRIX(dzu,0,0,0,n_dz-1);
  FREE_RMATRIX(zu_m,0,n_dz,0,499);
  FREE_RMATRIX(zu_temp,0,0,0,n_dz-1);
  FREE_RMATRIX(zu_temp2,0,0,0,n_dz-1);
}
```

A.1.1.2 The file *spline.c*

This routine is necessary to compute the parameters representing the interpolated function, and to calculate interpolated values for the acceleration.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <fcntl.h>
```

```
#include <unistd.h>

#define A1 (0.166666666657193)
#define A2 (0.00833333363787823)
#define A3 (0.000198409277128940)
#define A4 (0.0000027139911687000)
#define PHI(Z) (A1+(A2+(A3+A4*Z)*Z)*Z)
#define XPHI(Z) (Z*((Z+Z)*(A2+Z*Z*(2*A3+(3*A4)*Z*Z))))

int LEFT(int LXT,double **XT, double X, int MFLAG)
{
/*---------------------------------------------------------------------*/
/*   LEFT    FINDS INDEX LEFT OF AN ARRAY XT FOR WHICH XT(LEFT) */
/*        LIES IMMEDIATELY LEFT OF X */
/* */
/*PURPOSE: */
/*        FINDS INDEX LEFT OF AN ARRAY XT FOR WHICH XT(LEFT) */
/*        LIES IMMEDIATELY LEFT OF X */
/* */
/*INPUT ARGUMENTS: */
/*  LXT    : NUMBER OF ELEMENTS IN VECTOR XT */
/*  XT     : VECTOR OF LENGTH LXT STORING THE ABSCISSAE */
/*  X      : X-VALUE FOR WHICH THE INDEX LEFT IS TO BE FOUND */
/* */
/*OUTPUT ARGUMENTS: */
/*  LEFT   : INDEX FOR WHICH XT(LEFT) LIES IMMEDIATELY LEFT OF X */
/*  MFLAG : FLAG SET IN THE FOLLOWING MANNER */
/*          LEFT  MFLAG */
/*           1     -1     IF               X .LT. XT(1) */
/*           I      0     IF  XT(I)   .LE. X .LT. XT(I+1) */
/*          LXT     1     IF  XT(LXT) .LE. X */
/* */
/*METHOD: */
/*  THAT OF CARL DE BOOR AS DESCRIBED ON PAGE 91 FF. IN: */
/*  /1/ DE BOOR,C. (1978) A PRACTICAL GUIDE TO SPLINES. */
/*      APPLIED MATHEMATICAL SCIENCES, VOLUME 27. */
/*      NEW-YORK-HEIDELBERG-BERLIN: SPRINGER. */
/* */
/*IMPLEMENTED BY: */
/*   KRAFT,D., DLR - INSTITUT FUER DYNAMIK DER FLUGSYSTEME */
/*               D-8031 OBERPFAFFENHOFEN */
/* */
/*STATUS: 15. JANUARY 1980 */
/* */
/*SUBROUTINES REQUIRED: NONE       */
/*---------------------------------------------------------------------*/
/*---------------------------------------------------------------------*/

int IHI,ISTEP,MIDDLE,abbr;
static int ILO;

      ILO=0;
      IHI=ILO+1;
      if(IHI < LXT)  goto  A10;
      if(X >= XT[0][LXT])  goto A110;
      if(LXT <= 1)  goto  A90;
      ILO=LXT-1;
      IHI=LXT;
A10:  if(X >= XT[0][IHI]) goto  A40;
      if(X >= XT[0][ILO]) goto A100;
      ISTEP=1;
A20:  IHI=ILO;
      ILO=IHI-ISTEP;
      if(ILO <= 1)   goto  A30;
```

```
      if(X >= XT[0][ILO])  goto  A70;
      ISTEP=ISTEP+ISTEP;
goto  A20;
A30:  ILO=1;
      if(X < XT[0][1]) goto  A90;
goto  A70;
A40:  ISTEP=1;
A50:  ILO=IHI;
      IHI=ILO+ISTEP;
      if(IHI >= LXT) goto  A60;
      if(X < XT[0][IHI]) goto  A70;
      ISTEP=ISTEP+ISTEP;
goto  A50;
A60:  if(X >= XT[0][LXT]) goto A110;
      IHI=LXT;
A70:  MIDDLE=(ILO+IHI)/2;
      if(MIDDLE == ILO) goto A100;
      if(X < XT[0][MIDDLE]) goto  A80;
      ILO=MIDDLE;
goto A70;
A80:  IHI=MIDDLE;
  goto  A70;
A90:  MFLAG=-1;
      return(1);
goto A120;
A100: MFLAG=0;
      return(ILO);
goto A120;
A110: MFLAG=1;
      return(LXT);
A120: {}
}
/*END OF LEFT*/



double SPLINT(int MODE,int N,double **X,double **poly,double XS)
{
/*---------------------------------------------------------------------*/
/*---------------------------------------------------------------------*/
/*SPLINT  INTERPOLATES IN THE RANGE OF COEFFICIENT SETS FOR CUBIC */
/* */
/*        OR EXPONENTIAL SPLINES ESTABLISHED IN SUBROUTINE SPLINE */
/* */
/*PURPOSE: */
/*  INTERPOLATION  WITHIN COEFFICIENT SETS FOR CUBIC OR EXPONENTIAL */
/*  SPLINES */
/*  */
/*INPUT ARGUMENTS: */
/*  MODE  : INDICATES ORDER OF INTERPOLATION: */
/*          1 = PIECEWISE CONSTANT */
/*          2 = PIECEWISE LINEAR */
/*          3 = PIECEWISE CUBIC SPLINE */
/*          4 = PIECEWISE EXPONENTIAL SPLINE */
/*              WITH TENSIONS A PRIORI CALCULATED */
/*          5 = PIECEWISE EXPONENTIAL SPLINE */
/*              WITH TENSIONS GIVEN BY USER */
/*          6 = AKIMA'S INTERPOLANT */
/*  N     : NUMBER OF DATA FOR WHICH THE COEFFICIENT SET HAS BEEN */
/*          CALCULATED IN SUBROUTINE SPLINE PREVIOUSLY */
/*  X     : VECTOR OF LENGTH N STORING THE ABSCISSAE */
/*          AND THE DATA POINTS (2 x N) */
/*  poly  : MATRIX OF LENGTH N x 4 EACH STORING THE COEFFICIENT SET */
```

```
/*  XS     : ABSCISSA AT WHICH INTERPOLATION IS REQUIRED */
/* */
/*OUTPUT ARGUMENTS: */
/*  SPLINT: VALUE OF THE INTERPOLATING ORDINATE AT ABSCISSA XS */
/* */
/*METHOD: */
/*  THAT OF CHR. REINSCH, P. RENTROP & D. KRAFT AS DESCRIBED IN: */
/*  /1/ BULIRSCH,R.,RUTISHAUSER,H.: INTERPOLATION UND GENAEHERTE */
/*        QUADRATUR. IN:SAUER,R.,SZABO,I.(EDS.) MATHEMATISCHE HILFS-*/
/*        MITTEL DES INGENIEURS,VOL.III. BERLIN-HEIDELBERG-NEW YORK: */
/*        SPRINGER, 1968. */
/*  /2/ RENTROP,P.: AN ALGORITHM FOR THE COMPUTATION OF THE */
/*        EXPONENTIAL SPLINE. NUMER. MATH. 35 (1980) 81-93. */
/*  /3/ KRAFT,D.: FIRST DERIVATIVES OF EXPONENTIAL SPLINES. */
/*                UNPUBLISHED MANUSCRIPT (1984). */
/*  /4/ AKIMA, H: A NEW METHOD OF INTERPOLATION AND SMOOTH CURVE FITTING*/
/*        BASED ON LOCAL PROCEDURES. J. ACM 17 (1970) 589-602. */
/* */
/*IMPLEMENTED BY: */
/*   KRAFT,D., DLR - INSTITUT FUER DYNAMIK DER FLUGSYSTEME */
/*             D-8031 OBERPFAFFENHOFEN */
/*REFORMULATED IN C: */
/*   LEONPACHER,H., FACHHOCHSCHULE MUENCHEN FB03 */
/*             D-80335 MUENCHEN */
/* */
/*STATUS: 14. Januar 1997 */
/*SUBROUTINES REQUIRED: * = DIRECT CALL */
/*    LEFT   */
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/

int IP,IS,MFLAG;
double H,U,V,Z,D1,D2,HP,XT,DPHI,ZERO,HALF,zwi;

ZERO = 0;
HALF = 0.5;

/*  FIND INDEX OF ABSCISSA WHICH LIES IMMEDIATELY LEFT OF XS */

      IS = LEFT(N,X,XS,MFLAG);
      XT = XS;
      if(MFLAG == -1) XT = X[0][0];
      if(MFLAG ==  1) XT = X[0][N];

/*  CHOOSE INTERPOLATION ORDER  */

switch(MODE) {
/*  CONSTANT SPLINE  */
    case 1:{if(MFLAG ==  1) IS = IS-1; return(X[1][IS]); break;}
/*  LINEAR SPLINE  */
    case 2:{zwi = X[1][IS]; U = XT-X[0][IS]; if(U == ZERO) { return(zwi); break;}
    return(zwi+poly[0][IS]*U); break;}
/*  CUBIC SPLINE  & AKIMA SPLINE */
    case 3:{zwi = X[1][IS]; U = XT-X[0][IS]; if(U == ZERO) { return(zwi); break;}
    return(zwi+(poly[0][IS]+(poly[1][IS]+poly[2][IS]*U)*U)*U); break;}
/*  EXPONENTIAL SPLINE */
    case 4:{zwi = X[1][IS]; U = XT-X[0][IS]; if(U == ZERO) { return(zwi); break;}
    IP = IS+1; H = X[0][IP]-X[0][IS]; U = U/H; V = 1.0-U; HP = H*poly[3][IS];
    if(HP <= HALF) {HP = HP*HP; D1 = U*U; D2 = V*V; return(U*(X[1][IP]+
    poly[0][IS]*poly[2][IP]*(D1*PHI(HP*D1)-poly[1][IS]))
    + V*(X[1][IS]+poly[0][IS]*poly[2][IS]*(D2*PHI(HP*D2)
    -poly[1][IS]))); break;}
    else {D1 = exp(-HP*U); D2 = exp(-HP*V); return(U*X[1][IP]+
    (poly[0][IS]*D2*(1-D1*D1)-U)*poly[1][IS]*poly[2][IP]
    + V*X[1][IS]+(poly[0][IS]*D1*(1.0-D2*D2)-V)*poly[1][IS]*poly[2][IS]); break;}}
```

```
/*  EXPONENTIAL SPLINE */
    case 5:{zwi = X[1][IS]; U = XT-X[0][IS]; if(U == ZERO) { return(zwi); break;}
    IP = IS+1; H = X[0][IP]-X[0][IS]; U = U/H; V = 1.0-U; HP = H*poly[3][IS];
    if(HP <= HALF) {HP = HP*HP; D1 = U*U; D2 = V*V; return(U*(X[1][IP]+
    poly[0][IS]*poly[2][IP]*(D1*PHI(HP*D1)-poly[1][IS]))
    + V*(X[1][IS]+poly[0][IS]*poly[2][IS]*(D2*PHI(HP*D2)
    -poly[1][IS]))); break;}
    else {D1 = exp(-HP*U); D2 = exp(-HP*V); return(U*X[1][IP]+
    (poly[0][IS]*D2*(1-D1*D1)-U)*poly[1][IS]*poly[2][IP]
    + V*X[1][IS]+(poly[0][IS]*D1*(1.0-D2*D2)-V)*poly[1][IS]*poly[2][IS]); break;}}
/*  CUBIC SPLINE  & AKIMA SPLINE */
    case 6:{zwi = X[1][IS]; U = XT-X[0][IS]; if(U == ZERO) { return(zwi); break;}
    return(zwi+(poly[0][IS]+(poly[1][IS]+poly[2][IS]*U)*U)*U); break;}
       }


 /*END OF SPLINT*/
}


double DSPLNT(int MODE,int N,double **X,double **poly,double XS)
{
/*-------------------------------------------------------------------*/
/*  FIRST DERIVATIVE OF SPLINE FUNCTION   */
/*-------------------------------------------------------------------*/
int IS,IP,MFLAG;
double XT,ZERO,zwi,U,H,HP,V,HALF,D1,D2;

ZERO = 0;
HALF = 0.5;

      IS = LEFT(N,X,XS,MFLAG);
      XT = XS;
      if(MFLAG == -1) XT = X[0][0];
      if(MFLAG ==  1) XT = X[0][N];
      if(MFLAG ==  1) IS = IS-1;


switch(MODE) {
/*  CONSTANT SPLINE  */
    case 1:{return(ZERO); break;}
/*  LINEAR SPLINE  */
    case 2:{return(poly[0][IS]); break;}
/*  CUBIC & AKIMA SPLINE  */
    case 3:{zwi = poly[0][IS]; return(zwi); U = XT-X[0][IS]; if(U == ZERO) break;
    zwi = zwi+(2*poly[1][IS]+3*poly[2][IS]*U)*U; return(zwi); break;}
/*  EXPONENTIAL SPLINE  */
    case 4:{IP = IS+1; U = XT-X[0][IS]; H = X[0][IP]-X[0][IS]; U = U/H;
    V = 1.0-U; HP = H*poly[3][IS];
    if(HP <= HALF) {D1 = U*U; D2 = V*V; U = HP*U; V = HP*V; HP = HP*HP;
    zwi =  X[1][IP] + poly[0][IS]*poly[2][IP]*
(  D1*(3*PHI(HP*D1)+XPHI(U))-poly[1][IS])
      -X[1][IS] - poly[0][IS]*poly[2][IS]*
(  D2*(3*PHI(HP*D2)+XPHI(V))-poly[1][IS]); return(zwi/H);  break;}
    else {D1 = exp(-HP*U); D2 = exp(-HP*V);
    zwi =  X[1][IP] + (poly[0][IS]*HP*D2*(1.0+D1*D1)-1.0)*poly[1][IS]*poly[2][IP]
      - X[1][IS] - (poly[0][IS]*HP*D1*(1.0+D2*D2)-1.0)*poly[1][IS]*poly[2][IS];
return(zwi/H);  break;}}
/*  EXPONENTIAL SPLINE  */
    case 5:{IP = IS+1; U = XT-X[0][IS]; H = X[0][IP]-X[0][IS]; U = U/H;
    V = 1.0-U; HP = H*poly[3][IS];
    if(HP <= HALF) {D1 = U*U; D2 = V*V; U = HP*U; V = HP*V; HP = HP*HP;
    zwi =  X[1][IP] + poly[0][IS]*poly[2][IP]*
(  D1*(3*PHI(HP*D1)+XPHI(U))-poly[1][IS])
      -X[1][IS] - poly[0][IS]*poly[2][IS]*
```

```
(  D2*(3*PHI(HP*D2)+XPHI(V))-poly[1][IS]); return(zwi/H);  break;}
    else {D1 = exp(-HP*U); D2 = exp(-HP*V);
    zwi =  X[1][IP] + (poly[0][IS]*HP*D2*(1.0+D1*D1)-1.0)*poly[1][IS]*poly[2][IP]
      - X[1][IS] - (poly[0][IS]*HP*D1*(1.0+D2*D2)-1.0)*poly[1][IS]*poly[2][IS];
return(zwi/H);  break;}}
/*  CUBIC & AKIMA SPLINE  */
    case 6:{zwi = poly[0][IS]; U = XT-X[0][IS]; return(zwi); if(U == ZERO) break;
    zwi = zwi+(2*poly[1][IS]+3*poly[2][IS]*U)*U; return(zwi); break;}
}

 /*END OF DSPLNT*/
}
```

```
void SPLIN(int MODE, int N, double **X, double **poly)
{
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
/*SPLINE  CALCULATES COEFFICIENT SETS FOR LINEAR, CUBIC OR EXPONENTIAL */
/*        INTERPOLATING SPLINES */
/*         */
/*PURPOSE: */
/*  CALCULATION OF COEFFICIENT SETS FOR LINEAR, CUBIC, OR EXPONENTIAL */
/*  INTERPOLATING SPLINES. */
/*  ALSO TENSION PARAMETERS FOR VISUALLY PLEASING EXPONENTIAL SPLINE */
/*  INTERPOLANTS ARE CALCULATED. */
/*  TO BE USED FOR INTERPOLATION IN CONNECTION WITH FUNCTION SPLINT */
/*         */
/*INPUT ARGUMENTS: */
/*  MODE  : INDICATES ORDER OF INTERPOLATION: */
/*          1 = PIECEWISE CONSTANT */
/*          2 = PIECEWISE LINEAR */
/*          3 = PIECEWISE CUBIC SPLINE */
/*          4 = PIECEWISE EXPONENTIAL SPLINE */
/*              WITH TENSIONS A PRIORI CALCULATED */
/*          5 = PIECEWISE EXPONENTIAL SPLINE */
/*              WITH TENSIONS GIVEN BY USER */
/*          6 = AKIMA'S INTERPOLANT */
/*  N     : NUMBER OF DATA FOR WHICH THE COEFFICIENT SET IS TO BE */
/*          CALCULATED */
/*  X     : MATRIX OF LENGTH N x 2 STORING THE ABSCISSAE */
/*      AND THE DATA POINTS */
/*         */
/*CHANGED ARGUMENTS: */
/*  poly : MATRIX OF LENGTH N  x 4 STORING THE COEFFICIENT SET */
/*          TO BE USED FOR INTERPOLATION IN FUNCTION SPLINT. */
/*          THESE TOGETHER WITH X ARE INPUT ARGUMENTS TO */
/*          SPLINT AND MAY NOT BE CHANGED BY THE USER BETWEEN CALLS. */
/* */
/*REMARK: */
/*  A PRACTICAL WAY TO FIND SUITABLE STIFFNESS PARAMETERS */
/*  IS DESCRIBED IN /2/ AND IS IMPLEMENTED IN SUBROUTINE GENERA */
/* */
/*METHOD: */
/*  THAT OF CHR. REINSCH AND P. RENTROP AS DESCRIBED IN: */
/*  /1/ BULIRSCH,R.,RUTISHAUSER,H.: INTERPOLATION UND GENAEHERTE */
/*        QUADRATUR. IN:SAUER,R.,SZABO,I.(EDS.) MATHEMATISCHE HILFS-*/
/*        MITTEL DES INGENIEURS,VOL.III. BERLIN-HEIDELBERG-NEW YORK: */
/*        SPRINGER, 1968. */
/*  /2/ RENTROP,P.: AN ALGORITHM FOR THE COMPUTATION OF THE */
/*        EXPONENTIAL SPLINE. NUMER. MATH. 35 (1980) 81-93. */
/*  /3/ RENTROP,P.,WEVER,U.: THEORY AND APPLICATION OF THE */
/*        EXPONENTIAL SPLINE. REP. 282, MATH. INST. TU MUENCHEN, 1991. */
/*  /4/ AKIMA, H: A NEW METHOD OF INTERPOLATION AND SMOOTH CURVE FITTING*/
/*        BASED ON LOCAL PROCEDURES. J. ACM 17 (1970) 589-602. */
```

```
/* */
/*IMPLEMENTED BY: */
/*   KRAFT,D., DLR - INSTITUT FUER DYNAMIK DER FLUGSYSTEME */
/*              D-8031 OBERPFAFFENHOFEN */
/* */
/*COPYRIGHT : */
/*  *******   D L R   ******* */
/*  *******  1 9 8 6  ******* */
/*  *******   F H M   ******* */
/*  *******  1 9 9 1  ******* */
/* */
/*CHANGES: */
/*INCORPORATION OF ZERO-ORDER SPLINES */
/*TOGETHER WITH PARAMETER MODE FOR SPLINE SELECTION */
/*31. JANUAR 1986 */
/*INCORPORATION OF AKIMA'S INTERPOLANT */
/*14. JUNI 1989 */
/*INCORPORATION OF VISUALLY PLEASING INTERPOLANT */
/*OF RENTROP & WEVER */
/*16. JUNI 1991 */
/*REFORMULATED IN C: */
/*   LEONPACHER,H., FACHHOCHSCHULE MUENCHEN FB03 */
/*              D-80335 MUENCHEN */
/* */
/*STATUS: 14. Januar 1997 */
/* */
/*SUBROUTINES REQUIRED: NONE  */
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/

int I,IBACK,IS;
double C1,C2,D1,D2,H,HP,U,V,W,Z,ZERO,TWO,THREE,HALF,QUART,UMAX;

ZERO = 0;
HALF = 0.5;
TWO = 2;
THREE = 3;
HALF = 0.5;
QUART = 0.25;
UMAX = 25.0;
IS = 6;

/*  Is MODE in it's range?  */

     if((MODE < 1) || (MODE > 6)) {printf("SPLINE: MODE not in its range; MODE= %d\n", MODE); abort();}

/*  DO YOU HAVE MONOTONE ABSCISSAE?  */

     for(I=0;I<=(N-1);I++) {if(X[0][I] >= X[0][I+1]) for(I=0;I<=(N-1);I++) {if(X[0][I] <= X[0][I+1])
{printf("SPLINE: non-monotonic abscissae: %.3e, %.3e\n",  X[0][I], X[0][I+1]); abort();}}}

/*  CHOOSE INTERPOLATION ORDER  */

switch(MODE) {
    case 1:{ break;}
    case 2:{ for(I=0;I<=(N-1);I++) poly[0][I] = (X[1][I+1]-X[1][I])/(X[0][I+1]-X[0][I]); break;}
/*  LINEAR SPLINE COEFFICIENTS  */
    case 3:{ V = ZERO;  /*  COMPUTATION OF THE ELEMENTS OF THE TRIDIAGONAL SYSTEM  */
for(I=0;I<=(N-1);I++) {poly[2][I] = X[0][I+1]-X[0][I];
U = (X[1][I+1]-X[1][I])/poly[2][I];
    poly[1][I] = U-V; V = U;}                              /*  CUBIC SPLINE COEFFICIENTS  */
U = ZERO; V = U; poly[1][0] = V;
for(I=1;I<=(N-1);I++) {poly[1][I] = poly[1][I]+U*poly[1][I-1]; poly[0][I] = TWO*(X[0][I-1]-
    X[0][I+1])-U*V;  V = poly[2][I]; U = V/poly[0][I];}
```

```
/*  SOLUTION OF THE TRIDIAGONAL SYSTEM  */
poly[0][N] = ZERO; poly[1][N] = ZERO; poly[2][N] = ZERO;
for(I=1;I<=(N-1);I++) {IBACK = N-I; poly[1][IBACK] = (poly[2][IBACK]*poly[1][IBACK+1]-
poly[1][IBACK])/poly[0][IBACK];}
for(I=0;I<=(N-1);I++) {V = poly[2][I]; U = poly[1][I+1]-poly[1][I];
poly[2][I] = U/V; poly[1][I] = THREE*poly[1][I]; poly[0][I] = (X[1][I+1]-
X[1][I])/V-(poly[1][I]+U)*V;}  break;}

    case 4:{
/*  EXPONENTIAL SPLINE COEFFICIENTS  --  A PRIORI COMPUTATION OF THE TENSION PARAMETERS  */
for(I=0;I<=(N-1);I++) poly[2][I] = X[0][I+1]-X[0][I];
for(I=1;I<=(N-1);I++) poly[1][I] = (X[1][I+1]-X[1][I])/poly[2][I]-(X[1][I]-X[1][I-1])/
poly[2][I-1];
for(I=1;I<=(N-2);I++) {if(poly[1][I]*poly[1][I+1]  ==  ZERO)   /*   ABNORMAL CASE */
poly[0][I] = UMAX;
    else if(poly[1][I]*poly[1][I+1]  <  ZERO) /*    MONOTONICITY    */
{if(X[1][I+1]  ==  X[1][I]) poly[0][I] = UMAX;
else poly[0][I] = poly[2][I]*fabs((poly[1][I+1]-poly[1][I])/(X[1][I+1]-X[1][I]));}
    else  /*     CONVEXITY      */
   {H = fabs(poly[1][I]/poly[1][I+1]); if (H >= (1.0/H)) poly[0][I] = H ;else poly[0][I] = 1.0/H;}
    if (UMAX <= (poly[0][I])) poly[0][I] = UMAX;
    if(poly[0][I]  <  THREE) poly[0][I] = ZERO;}
poly[0][0] = poly[0][1]; poly[0][N-1] = poly[0][N-2];
for(I=0;I<=(N-1);I++) poly[3][I] = poly[0][I]/poly[2][I];
/*  COMPUTATION OF THE ELEMENTS OF THE TRIDIAGONAL SYSTEM  */
U = X[1][0];
for(I=1;I<=(N);I++) {V = X[1][I]; H = X[0][I]-X[0][I-1]; poly[2][I] = (V-U)/H; U = V;
    HP = H*poly[3][I-1];
    if(HP > HALF) {D1 = exp(-HP); D2 = D1*D1; W = 1.0-D2; C1 = W*HP; C2 = W/HP; W = H/C1;
poly[0][I] = (1.0-C2+D2)*W; poly[1][I] = (C2-TWO*D1)*W;}
    else {HP = HP*HP; C1 = PHI(HP); W = H/(1.0+HP*C1); HP = QUART*HP;
C2 = 1.0+HP*PHI(HP);
poly[0][I] = (HALF*C2*C2-C1)*W; poly[1][I] = C1*W;}}
/*  GENERATE THE TRIDIAGONAL SYSTEM  */
U = ZERO; poly[2][0] = U;
for(I=1;I<=(N-1);I++) {
    poly[0][I] = poly[0][I]+poly[0][I+1]-U*poly[1][I];
    poly[2][I] = poly[2][I+1]-poly[2][I]-U*poly[2][I-1];
    U = poly[1][I+1]/poly[0][I];}
/*  SOLUTION OF THE TRIDIAGONAL SYSTEM  */
poly[2][N] = ZERO;
for(I=1;I<=(N-1);I++) {IBACK = N-I;
    poly[2][IBACK] = (poly[2][IBACK]-poly[1][IBACK+1]*poly[2][IBACK+1])/poly[0][IBACK];}
/*  STORE AUXILIARY TERMS FOR SPLINT & DSPLNT IN A & B  */
for(I=0;I<=(N-1);I++) {H = X[0][I+1]-X[0][I]; HP = H*poly[3][I];
    if(HP > HALF) {D1 = exp(-HP); poly[0][I] = 1.0/(1.0-D1*D1);
poly[1][I] = 1.0/(poly[3][I]*poly[3][I]);}
    else {HP = HP*HP; D1 = PHI(HP); D2 = H*H/(1.0+HP*D1);
poly[0][I] = D2; poly[1][I] = D1;}} break;}

    case 5:{ /*  COMPUTATION OF THE ELEMENTS OF THE TRIDIAGONAL SYSTEM  */
U = X[1][0];
for(I=1;I<=(N);I++) {V = X[1][I]; H = X[0][I]-X[0][I-1]; poly[2][I] = (V-U)/H; U = V;
    HP = H*poly[3][I-1];
    if(HP > HALF) {D1 = exp(-HP); D2 = D1*D1; W = 1.0-D2; C1 = W*HP; C2 = W/HP; W = H/C1;
poly[0][I] = (1.0-C2+D2)*W; poly[1][I] = (C2-TWO*D1)*W;}
    else {HP = HP*HP; C1 = PHI(HP); W = H/(1.0+HP*C1); HP = QUART*HP;
C2 = 1.0+HP*PHI(HP);
poly[0][I] = (HALF*C2*C2-C1)*W; poly[1][I] = C1*W;}}
/*  GENERATE THE TRIDIAGONAL SYSTEM  */
U = ZERO; poly[2][0] = U;
for(I=1;I<=(N-1);I++) {
    poly[0][I] = poly[0][I]+poly[0][I+1]-U*poly[1][I];
    poly[2][I] = poly[2][I+1]-poly[2][I]-U*poly[2][I-1];
```

```
    U = poly[1][I+1]/poly[0][I];}
/*  SOLUTION OF THE TRIDIAGONAL SYSTEM  */
poly[2][N] = ZERO;
for(I=1;I<=(N-1);I++) {IBACK = N-I;
    poly[2][IBACK] = (poly[2][IBACK]-poly[1][IBACK+1]*poly[2][IBACK+1])/poly[0][IBACK];}
/*  STORE AUXILIARY TERMS FOR SPLINT & DSPLNT IN A & B  */
for(I=0;I<=(N-1);I++) {H = X[0][I+1]-X[0][I]; HP = H*poly[3][I];
    if(HP > HALF) {D1 = exp(-HP); poly[0][I] = 1.0/(1.0-D1*D1);
poly[1][I] = 1.0/(poly[3][I]*poly[3][I]);}
    else {HP = HP*HP; D1 = PHI(HP); D2 = H*H/(1.0+HP*D1);
poly[0][I] = D2; poly[1][I] = D1;}} break;}

    case 6:{ /*  AKIMA SPLINE COEFFICIENTS  */
for(I=0;I<=(N-1);I++) poly[3][I] = (X[1][I+1]-X[1][I])/(X[0][I+1]-X[0][I]);
/*  SLOPES AT NODES  */
for(I=2;I<=(N-2);I++) {
    if((poly[3][I-2] == poly[3][I-1]) && (poly[3][I] == poly[3][I+1]))
if(poly[3][I-1] != poly[3][I]) poly[0][I] = HALF*(poly[3][I-1]+poly[3][I]);
else poly[0][I] = poly[3][I];
    else {U=fabs(poly[3][I+1]-poly[3][I]); V=fabs(poly[3][I-1]-poly[3][I-2]);
poly[0][I] = (U*poly[3][I-1]+V*poly[3][I])/(U+V);}}
/*  BACKWARD EXTRAPOLATION  */
U = X[0][0]+X[0][1]-X[0][2]; H = U-X[0][0];
C1 = X[1][0]+H*(poly[3][0]+poly[3][0]-poly[3][1]);
D1 = (C1-X[1][0])/H; V = U+X[0][0]-X[0][1]; H = V-U;
C2 = C1+H*(D1+D1-poly[3][0]); D2 = (C2-C1)/H;
if((D2 == D1) && (poly[3][0] == poly[3][1]))
    if(D1 != poly[3][0]) poly[0][0] = HALF*(D1+poly[3][0]);
    else poly[0][0] = poly[3][0];
else {
    U = fabs(poly[3][1]-poly[3][0]);
    V = fabs(D1-D2);
    poly[0][0] = (U*D1+V*poly[3][0])/(U+V);}
if((D1 == poly[3][0]) && (poly[3][1] == poly[3][3]))
    if(poly[3][0] != poly[3][1]) poly[0][1] = HALF*(poly[3][0]+poly[3][1]);
    else poly[0][1] = poly[3][1];
else {
    U = fabs(poly[3][3]-poly[3][1]);
    V = fabs(poly[3][0]-D1);
    poly[0][1] = (U*poly[3][0]+V*poly[3][1])/(U+V);}
/*  FORWARD EXTRAPOLATION  */
U = X[0][N]+X[0][N-1]-X[0][N-2];
H = U-X[0][N];
C1 = X[1][N]+H*(poly[3][N-1]+poly[3][N-1]-poly[3][N-2]);
D1 = (C1-X[1][N])/H;
V = U+X[0][N]-X[0][N-1];
H = V-U;
C2 = C1+H*(D1+D1-poly[3][N-1]);
D2 = (C2-C1)/H;
if((poly[3][N-3] == poly[3][N-2]) && (poly[3][N-1] == D1))
    if(poly[3][N-2] != poly[3][N-1]) poly[0][N-1] = HALF*(poly[3][N-2]+poly[3][N-1]);
    else poly[0][N-1] = poly[3][N-1];
else {
    U = fabs(poly[3][N-1]-D1);
    V = fabs(poly[3][N-2]-poly[3][N-3]);
    poly[0][N-1] = (U*poly[3][N-2]+V*poly[3][N-1])/(U+V);}
if((poly[3][N-2] == poly[3][N-1]) && (D1 == D2))
    if(poly[3][N-1] != D1) poly[0][N] = HALF*(poly[3][N-1]+D1);
    else poly[0][N] = D1;
else {
    U = fabs(D1-D2);
    V = fabs(poly[3][N-1]-poly[3][N-2]);
    poly[0][N] = (U*poly[3][N-1]+V*D1)/(U+V);}

for(I=0;I<=(N-1);I++) {
```

```
    H = X[0][I+1]-X[0][I];
      poly[1][I] = (poly[3][I]+poly[3][I]+poly[3][I]-poly[0][I]-poly[0][I]-poly[0][I+1])/H;
    poly[2][I] = (poly[0][I]+poly[0][I+1]-poly[3][I]-poly[3][I])/(H*H);}

}
/*   END OF SPLINE   */
}}
```

A.1.1.3 The procedure *ODE_FUN*

This function represents and calculates the dynamic behaviour of the mechanical system. The various influences of friction, damping and acting forces are represented using ordinary differential equations.

```
/*-------------------------------------------------------*/
/* Calculation of the derivatives in X,Y                 */
/*-------------------------------------------------------*/
void ODE_FUN(RAAL **dzu,RAAL **zu_temp,RAAL temppx,RAAL temppy,
RAAL **Ffun,RAAL t,RAAL t_acc,RAAL **l_end)
{
 RAAL m_c,m_h,c,b,mu_r,F_c,F_b,F_r;

/*-------------------------------------------------------*/
/* Setting up the system                                 */
/*-------------------------------------------------------*/

 m_c  = 5.24;  /*  Mass of the cart            */
 m_h  = 2.5;   /*  Mass of the holding device  */
 c    = 50;    /*  Spring coefficient          */
 b    = 80;    /*  Damping coefficient         */
 mu_r = 0.1;   /*  Friction coefficient  */

/*-------------------------------------------------------*/
/* Calculation of various forces                         */
/*-------------------------------------------------------*/

   F_c = c*(zu_temp[0][0]-zu_temp[0][4]);  /* Spring force   */
   F_b = b*(zu_temp[0][1]-zu_temp[0][5]);  /* Damping force  */
   F_r = mu_r*((m_c+m_h)*9.81+Ffun[0][1]); /* Friction force */

/*-------------------------------------------------------*/
/* Calculation of the system of derivatives   ODE's      */
/*-------------------------------------------------------*/

   dzu[0][0] = zu_temp[0][1];
   dzu[0][1] = 1/(m_c)*(Ffun[0][0]-F_c-F_b);
   dzu[0][2] = zu_temp[0][3];
   dzu[0][3] = temppy;
   dzu[0][4] = zu_temp[0][5];
   dzu[0][5] =  1/(m_h)*(temppx-F_r+F_c+F_b);
```

}

A.1.2 NaSt2D discretisation

In this section of the appendix the discretisation of the Navier–Stokes equations within the CFD code NaSt2D are given. In the conservation of linear momentum the finite difference represention in the X-direction will be represented in the middle of the right edge of cell (i, j) with

$$\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} = \frac{1}{\Delta x}\left(\left(\frac{u_{i,j}+u_{i+1,j}}{2}\right)^2 - \left(\frac{u_{i-1,j}+u_{i,j}}{2}\right)^2\right) + \alpha\frac{1}{\Delta x}\left(\frac{|u_{i,j}+u_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i,j}|}{2}\frac{(u_{i-1,j}-u_{i,j})}{2}\right),$$

$$\left[\frac{\partial(uv)}{\partial y}\right]_{i,j} = \frac{1}{\Delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right) + \alpha\frac{1}{\Delta y}\left(\frac{|v_{i,j}+v_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i+1,j-1}|}{2}\frac{(u_{i,j-1}-u_{i,j})}{2}\right),$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} = \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\Delta x)^2},$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} = \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{(\Delta y)^2},$$

$$\left[\frac{\partial p}{\partial x}\right]_{i,j} = \frac{p_{i+1,j}-p_{i,j}}{\Delta x}.$$

The finite difference represention in the Y-direction of the conservation of linear momentum represented in the middle of the top edge of cell (i, j) can be given as

$$\left[\frac{\partial(v^2)}{\partial y}\right]_{i,j} = \frac{1}{\Delta y}\left(\left(\frac{v_{i,j}+v_{i,j+1}}{2}\right)^2 - \left(\frac{v_{i,j-1}+v_{i,j}}{2}\right)^2\right) + \alpha\frac{1}{\Delta y}\left(\frac{|v_{i,j}+v_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i,j}|}{2}\frac{(v_{i,j-1}-v_{i,j})}{2}\right),$$

$$\left[\frac{\partial(uv)}{\partial x}\right]_{i,j} = \frac{1}{\Delta x}\left(\frac{(u_{i,j}+u_{i,j+1})}{2}\frac{(v_{i,j}+v_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i-1,j+1})}{2}\frac{(v_{i-1,j}+v_{i,j})}{2}\right) + \alpha\frac{1}{\Delta x}\left(\frac{|u_{i,j}+u_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i-1,j+1}|}{2}\frac{(v_{i-1,j}-v_{i,j})}{2}\right),$$

$$\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} = \frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{(\Delta x)^2},$$

$$\left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j} = \frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{(\Delta y)^2},$$

$$\left[\frac{\partial p}{\partial y}\right]_{i,j} = \frac{p_{i,j+1}-p_{i,j}}{\Delta y}.$$

The terms in the equation for the conservation of mass are represented in the middle of cell (i,j)

$$\left[\frac{\partial u}{\partial x}\right]_{i,j} = \frac{u_{i,j}-u_{i-1,j}}{\Delta x}, \qquad \left[\frac{\partial v}{\partial y}\right]_{i,j} = \frac{v_{i,j}-v_{i,j-1}}{\Delta y}.$$

The parameter α can be chosen between 0 and 1. If $\alpha = 0$ the Euler forward in time, forward in space (FTFS) scheme is applied. This can lead to an oscillating behaviour in the solution, similar to the error growth in an Euler forward in time, backward in space (FTBS) formulation with $c = 2.0$, to overcome the problem α will be encreased until $\alpha = 1$ and the so called Donor-Cell-scheme is achieved. This scheme is very similar to the Marker and Cell method which is related to the volume tracking methods. In practice an intermediate scheme is often used $(0 < \alpha < 1)$ with

$$\alpha > \max\left(\left|\frac{u\Delta t}{\Delta x}\right|, \left|\frac{v\Delta t}{\Delta y}\right|\right).$$

The spacial and time discretisation of the Poisson equation is performed in NAST2D as following:

$$\frac{p^{(n+1)}_{i+1,j}-2p^{(n+1)}_{i,j}+p^{(n+1)}_{i-1,j}}{(\delta x)^2} + \frac{p^{(n+1)}_{i,j+1}-2p^{(n+1)}_{i,j}+p^{(n+1)}_{i,j-1}}{(\delta y)^2} + = \frac{1}{\Delta t}\left(\frac{F^{(n)}_{i,j}-F^{(n)}_{i-1,j}}{\delta x} + \frac{G^{(n)}_{i,j}-G^{(n)}_{i,j-1}}{\delta y}\right). \tag{A.1}$$

The discretisation scheme leads to the following stability conditions where the second and the third condition are the well known Courant-Friedrichs-Lewy conditions which are conditionally stable.

$$\frac{2}{Re}\Delta t < \frac{(\Delta x)^2(\Delta y)^2}{(\Delta x)^2 + (\Delta y)^2}, \quad |u_{\max}|\Delta t < \Delta x, \quad |v_{\max}|\Delta t < \Delta y.$$

A.2 Optimisation programme files

In order to simplify the data exchange between the MATLAB OPTIMIZATION TOOLBOX and the CFD-code NAST2D, file transfer has been used to exchange data and information. This method also allows the preservation of information even if one of the codes crashes.

In order to start an optimisation using MATLAB and NAST2D the following files have been necessary:

- A startup script, to set initial parameters of MATLAB and start the optimisation.
- A parameter file, setting the initial acceleration profile.
- A parameter file, setting the initial configuration for NAST2D.
- An optimisation function which starts NAST2D for the various parameter variations and calculates the violations of the constraints.

Within the startup script and the optimisation function several data backup calls have been implemented to track the iterative process of the optimisation and the improvements within the solution.

A.2.1 Optimisation startup script

```
%  Startup script file for parameter optimisation
%   The system and the bounds get fixed
%   c Harald Leonpacher 1997

echo off;

% Load initial value file

load init_XF.val

% Evaluation of the final time
```

```
tf = init_XF(length(init_XF(:,1)),1);

% Selection of n parametervalues from t_0 to t_f
% n is given by the length of the matrix 'init_XF'

teil = 120;
el = length(init_XF)-1;
u0 = [init_XF(3:el-2,2)]/teil;
ux = [init_XF(1,2) init_XF(el,2)];
uy = [init_XF(2:el-1,3)];

% Selection of n timeintervals from t_0 to t_f

ti_o = [0:(1/(length(init_XF(:,1))-4)):1];
ti = [0, 0.0000001, ti_o(2:length(ti_o)-1), 0.9999999, 1];

% Initial time

t0 = init_XF(1,1);

% Criteria for optimisation

options=foptions;
options(01)=1;
options(02)=0.001;
options(03)=0.001;
options(04)=1e-4;
options(13)=0;
options(14)=1000;
options(16)=0.0051;
options(17)=0.51;

% Selection of upper and lower bounds on the parameters
%   As an example, the acceleration is bounded by [+/- 500]
%   As an example, the time is bounded by [0.1 .. 1.4]

k  = length(u0(1,:));
tf = tf*ones(1,k);
u0 = [u0;tf];
l  = length(u0(:,1));
vlb=zeros(l-1,k);
vlb(:,1)=vlb(:,1)-(500/teil);
lb = ( 0.1 ) * ones(1,k);
vlb=[vlb;lb];
vub=[zeros(l-1,1)+(500/teil)] ;
ub = ( 1.4 ) * ones(1,k);
vub=[vub;ub];


% Optimisation

u = constr_h('opti_sch',u0,options,vlb,vub,[],t0,ti,uy,ux);
```

```
% Final calculation of the fluid motion based on the optimised
% parameter set

!run1 schwapp.end
```

A.2.2 Optimisation function

```
function [fout,gout] = opti_1(u0,t0,ti,uy,ux)

% Function to caculate the objective function and the bounds
% within the specific optimisation problem of sloshing

% c Harald Leonpacher 1997

% Some helpful values

l  = length(u0(:,1));
k  = length(u0(1,:));
teil = 120;

% Declaration and saving of the parameter value set for every
% single function evaluation within the optimisation

initxy=[[ti u0(1,1)]' ...
[ux(1);ux(1);u0(1:l-1,1)*teil;ux(2);ux(2);u0(1,1)] ...
[0;uy;0; u0(1,1)]];
save initxy.val initxy -ascii

% Start of the cart and fluid motion simulation to achieve
% output results for the boundary violations

!run schwapp.par

% Loading of the output results for the boundary violations

load maxlr.val
max_i = maxlr;

% Scaling of the sloshing boundary violation
% Linear penalty function

if maxlr(1)>1
      maxlr(1)=(maxlr(1)-1)*1000+maxlr(1);
end
if maxlr(2)>1
   maxlr(2)=(maxlr(2)-1)*1000+maxlr(2);
end
```

```
% Setting of the objective value

fout = u0(1);

% The constraint violations are evaluated

tol_ = 0.0002;      % to get inequality constraints only
dist = 0.9;         % Distance to travel in 10^(-1) [m]
gout(1) = (maxlr(1)-1) ;
gout(2) = (maxlr(2)-1) ;
gout(3) = (maxlr(3)-(dist+0.5*tol_));
gout(4) = ((dist-0.5*tol_)-maxlr(3));
gout(5) = maxlr(4)-(0.5*tol_) ;
gout(6) = -(0.5*tol_)-maxlr(4) ;
  gout(7) = (maxlr(8)-5.0) ;
% The velocity is not allowed to be
% greater than 0.5 [m/s]= 333 rpm;
% nondimensionless: u_oo=0.1 => boundary value 5.0

% Writing of the comprehensive output file
% tracking every function evaluation

fid=fopen('track_sp.dat','a');
 for co=1:1-1
   fprintf(fid,'%1.4f ',u0(co,1)*teil);
 end
fprintf(fid,'%1.4f .... %1.5f \n' ...
        ,fout,gout(1),gout(2),gout(3),gout(4),gout(5) ...
,gout(6),gout(7),max_i(1),max_i(2));
fclose(fid);
```

Appendix B

Relevant published papers

The research program was funded by the bavarian government and the bavarian research foundation and is part of the FORTWIHR research federation.

Within this federation several presentations have been carried out. In particular the author gave the following talks and presentations:

- *Cost Reduction using Modelling, Simulation and Optimisation.*, Presentation FH-München, 1996
- *Optimal Control of Sloshing Liquids.* 12th Conference on Variational Calculus, Optimal Control and Applications, Trassenheide, Usedom, 1996
- *Dynamic Warehouse Optimisation.*, Annual FORTWIHR-presentation, Erlangen, 1996
- *R&D Aspects in Modelling, Simulation, and Optimisation.*, 2. International Research Forum 'Market meets Science', München, 1997
- *Dynamic Warehouse Optimisation.*, Annual FORTWIHR-presentation, München, 1997
- *Optimal Control of Sloshing Liquids.* International Series of Numerical Analysis, ISNM Vol. 124, pp. 303-312, Birkhäuser, Basel, 1998
- *Modelling, Simulation and Optimisation of complex systems.*, FHM-Journal, Nr. 1.98, pp. 69-73, München, 1998
- *Simulation and Optimization of Logistic Processes Involving Sloshing Media.* In: Bungartz, H.-J., Durst, F., Zenger, Ch. (Eds.): High Performance Scientific and Engineering Computing, Springer, Berlin, 1999.