Multidisciplinary Design and Optimisation of Liquid Containers for Sloshing and Impact

by

Thomas Charles Kingsley

Submitted in partial fulfilment of the Degree of
Masters of Engineering (Mechanical Engineering)

In the Faculty of Engineering, the Built Environment and Information Technology

University of Pretoria

April 2005

Summary

Multidisciplinary Design and Optimisation of Liquid Containers for Sloshing and Impact

by

Thomas Charles Kingsley

Study Leader:	Prof. Ken Craig
Year:	2005
Department:	Department of Mechanical and Aeronautical Engineering

The purpose of this study is to perform an investigation of the numerical methods that may contribute to the design and analysis of liquid containers. The study examines several of these methods individually, namely Computational Fluid Dynamics (CFD) analysis of sloshing and Finite Element Methods (FEM) analysis of impact, to evaluate their contribution to the design cycle. Techniques that enhance the use of the various methods are presented and examined to demonstrate effectiveness. In the case of sloshing analysis, experimental tests performed add to the understanding of the phenomena at hand and qualifies the validity of the numerical method used (CFD). As a final contribution, the study presents a method of utilising impact analysis tools, FEM, and CFD in a Multidisciplinary Design Optimisation (MDO) environment. This is an introductory attempt at demonstrating a single coupled multidisciplinary method of designing liquid containers.

The results of the study demonstrate a number of valuable numerical techniques that may be used in the design of liquid containers. The presented Total Deviation Value (TDV) proves to be an effective single quantification of sloshing performance and the CFD tools used to determine the value demonstrate sufficient ability to reproduce the sloshing event itself. More advanced experimental facilities would provide a more in-depth understanding of the limitations of the CFD analysis. The use of numerical optimisation adds a valuable dimension to the use of numerical simulations. Significant design improvements are possible for several design variables without performing exhaustive studies and provide interesting information about design trends. Finally, the use of multiple disciplines, FEM and CFD, in conjunction with the available numerical optimisation routines offers a powerful multidisciplinary design tool that can be adapted to any base geometry and is capable of finding optimal trade offs between the two disciplines according to the designer's needs.

This study provides a platform for further investigations in the use and coupling of sloshing and impact analysis in the design of industrial liquid container applications.

Keywords: Liquid Sloshing, Multidisciplinary Optimisation (MDO), Fluid Structure Interaction (FSI), Computational Fluid Dynamics (CFD), Finite Element Methods (FEM), Mathematical Optimisation, Successive Response Surface Method (SRSM), Volume Of Fluids (VOF), Total Deviation Value (TDV), Impact Analysis.

Acknowledgements

The author would like to thank the following people for the contributions to the successful completion of this study:

Prof. Ken Craig
Mr De Kock
Mr Haarhoff
Mr Pretorius
Ms Angela Kingsley

Without the selfless assistance of these people the completion of this study would not have been possible.

Table of Contents

List of Figures and Tables

CHAPTER 1: Introduction

This dissertation documents the work done during a master's study entitled the multidisciplinary design and optimisation of liquid containers for sloshing and impact.

The advent of modern computing capabilities, their dramatic reduction in cost and their widespread integration into the modern engineering design office have changed the way the design process is approached by the modern engineer. Computer aided design packages, solid modelling tools, finite element method packages, computational fluid dynamics software, and more recently numerical optimisation software are just some of the highly technical software solutions provided to the modern engineer. Any one of these commercial software packages may be used in the design process and, in more recent history, may in fact be used in combination. The aim of this study is to evaluate some of these tools within the context of the design of a specific type of equipment, i.e., liquid containers.

Liquid containers are items that are found throughout the engineering design environment, across all engineering disciplines. The design of certain of these containers command high levels of research and development. Fuel containers for aeronautic and astronautic applications are examples that traditionally have enjoyed the pinnacle of design efforts in this field. It is largely thanks to many of the related high level research institutions that many of the above mentioned engineering design tools were developed. There are endless applications where many of these tools are tragically under utilised. In principle, the design challenges experienced in all liquid container applications are very similar, and this means that many of the advanced design techniques developed are universally applicable.

The aim of the study is to examine the available techniques in a generic sense as an exploratory evaluation of their corresponding contributions and effectiveness in the design of liquid containers. A quantitative measure of sloshing will be presented, and experimental tests will qualify the use of the numerical methods used to model sloshing. The development of methods for modelling sloshing by using Computational Fluid Dynamics (CFD) and the modelling of liquid container impact using Finite Element Methods (FEM) will form part of the study. The combination of numerical modelling techniques, like CFD and FEM, with a number of the available mathematical optimisation methods, like Response Surface Methods (RSM), will be presented and the results will be evaluated. As a final contribution, the use of CFD, FEM and mathematical optimisation will be combined in an automated cycle and presented as a tool with which to tackle the optimal design of a simple liquid container structure. The study as a whole is aimed at presenting the use of multiple mathematical design tools in the design of liquid containers. The work will collectively provide a platform from which future academic research may emanate in a subject field that suggests a number of possibilities for further development. The combination of multiple disciplines, optimisation, and multiphase unsteady free-surface behaviour is regarded as new work, in that it has not been studied, to the knowledge of the author, at an academic level or published in any known scientific journals.

Chapter 2 of the dissertation covers a literature study of work in the field of liquid container design and the understanding of transient liquid motion. The work done on liquid motion is mostly confined to linear oscillatory motion of shallow and deep waves. Other more historical methods of liquid motion representation are presented, including Equivalent Mechanical Systems. These simplified systems are analysed and their advantages and short comings identified. The chapter also provides a description of the various engineering design tools that are utilised in the study. The tools presented include the simulation packages used for Computational Fluid Dynamics and Finite Element Analysis and an overview of some of the more significant models included in these packages and how they may be utilised in liquid container design. An overview of experimental methods outline some of the criteria used in modern time to define the safety regulation that control the design of liquid containers used in the automotive environment. Finally the later sections of the chapter cover available mathematical optimisation techniques and how they may be used in a design process.

Chapter 3 documents the modelling of sloshing and the methods employed within the simulation tools utilised. The chapter covers automated methods of using the simulations tools so that they may later be incorporated within a mathematical optimisation routine. To gain confidence in the techniques chosen, several experimental verification methods are presented. The experimental results include both a qualitative analysis provided through free-surface images and a quantitative analysis of pressures in a liquid container in the time and frequency domain.

The optimisation of the liquid container for sloshing is presented in Chapter 4. To incorporate the Computational Fluid Dynamics modelling techniques into an inherently quantitative mathematical optimisation environment, a method for quantifying the level of sloshing in a liquid container is proposed. The behaviour of this quantification is analysed within the spectrum of sloshing phenomena expected. The bulk of the chapter provides documentation of a systematic comparison of the setup and results of the various optimisation techniques available. The strengths and weaknesses of the various methods are identified and further means of analysing the data available from an optimisation process are presented.

Mathematical optimisation for impact is covered in Chapter 5 and provides an overview of the Finite Element Analysis techniques used to simulate the fluid structure interaction that results in the deflection of the baffles in a liquid container. The ensuing stresses are used to formulate an automated optimisation routine that reduces the mass of the liquid container while adhering to predefined constraints on structural integrity.

Chapter 6 brings together a number of the methods investigated into a single optimisation routine. It covers the multidisciplinary optimisation of liquid containers for both liquid sloshing and structural integrity during impact. The chapter demonstrates the results one can expect while performing a multidisciplinary optimisation process. The idea is that one could adapt the methods utilised to any specific combination of geometry, input load curves and constraints, structural or otherwise, and the same principles will apply. It should for this reason be noted that the geometries and load curves have been simplified throughout this study so as to effectively demonstrate as many techniques and combinations as possible within the available time frames.

The dissertation is concluded with a chapter containing conclusions and suggestions for future work.

CHAPTER 2: Literature Study

2.1 Introduction

This section of the dissertation provides an overview of some of the major fields of interest either utilised or examined in this study. This chapter discusses some of the major publications made in the fields of interest, as well as contextualising them in terms of this study.

2.2 Sloshing

Sloshing is a phenomenon explained in the dictionary as "spill or splash copiously or clumsily, make a splashing sound". The event however has far reaching implications in the realms of engineering. Sloshing can be described as the motion of a fluid as it attempts to attain a state of equilibrium for the effective instantaneous acceleration (gravitational, translational, etc.) felt by the fluid. The momentum of the fluid and external loads on the fluid container will prevent this state of equilibrium. In turn, the motion of the fluid has a number of side effects. These include the acoustical affects due to the higher velocity motion and pressure fluctuations near the free surface, the impulsive loads the fluid may exert on the container or other structural bodies inside the container, and an effect on the dynamic stability of the container as a whole.

The phenomenon of sloshing has evoked the attention of scientific researchers since as early as the 1950s with the work of Graham and Rodriguez [1, 2] who studied the effects of fuel motion on airplane dynamics. In the 1960s the work was continued by the aerospace industry with work similar to that performed by Abramson of NASA [3]. Since then numerous other publications have been made, with wide-reaching applications [4-8, 14-18] (See section 2.4). The major fields of interest within the context of sloshing include fuel tanks of almost all vehicle types, liquid transport containers, seismic and wind induced oscillations of tall buildings, aerospace, maritime, liquefied natural gas (LNG) tanks, etc. More recently the source of many of these investigations is the study of numerical techniques for solving the physics involved, in a hope that such techniques could assist in the design process of liquid containers.

Sloshing is often analysed in a simpler form where no overturn takes place, that is to say when the free surface stays intact. Assumptions like incompressibility, irrotational flow, inviscid, no ambient velocity, two-dimensional, and small amplitudes allow for a simplified analysis via linear wave theory [9]. The theory was first proposed by Lagrange (1776) and Airy (1845) and provides some insight into the behaviour of waves. The solution is then further subdivided into shallow and deep waves. Figure 2.1 below shows the general form of a wave, where C is the wave speed, h is the fluid level, L is the wave length, and η is the free-surface deviation from the mean water line.

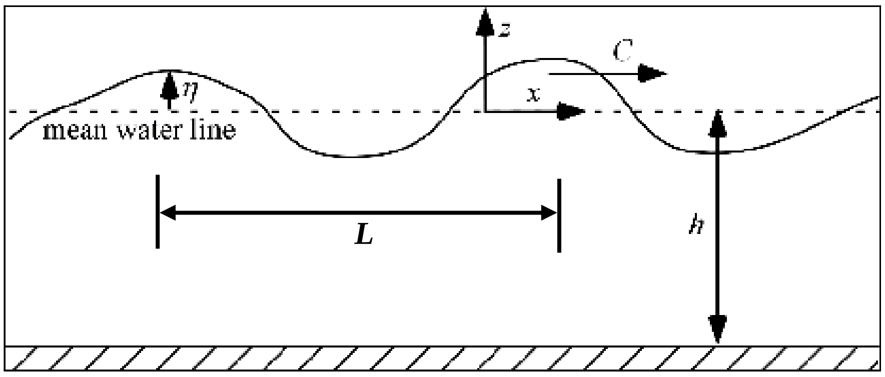


Figure 2.1: General wave form [9]

Mathematically, the difference between shallow and deep waves manifests as a wave-speed dependence on the wave number (k). For deep-wave theory, the wave speed is dependent on the wave number ($h/L > 0.5$) and the motion is dispersive. Conversely, when the wave speed displays independence from the wave number, usually for $h/L <$ 0.05, we refer to this as shallow wave theory (non-dispersive). The wave number is simply the reciprocal of the wave length (L). Equation 2.1 below shows the actual relation for deep waves, while equation 2.2 shows that which applies to shallow waves for the wave speed.

$$C = \sqrt{\frac{g}{k}} \tag{2.1}$$

$$C = \sqrt{gh} \tag{2.2}$$

Visually one can see the difference between the behaviour of particles in “deep water” in Figure 2.2 below, a picture of particle trajectories in plane periodic water waves. The waves are travelling across the image with no reflection. The particle trajectories can be seen to be circular at the free surface and flattened toward the bottom. The particles midway between the free surface and the bottom are intuitively more elliptical in shape as the physical constraint of the bottom has more effect. Section 2.4 provides more detail on linear wave theory.

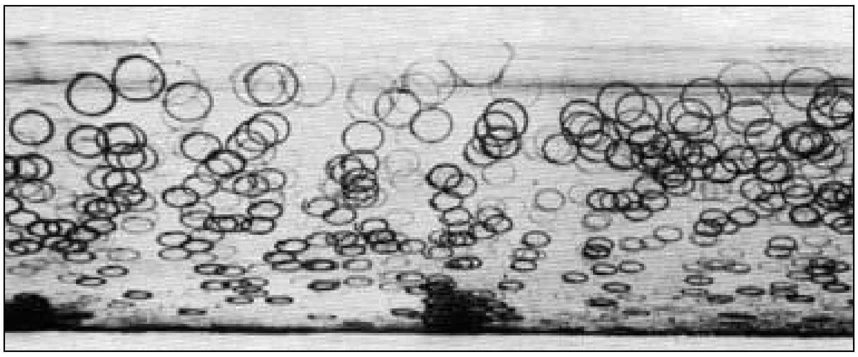


Figure 2.2: Particle Trajectories in plane periodic water waves [10].

Reflected waves on the other hand behave somewhat differently. Figure 2.3 below shows the particle trajectories for a pure standing wave, 100% reflection. The trajectories now show the streamlines for pure standing waves. This is quite similar in behaviour to the oscillation of a liquid in a tank with no baffles.

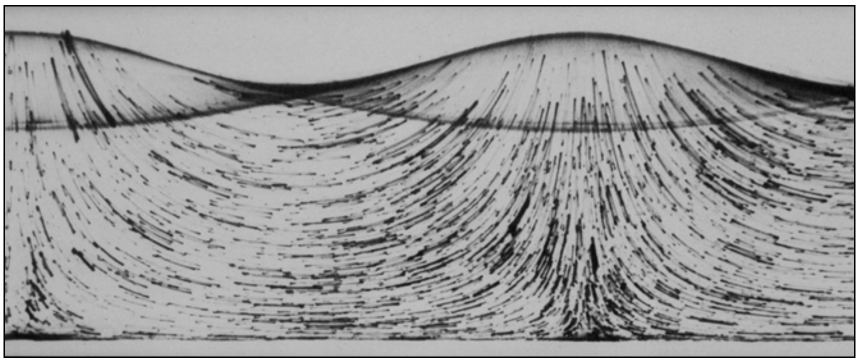


Figure 2.3: Particle trajectories in pure standing waves [10]

Often in cases of sloshing in containers the behaviour of the waves is somewhat non-linear. Figure 2.4 below (extracted from the experiments in chapter 3) shows a clear illustration of such an instance. Non-linear wave behaviour with turnover is a complex phenomenon and is not easily represented mathematically, but section 2.4 provides a further discussion of some methods used. This behaviour is typical of sloshing of fuel in fuel tanks, and other containers that may experience similar accelerations. The acoustical affects felt during sloshing are also associated with this level of fluid motion, something which is of interest to vehicle manufacturers due to the undesirability of driver-experienced noise levels. This forms part of the field of Noise, Vibration and Harshness (NVH), and an ongoing effort to make cars run quieter and more smoothly.

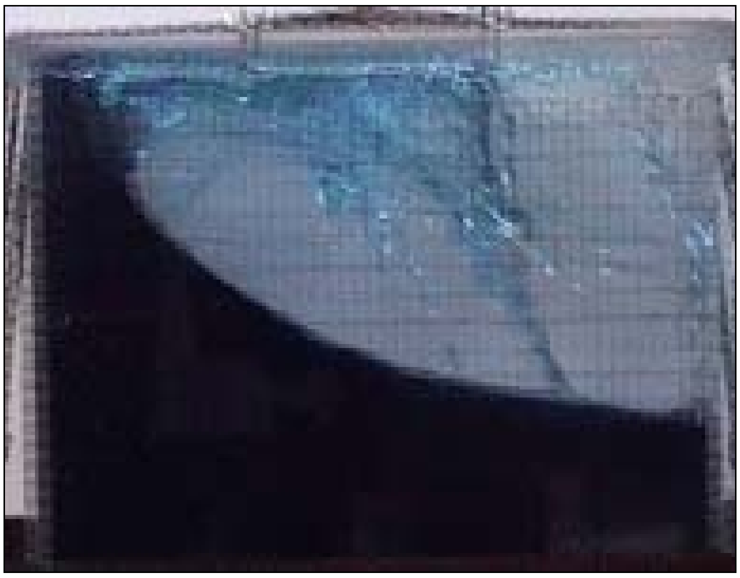


Figure 2.4: Non-linear sloshing of water in a rectangular tank

2.3 Fluid-Structure Interaction

Fluid-Structure Interaction (FSI) is a field which refers to the coupling of unsteady fluid flow and structural deformation. This is ideally a two-way coupling of pressure and deflection. A number of interesting applications have stimulated research in this field. Typical fields of interest would be airbag modelling, fuel tank sloshing, heart valve modelling, helicopter crash landings, etc. The important point, and the reason for the initiation of the FSI field of study, is that the fluid mechanics may affect and be affected by the structural mechanics, and vice versa. We might consider the case of an aeroplane wing. Once air starts to pass over a wing it typically produces some amount of lift, this will cause the wing to deflect and in turn alter the properties of the wing and the amount of lift it produces. The cycle may be continuous (e.g. flutter) or it may settle to some stable point of equilibrium. Other instances of FSI might be where the stress in a structural component may be induced by the fluid that surrounds it. Clearly in this case the coupling of the fluid's pressures and the motion of the structure is essential. Figure 2.5 below shows images of both the physical testing of airbags with crash test dummies as well as the simulations of the same event in a FSI-capable software package. One can appreciate that the cost effectiveness of being able to simulate the event with computer software would evoke much interest from vehicle manufacturers.

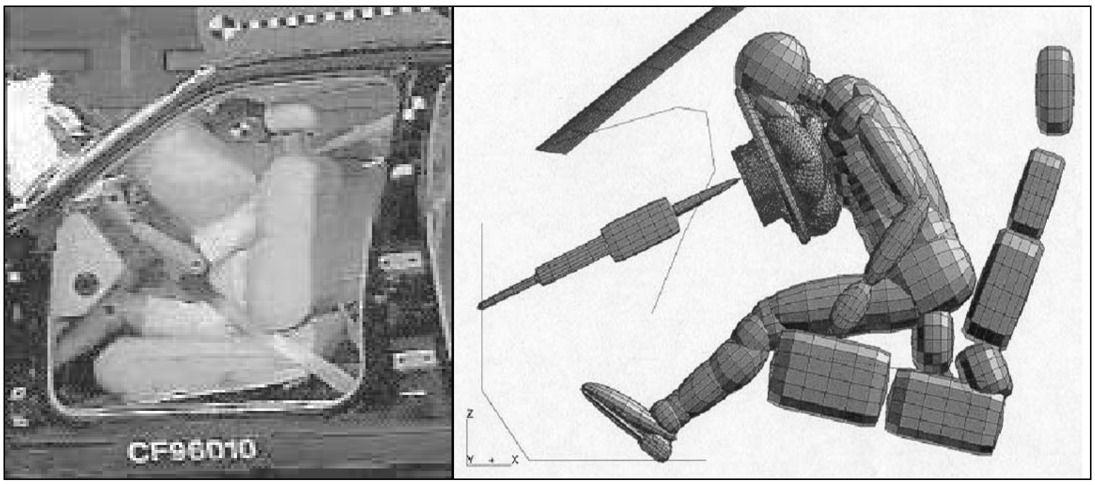


Figure 2.5: Airbag/Crash testing and simulation [11,12]

Section 2.4.4 gives a brief description of the mathematical methods involved in FSI and the suitability of certain methods.

2.4 Mathematical modelling

Mathematical modelling in general is the representation of some physical event through mathematical equations. Due to the fact that most interesting physical events are very difficult to model exactly, i.e., that the equations provide an exact representation of the reality, many possible mathematical representations may exist for any given event. In general, this implies a trade-off between equation complexity and accuracy. However, innovations in a field where mathematical modelling is concerned may lead to simpler equations giving similar or even improved results. This trend is usually initiated when more complex governing equations are simplified for very specialised cases of the broader physical event type. The use of specialised turbulence models for external flow over a wing is one such example.

The motivation for the use of mathematical models stems from a need to understand trends without having to reconstruct the event. Furthermore, if we have reasonable confidence in the accuracy of the model, we may use it as a method with which to analyse the event itself. This may in turn form part of a design cycle and, more applicably, part of a mathematical optimisation cycle (Mathematical optimisation is further discussed in section 2.6). One must however always keep in mind that the success of these techniques is strongly dependent on the accuracy of the model.

The remainder of this section introduces some of the mathematical techniques that have traditionally been used in the modelling of sloshing events or events that are similar in physics to sloshing.

2.4.1 Linear Wave Theory

Linear wave theory, or Airy's wave theory, is one of the first types of mathematical modelling used to analyse wave motion [9]. Briefly introduced in section 2.2 it provides some insight into wave motion at a relatively simple level. The mathematical theory is based on the governing equation of continuity (equation 2.3) in two dimensions and potential flow assumptions. The velocity potential ϕ is defined as in equation 2.4 - 2.5, and the combined result gives the Laplacian form of the continuity equation (equation 2.6), for which well-established standard solutions exist. Again, all equations are defined within the coordinate system described in Figure 2.1 above.

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0 \tag{2.3}$$

$$u(x,z,t) = \frac{\partial \phi(x,z,t)}{\partial x} \tag{2.4}$$

$$w(x,z,t) = \frac{\partial \phi(x,z,t)}{\partial z} \tag{2.5}$$

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial z^2} = 0 \tag{2.6}$$

The simple case of linear pure progressive wave motion lends itself to a solution by separation of variables. Equation 2.7 below represents the assumed sinusoidal form of the free-surface position $\eta(x,t)$. After all boundary conditions are simplified and applied under the assumption that the wave amplitude ε is small, i.e., all $d\phi/dx$ product terms are neglected, we obtain a non-dimensional solution as in equation 2.8 below. In this equation C is the wave speed, ε is the amplitude, L is the wavelength and h is the mean free-surface level. Equation 2.7 is valid for $\varepsilon << h$ and $\varepsilon << L$.

$$\eta(x,t) = \varepsilon.\sin\frac{2\pi}{L}(x - Ct) \tag{2.7}$$

$$\frac{C^2}{gh} = \frac{L}{2\pi h}\tanh\frac{2\pi h}{L} \tag{2.8}$$

As discussed in section 2.2, two possible assumptions can be made at this point. The two possibilities are either *h<<L* or only *L<<h,* which intuitively would describe a wave in shallow water and deep water respectively. If we consider the shallow water assumption, then *2πh/L* in equation 2.8 will become small and the *tanh* term will approach the value of this ratio. We then obtain the relation in equation 2.9. The deep water assumption will conversely result in the *tanh* term approaching unity, which will result in the relation shown in equation 2.10.

$$\frac{C^2}{gh} = 1 \tag{2.9}$$

$$\frac{C^2}{gh} = \frac{L}{2\pi h} \tag{2.10}$$

Graphically, the difference can be seen in Figure 2.6 below, which shows the relationship between wave propagation speed and wave length in non-dimensional terms.

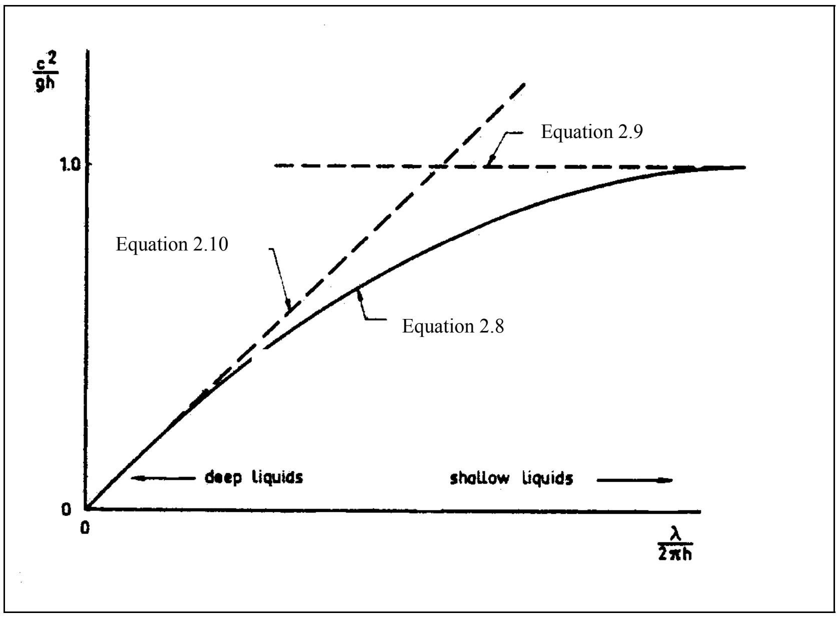


Figure 2.6: Relation between propagation speed and wavelength [13]

However, some assumptions made early in this formulation result in inaccuracies in these relations. For example, very closely packed waves (short wavelength) exhibit high propagation speeds and as a result are quickly dissipated. Figure 2.7 below shows a more realistic relation near this limit where the waves are termed 'capillary' waves.

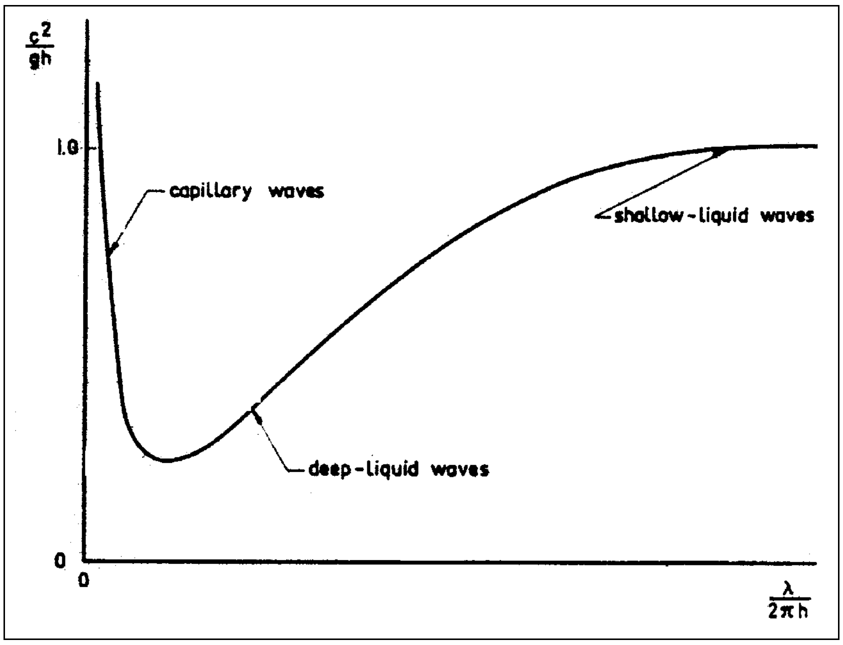


Figure 2.7: Dissipation effect of short wavelengths on wave propagation speed [13]

Linear wave theory for a 3-dimensional liquid container yields equation 2.11 below which represents the nth-mode oscillation frequency 'ω_n' in a container of length 'a' and fluid height 'h' [13]. Note the independence on tank width at this stage.

$$\omega_n^2 = \frac{n\pi g}{a}\tanh\left(\frac{n\pi h}{a}\right) \tag{2.11}$$

The linear wave theory equations described above have been expanded in several studies, e.g., Warnitchai and Pinhaew [14] expanded the theory in an attempt to develop an equation for the 1st mode frequency in a container whilst considering flow damping devices or Tuned Liquid Dampers (TLD). The study yielded equation 2.12 below, which is an adaptation of equation 2.11 above for n=1 with specific consideration of the damping effect of a single rigid cylinder standing vertically in the centre of the tank. C is the coefficient of inertia of the cylinder, A is the wave amplitude, and b is the width of the 3-D tank.

$$\omega_1^2 = \frac{\pi g}{a}\tanh\left(\frac{\pi h}{a}\right)\left(1+\frac{A}{ab}C\left(1+{2\pi\frac{h}{a}}\Big/{\sinh\left(2\pi\frac{h}{a}\right)}\right)\right)^{-1} \tag{2.12}$$

The effect is however small due to the relatively small magnitude of the *A/ab* term. Equation 2.11 is however still of interest when analysing sloshing in a tank without damping devices. Modi and Munshi [15] examined in detail the effect of liquid depth and modal frequency on their unique dampers, making the point that static tuned liquid dampers only operate optimally at a single depth and wave speed. Modal analysis received further attention by Schotte and Ohayon [16] with their demonstration of the apparent gravity approach for accelerating bodies.

Although linear solutions can work well for simple cases, they are limiting. Other solutions similar to the one described exist for several very simple flow problems, but simple cases like this do not represent the majority of engineering problems. In the 1970s, Faltinson [17] extended the field to approximate theoretical forms for inviscid sloshing in moving tanks. More recently, Frandsen [18] presented the combination of potential flow theory with physical-to-computational transformation techniques to simplify the tracking of the free surface. However, alternative methods of solving fluid sloshing need to be utilised in most cases that are not highly simplified. This necessity has led to the development of methods like Computational Fluid Dynamics (CFD), as discussed in section 2.4.3.

2.4.2 Equivalent Mechanical Systems

Equivalent Mechanical Systems (EMS) are mechanical systems that respond in the same way as the original/actual systems would. Sometimes referred to as lumped parameter models (LPM), these systems would typically consist of a series of masses, springs, and dampers that when excited will exhibit similar feedback forces. These models are a legacy of a time when computational power was restricted to a pen and paper, and are often restricted by small angle assumptions. Models of this nature do however still exist [1,2,4] and may still give some level of insight into a problem. It must also be said that even today, computational power can still become a factor when considering the cost of solving complex flow or structural problems.

A very simple mechanical model was introduced in 1951 by Graham [2] who represented the fuel in the tank as an equivalent pendulum. This analogy has more recently been investigated by Fernando [4] where some experimental work was performed to determine the spectrum of validity. Figure 2.8 shows a schematic representation of the pendulum analogy.

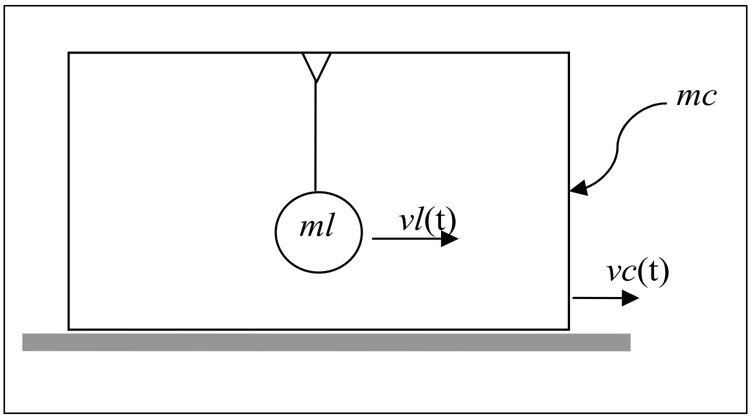


Figure 2.8: Schematic of pendulum analogy

The mass of the liquid in the container is represented by the mass of the pendulum (*ml*), while the pendulum's velocity (v*l*) represents the velocity of the liquids centre of mass. The container retains its mass (*mc*) and velocity (v*c*). Fernando's study [4] centres on the motion of the container relative to the surface it is resting on. The motion is influenced by the behaviour of the fluid in the container. By comparing experimental data with the analytical model, he shows the regimes within which the model exhibits acceptable performance. As one would expect, the model's spectrum of validity is limited to smaller excitation forces, since non-linear free-surface behaviour cannot be accounted for by a pendulum. The model does however provide some interesting insight into level of sloshing as a function of the fluid level. Continuing with the analogy as introduced above, Fernando considers the frictional force (F_f), and the force on the container due to the liquid sloshing (F_c). Figure 2.9 below illustrates the variation in forces F_f and F_c with respect to the mass of fluid in the container m_l. F_{f1} and F_{f2} represent two frictional force curves corresponding to the two container masses (mc_1 and mc_2). For the case represented by the F_{f1} curve, the tank will oscillate over the interval $m_{l1} < m_l < m_{l2}$, i.e., when $F_c > F_f$. When $F_c < F_f$, as at points A and D, there will be no relative motion between the container and the surface. The shaded area indicates where motion will occur for all container masses greater than mc_1. Some of the major observations that can be made include the non-linear behaviour of F_c. This can be explained by the fact that not all the fluid is involved in the oscillation. For low fill levels, fluid mass below point B, the force F_c is minimal due to the lack of fluid in the tank, i.e., inertial effects are low. For high fill levels, fluid masses above point C, insufficient room (gas) is in fact available in the tank for complete oscillation, and when the container is full (m_{l2}) no oscillation takes place at all. A further observation is that the point of zero gradient of *Fc*(*ml*) point E, or maximum *Fc* is in fact not at the 50% fill point. The data in fact suggests a value between 65 to 70%. This value will prove important at a later point in this study.

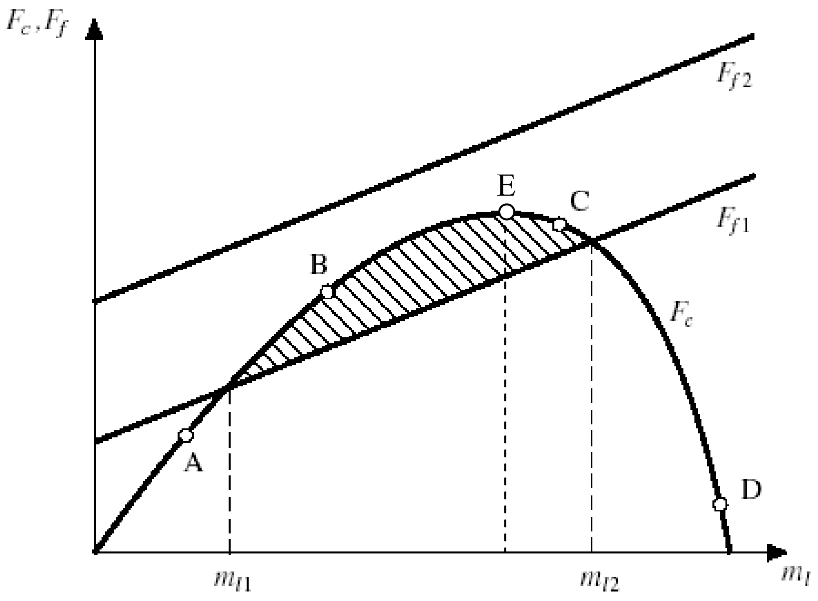


Figure 2.9: The relation between Fc/Ff and ml [4]

A more complex LPM was introduced by Graham and Rodriguez [1] in 1952. The model was developed in an attempt to analyse the forces induced on an aeroplane by the motion of the fuel in the fuel tanks during flight. Figure 2.10 below shows a schematic of this LPM. The various masses are used to model the various wave harmonics.

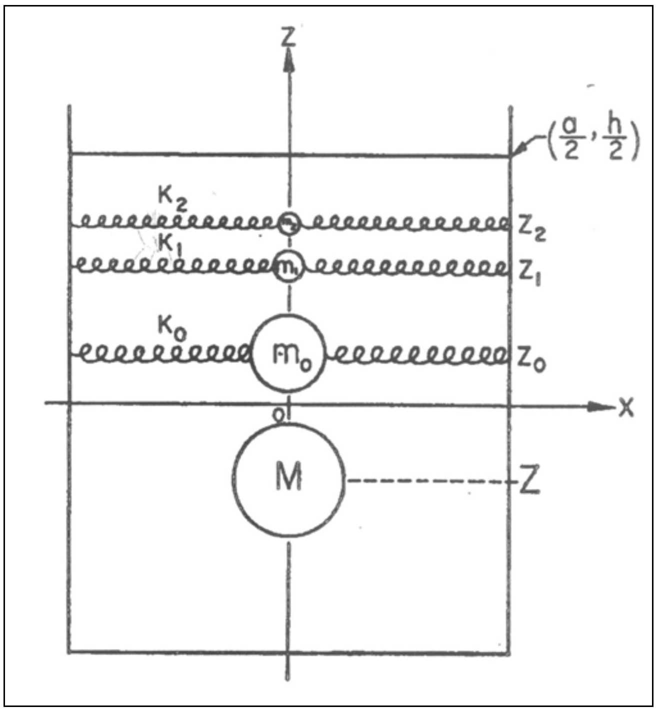


Figure 2.10: Graham and Rodriguez's LPM [1]

Graham and Rodriguez’s model is restricted to small accelerations compared to gravity and only very small angular displacements. In fact, due to the application, the model is restricted to three basic tank motions; x-axis translation, pitching in the x-z plain, and yaw about the z-axis (figure 2.10). However, even when considering significant simplifications, the equations that represent the model are very involved and the author suggests that a model that would respond to all tank motions “cannot” be constructed. One major drawback of the system is that no damping exists, however the location of natural frequencies can be extracted. Figure 2.11 [1] below shows a force ratio (force produced by the mechanical system due to a horizontal oscillation divided by the force that would be produced if the fuel were a solid) to dimensionless frequency for a 0.25 aspect ratio tank. This is the typical format of the data one would be able to extract from the model and clearly shows natural frequencies. This can ultimately be used in the design of the aeroplane’s fuel tank to ensure that certain natural frequencies will not adversely interfere with the behaviour of the plane.

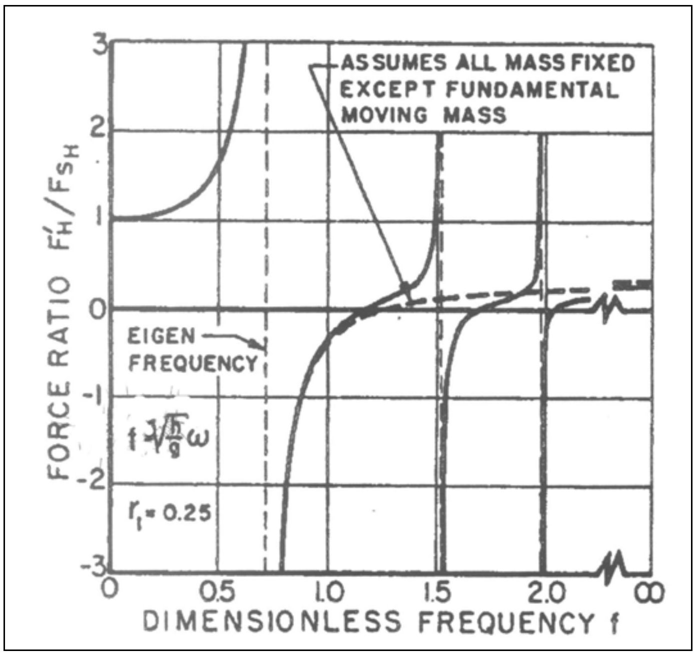


Figure 2.11: Force ratio vs. dimensionless frequency for Graham and Rodriguez's LPM [1]

2.4.3 Navier-Stokes Methods

Computational Fluid Dynamics (CFD) is the numerical solution of the discritised form of the partial differential equations that govern fluid flow (Navier-Stokes equations). The three principle governing equations or major conservation laws are momentum, mass, and energy conservation. CFD is now a widely used tool for solving and visualising fluid flow processes in the engineering world. A number of CFD codes are commercially available and for the purposes of the study, due to availability, Fluent v6.x, as distributed by Fluent Incorporated [19], is utilised. A brief explanation of the equations solved by CFD codes will follow.

2.4.3.1 Conservation of Mass

The mass conservation or continuity equation solved is as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho \vec{V}) = S_m \qquad (2.13)$$

where ρ is the density, t is time, and $\vec{V}$ the velocity vector. S_m is a mass source term, e.g., the mass added by one phase from another phase, e.g., during the vaporisation of liquid droplets. Equation 2.13 is a general form of the continuity equation and is valid for both compressible and incompressible flow.

2.4.3.2 Conservation of Momentum

The conservation of momentum in the *i*-th direction, assuming a non-accelerating reference frame, is given by:

$$\frac{\partial}{\partial t}(\rho V_i) + \frac{\partial}{\partial x_i}(\rho V_i V_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i + F_i \qquad (2.14)$$

where τ_{ij} is the stress tensor given by equation 2.15, g is the component of gravitational acceleration, p is the static pressure, and F_i is an external body force and behaves much like the gravitational body force, but allows for user-defined source terms, e.g., momentum source. Momentum source has the units of kg/m^{2}s^2 and is the multiple of the density of a specific mesh cell and the instantaneous acceleration. This term will prove very useful in the implementation of load curves in this study. The stress tensor is as follows:

$$\tau_{ij} = [\mu(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i})] - \frac{2}{3}\mu\frac{\partial V_l}{\partial x_l}\delta_{ij} \tag{2.15}$$

where μ is the molecular viscosity and the term after the negative sign is an effect due to volume dilation (size change). The Kronecker delta function δ_{ij} represents a value of zero if $i \neq j$ or a value of one if $i=j$. The momentum conservation equations are also known as the Navier-Stokes equations.

To take into account the effect of turbulence fluctuations, it is customary to perform a time averaging of equation 2.14, resulting in the Reynolds-Averaged Navier-Stokes equations. The equations contain turbulent stress terms that are modelled (See section 2.4.3.4 below).

2.4.3.3 Conservation of Energy Equation

The energy conservation is solved in terms of conservation of enthalpy, h, given by:

$$h = \sum_i m_i h_i \tag{2.16}$$

where

$$h_i = \int_{Tref}^{T} c_{p,i}\, dT \tag{2.17}$$

where m is mass, c_p is specific heat, and T_{ref} is a reference temperature. The energy equations can be written in terms of h as in equation 2.18 below.

$$\frac{\partial}{\partial t}(\rho h)+\frac{\partial}{\partial x_i}(\rho u_i h)=\frac{\partial}{\partial x_i}(k\frac{\partial T}{\partial x_i})-\frac{\partial}{\partial x_i}\sum_{j1} h_{j1}J_{j1}+\frac{\partial p}{\partial t}+u_i\frac{\partial p}{\partial x_i}+\tau_{ij}\frac{\partial u_i}{\partial x_j}+S_h \quad (2.18)$$

where T is the temperature, τ_{ij} is the viscous stress tensor, J_{j1} is the flux of species $j1$, and k is the mixture thermal conductivity. S_h is a source term that includes sources like energy due to chemical reaction, radiation, and heat exchange. The second last term is a viscous heating term that is an optional term that should be activated when assuming compressible flow.

2.4.3.4 Turbulence Modelling

This study utilises Eddy Viscosity Models (EVM) to close the Reynolds-Averaged Navier- Stokes equations. An example of this is the k-ε turbulence model that assumes proportionality between Reynolds stresses in the fluid and mean velocity gradients. Although the form of the momentum equations remain the same, the viscosity term becomes an effective viscosity μ_{eff}, and is determined by the sum of the molecular viscosity μ and a turbulent viscosity μ_t. The turbulent viscosity can be determined from equation 2.19 below.

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \quad (2.19)$$

where k, is the turbulent kinetic energy, ε the turbulent dissipation rate and C_μ is an empirically-derived constant of proportionality (default value equals 0.09 in Fluent).

A further example of an EVM is Wilcox's k-ω turbulence model [20]. This is a relatively new model, more capable of handling higher curvature flows than the standard k-ε model.

In the implementation of the *k*-*ε* turbulence model and other EVMs, the ‘near-wall functions’ model near-wall turbulence. This method prescribes a certain shape or function that describes the near-wall velocity profiles but satisfies no-slip conditions. This model proves very effective provided a certain y^+ range is adhered to. y^+ is a parameter that describes the size of a grid volume relative to the assumed velocity profile, in non-dimensional terms and can be compared to a wall Reynolds number. It is generally used to establish the correctness of near-wall grid density according to the requirements stipulated for wall functions.

2.4.3.5 Volume of Fluid Method

A further model that needs to be discussed, due to its relevance to this study, is the multiphase volume of fluids method (VOF). The model was originally developed by Hirt, et al. [21]. This model is used for the tracking of the interface between phases. The tracking is done via the solution of a continuity equation as shown in equation 2.20 below.

$$\frac{\partial \alpha_q}{\partial t} + u_i \frac{\partial \alpha}{\partial x_i} = S_{\alpha q} \tag{2.20}$$

where α is the volume fraction of the q-th phase, and the source term on the right-hand side allows for the use of cavitation models. Intuitively, the following constraint will apply for n phases:

$$\sum_{q=1}^{n} \alpha_q = 1 \tag{2.21}$$

When determining the properties in each control volume, an equation of the form shown in 2.22 below is solved, where A is a typical property like density or viscosity.

$$A = \sum \alpha_q A_q \tag{2.22}$$

The VOF model is a simple and efficient model and works well for the tracking of large free surfaces, as opposed to mixing and dispersing flow situations [22]. Some current restrictions of this model include not being compatible with Large Eddy Simulations (LES), or species mixing and reacting flows. Although other methods exist for the tracking of free surfaces it has been shown through studies like those performed by Cariou and Casella [6] that the VOF model is both the most efficient and most commonly used method.

Many years of development and verification have led to the point that modern day CFD codes are considered to be reliable engineering design tools, used even for standardisation and pre-release testing in the aerodynamics industry. However this does not mean that experimental verification is not necessary. The nature of the technique allows for a lot of adjustments in solver setup, a factor which can influence both the speed and accuracy of the solution.

2.4.4 Structural Finite Element Methods

Finite Element Methods (FEM) has become a powerful and widely used method for the numerical analysis of engineering problems. Although the method is often confined to the analysis of structures, more complex problems like fluid flow or magnetic fluxes can also be tackled with this method. The basic approach involves the discretisation of a potentially complex region that defines the continuum, into simple geometric shapes called finite elements. The properties of the appropriate material are considered in conjunction with the governing equations and expressed in terms of unknown values at the element corners or nodes. These unknowns are in turn determined by the loading and constraints that apply to that portion of the continuum. Solution of the resulting equations will give us an approximation to the behaviour of the continuum, with results manifesting in the form of stresses and strains [23].

Of more direct interest in this study are the models that are used when analysing Fluid-Structure Interaction (FSI) problems. Typical instances of computational FSI are in aero-elastics, where flow over an elastic aerofoil or oscillating cylindrical objects are solved, or in the bio-mechanics field where elastic behaviour of micro-pumps or artificial valve membranes may be of interest. FSI in general is the coupling of fluids and structures and can be done in either a partitioned solution or in a single code. There are advantages and disadvantages in both techniques. The most significant point in the development of these methods is one of accuracy versus performance. This is in part due to the variation in formulation of the fluids (Eulerian) and structures (Lagragian) equations, and thus the treatment of a moving boundary in a fluid domain. It is accepted that partitioned solutions exhibit the most accurate results, since separate yet well-established solution techniques can be used for each sub-problem. It does however mean that each sub-problem is solved separately and coupling data are exchanged at the end of each iteration or time step. This in turn translates to quite a computationally expensive technique, and from a practical perspective requires the use of two computational solution codes, that can become financially expensive. A good example of the use of partitioned techniques can be seen in Matties and Steindorf [24]. An alternate option is one where both continua are solved simultaneously and linked within a single solution code. An example of this approach is the Multi-Material Arbitrary Lagragian Eulerian (ALE [25]) formulation that can be seen in a commercial capacity in LS-DYNA [26]. An example of the use of this technique can be seen in Le Sourne, et al. [27]. Even when solving both problems in one code, a coupling formulation must still exist. The two coupling techniques that are in use in LS-DYNA are the Constraint-based formulation and the Penalty-based formulation [28]. Although these are not the only coupling formulations that exist (other methods include Nitsche's method [29]) they are well established and accepted within their suggested applications. The constraint-based formulation is an algorithm that alters the velocities of the nodes of the solid and shell elements implicitly and forces them to follow each other. In this study, the solid elements are used to model the fluid, while the shell elements are used to model the container structures. The method attempts to conserve momentum but not energy, and is regarded as quite stable. The Penalty-based formulation on the other hand applies nodal forces explicitly by tracking the relative motion of a given point. The method conserves energy but is not as stable as the constraint-based formulation.

The ALE method allows for the use of both Eulerian meshes and Lagrangian structures within the same model simultaneously. This is particularly useful in this study where a simple crash analysis of a fuel tank requires Lagragian shell elements for the baffles and a rigid body for the tank that will "contain" the multi-material Eulerian mesh for modelling the partially-filled container. More detail on the modelling procedure used will be given in section 5.2.

2.5 Experimental Methods

In this section, two fields of experimental methods are discussed. The first group covers those methods that are used to verify the integrity of products according to certain industrial safety standards. Another group of experimental methods is intended to assist in the verification of the validity of numerical models. Both groups are very important and unlikely to disappear in the near future, but high costs are often associated with these techniques.

Vehicle safety is one of the foremost issues in motor vehicle manufacture, primarily because people demand safer vehicles but also because certain industrial safety standards must be met before a vehicle may be sold. The National Highway Traffic Safety Administration (NHTSA) has been issuing Federal Motor Vehicle Safety Standards (FMVSS) since 1967 to which motor vehicle manufacturers in the United States must conform and certify compliance. NHTSA crashes vehicles from many manufacturers every year in an attempt to improve the safety standards themselves. Car manufacturers in turn crash up to 100 of the same vehicle to improve their levels of safety. Since motor manufacturers are well aware of these standards, very few vehicles ever fail the FMVSS tests. This prompted NHTSA to develop the New Car Assessment Program (NCAP). FMVSS basically involves crashing a vehicle at 30mph and confirming compliance with all safety regulations, while NCAP involves crashing the vehicle at 35mph and determining the level of safety in the vehicle. NCAP results provide a star rating for a vehicle that is based on the probably of the driver or passenger sustaining serious injury. The crashes involve both frontal and side impacts and examine criteria like Head Injury Criteria (HIC), Chest deceleration, Femur load, Thoracic Trauma Index (TTI), and Lateral Pelvis Acceleration (LPA). The FMVSS are however far more extensive, and of particular interest is FMVSS 301, or the Fuel System Integrity standard. The standard involves three crash scenarios (front, rear, and side) to be covered by two future standards FMVSS 208 and FMVSS 214. After the crash event, the vehicle's fuel tank is inspected for integrity and fuel cut-off systems are evaluated. The actual regulation is as follows: FMVSS 301 [11] *"In the frontal impact test, a vehicle is driven forward into a fixed barrier at 48 km/h (30 mph), while in the side impact test, a 1,814 kg (4,000 lb) barrier moving at 32 km/h (20 mph) is guided into the side of a stationary vehicle, and in the rear impact test, a 1,814 kg (4,000 lb) barrier moving at 48 km/h (30 mph) is guided into the rear of a stationary vehicle. The standard limits fuel spillage from crash-tested vehicles to 28 grams (1 ounce) by weight during the time period beginning with the start of the impact and ending with the cessation of vehicle motion and to a total of 142 grams (5 ounces) by weight during the 5-minute period beginning with the cessation of motion. During the 25-minute period beginning with the end of the 5-minute period, fuel spillage during any 1-minute interval is limited to 28 grams (1 ounce) by weight."* Statistically fuel system integrity is of critical importance, according to the US Fatality Analysis Reporting System (FARS) in 1998, four percent (1,411) of light vehicle occupant fatalities occurred in crashes involving fire [11]. For all the above reasons experimental testing of vehicles, their components and of potential occupants (in the form of crash-test dummies) is of great importance in the vehicle design process.

The second group of experimental methods adopted by engineers is those used to validate mathematical models. With the increased use of Finite Element (FE) methods and Computational Fluid Dynamics (CFD) codes in the design process of vehicles and many other engineering structures and machinery, it is always important to maintain a reasonable level of understanding of the capabilities and limitations of the codes used. FE codes are typically verified by comparing stress, deflection, or acceleration data. An example of one such comparison can be seen in Marzougui, et al. [30], where acceleration data, both raw and processed (HIC, etc.), are compared for both a full scale crash test and the LS-DYNA model that is presented. Typically the author will discuss the results with the intention of describing pitfalls and successes, followed by suggestions for improvement. CFD codes on the other hand are typically verified by comparing pressure, velocity, or temperature data. An example of one such comparison can be seen in an application brief presented by Fluent.Inc [19], Hadzic, et al. [31], where pressure data are compared for both an experimental setup as well as the presented model. A further method of comparison can also be seen in this particular study, one which is usually specific to transient analyses, and that is a visual comparison. In this study the behaviour of the free surface with respect to time is compared. It is interesting to note that this brief is in fact presented by the company that provides the CFD software package. The excellent agreement between experimental and simulated results is clearly a good marketing point.

2.6 Mathematical Optimisation

As this study combines CFD and mathematical optimisation, this section describes some of the methods and terms used in the field of mathematical optimisation.

Optimisation is a tool used in many engineering fields, both consciously and subconsciously. The most common form of design optimisation is usually based on a trial-and-error method and requires large amounts of experience and resources if any level of frequent success is to be achieved. If however the performance criteria in question can be quantified, the more precise method of mathematical optimisation may be employed. A branch of this method is numerical design optimisation that further involves the numerical modelling of the design to quantitatively evaluate the performance of the design. The flow chart shown below in Figure 2.12 provides an overview of the basic mathematical optimisation method.

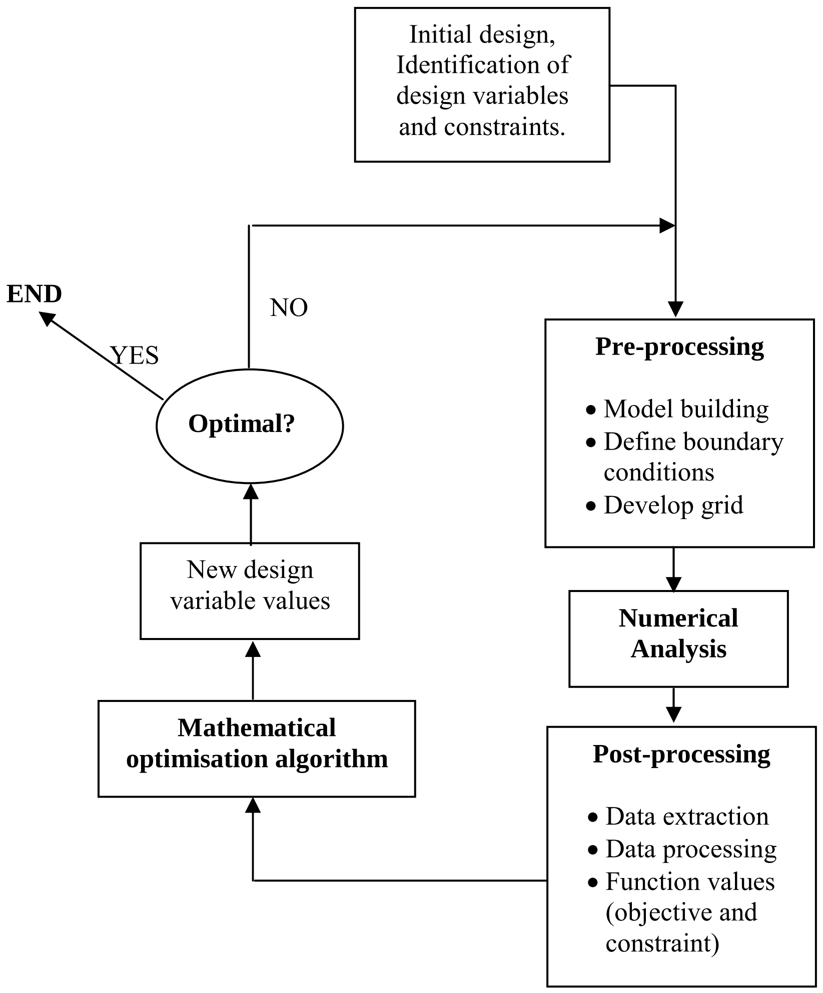

Initial design, Identification of design variables and constraints.
END
NO
YES
Optimal?
Pre-processing
• Model building
• Define boundary conditions
• Develop grid
New design variable values
Numerical Analysis
Mathematical optimisation algorithm
Post-processing
• Data extraction
• Data processing
• Function values (objective and constraint)

Figure 2.12: Basic mathematical optimisation flow-chart

The basic format of a mathematical optimisation method is as follows:

Minimise the objective function: $f(\mathbf{x})$

$\mathbf{x}=[x_1,x_2,\ldots x_n]^T \in R^n$

Subject to m inequality constraint functions: $g_j(\mathbf{x}) < 0$ $\qquad j = 1, 2,\ldots m$

and r equality constraints: $h_j(\mathbf{x})=0$ $\qquad j = 1,2,\ldots r$

where **x** is the vector of design variables in question that require optimisation under these criteria.

An established optimisation algorithm is the leap-frog method for constrained problems (LFOPC) (Snyman) [32]. The method is used internally in a number of design optimisation software packages, e.g., TDO [33], LS-OPT [34], to find the optimum on the approximated surface. The method has proven to be very appropriate for the optimisation of engineering problems where function values are time consuming to evaluate and may exhibit somewhat noisy results. Some of the major characteristics of the basic method are as follows [35]:

- Uses only gradient information for the function value.
- No explicit line searches are performed.
- Very robust, handles steep valleys, discontinuities, and noise in the objective function.
- It seeks relatively low local minima and thus provides a good basis for global optimisation.
- Less efficient than classical methods for smooth near quadratic functions.

LFOPC is a gradient method that generates a dynamic trajectory path from a given starting point towards a local optimum. The underlying analogical principal for this optimisation algorithm is that a particle's motion is traced as it moves over a surface, supposedly under the influence of a conservative force field. The higher the function value, the higher the particle's potential energy. So the point of lowest potential energy will be the optimum. Extending the analogy, as the kinetic energy of the particle increases, so the function value decreases. To ensure that the particle does not continue to oscillate in a valley, interfering strategies are imposed whenever the kinetic energy decreases. The method was extended to constrained problems (LFOP to LFOPC) by the addition of a penalty function, which effectively artificially creates a steep gradient when a constraint is violated.

Global optimisation in typical engineering applications is an inherently difficult goal to achieve since knowledge of the objective is usually quite localised (also, most engineering functions are not strictly convex over **x**). However, certain methods can alleviate the difficulties experienced to some extent. The practice of multi-starting [36,37] is one that involves initiating the optimisation process from various randomly selected initial values for **x**. Further global optimisation techniques like Neural Network Metamodels and Kriging Interpolations (see section 2.6.2.3/4) can also go some way to alleviating these problems. These methods retain all information from previous function value evaluations in their approximation of the objective and constraint functions, thus maintaining a better global perspective of the problem.

A further important term in optimisation is *saddle points*. Figure 2.13 below illustrates an example of a function that has a saddle point. The saddle point is shown as a black star. Mathematically, a saddle point is described as follows [35]:

f($\mathbf{x}$) has a saddle point at $\mathbf{x'} = [\mathbf{x}^\circ, \mathbf{y}^\circ]^T$ if an $\varepsilon > 0$ $\exists$ such that for all $\mathbf{x}$, $\| \mathbf{x} - \mathbf{x}^\circ \| < \varepsilon$

and $\mathbf{y}$, $\| \mathbf{x} - \mathbf{x}^\circ \| < \varepsilon : f(x, y^\circ) \leq f(x^\circ, y^\circ) \leq f(x^\circ, y)$

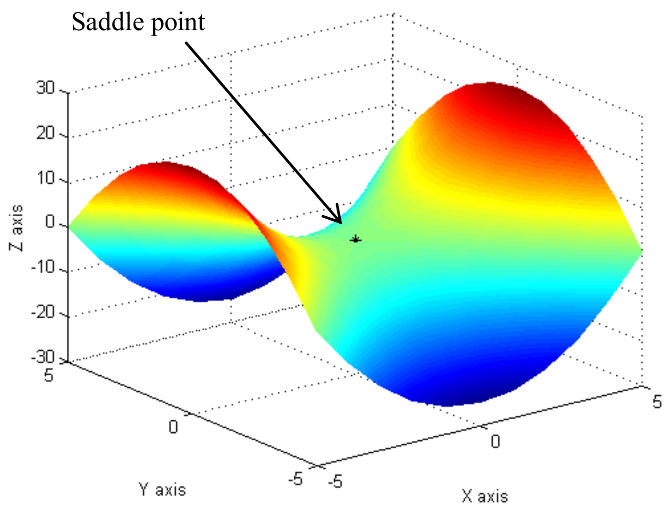


Figure 2.13: 3D function with saddle point ($Z = X^2 – Y^2$)

This concept will also prove to be of significance in this study, where a worst case needed to be found for one of the variables. One of the variables may represent an operational state for the design, in which case the design needs to perform best at its worst operational state.

The sections that follow describe some of the commercially available methods for numerical optimization, as well as methods that are under development.

2.6.1 Dynamic-Q

Snyman’s DYNAMIC-Q method [38] is one of the optimisation methods investigated in this study. This method solves successive quadratic sub-problems using a gradient-based optimisation technique to find local minimums for the real or objective function. The successive sub-problems or approximations are generated by sampling the behaviour of the objective or exact function at specific points in the field. A quadratic sub-problem approximation is then generated based on the gradient at a specific point in the field. Constraints within the field of optimisation can be accommodated by altering the sub-problem to generate a penalty function that, as previously mentioned, effectively creates a steep gradient when the constraint is violated. For each optimisation step, it is necessary to evaluate the objective function value (n+1) times, where n represents the number of variables in question. The DYNAMIC-Q method is also characterised by its use of 'move limits', as an aid for convergence. These represent pre-specified limits on the distance the algorithm may move the design from its current position. In general, this prevents the design from overshooting the optimum due to a quadratic sub-problem whose optimum lies far beyond the local minimum. DYNAMIC-Q is regarded as a very robust method for determining local minimums, and is considerably more economical than genetic or other stochastic type algorithms [39].

For simplification of use, the DYNAMIC-Q optimisation method has been made available in a package (by the University of Pretoria) that has other optimisation methods as options as well as guiding the user through the steps using a graphical user interface (GUI). The package used for this study is the TOOLKIT for DESIGN OPTIMISATION® (TDO) [33] developed by the Multidisciplinary Design Optimisation Group (MDOG) at the University of Pretoria.

2.6.2 LS-OPT

LS-OPT [34] is a further optimisation "toolkit" which is currently packaged with a full purchase of LS-DYNA, and like LS-DYNA is also developed by LSTC [26]. LS-OPT utilises a number of techniques that can be collectively called Response Surface Methodology (RSM) [40], to construct smooth approximations to the actual objective functions in the appropriate number of dimensions, i.e., the dimensions represent the design variables. These response surfaces lead to the approximated sub-problem, and it is the sub-problem which is mathematically optimised. The main reason for the commitment to RSM is that it alleviates the effects of noise, something that can be a major issue when using gradient-based methods.

A further interesting feature of LS-OPT is its trade-off curve capabilities. This tool uses the constructed response surface and the principal of Pareto[†] optimality to establish a curve that represents the best possible result of one criterion if a compromise must be made on another. The curve is constructed by considering a number of values for the criteria to be compromised, reporting the best result, and joining the results in a piecewise linear fashion.

Within an LS-OPT reference frame, experimental design is the procedure through which the points required for the construction of the response surface are selected. These are the points that must be analysed using the solution code in question (e.g. FEM or CFD) to determine their effectiveness. There are a number of techniques available for selecting these points [40]. LS-OPT provides factorial, Koshal, composite, *D*-Optimal, and Latin Hypercube methods. However, only *D*-Optimal design [37, 40] is considered in this study due to its abilities in handling strict constraints, e.g., geometric constraints and irregular design spaces. *D*-Optimal design selects its points based on the solution of max$|X^TX|$, where X is a matrix of possible experimental design points and X^TX occurs in the definition of the least-squares fit coefficients. In LS-OPT, a genetic algorithm is used to solve this maximisation problem. X is a subset of points selected from a larger group of basis points that are spread over the region of interest in either an organised or random manner. The number of points in X will depend on the chosen response surface. The idea behind *D*-Optimal design is to try to reduce the response surface's approximation error due to variance in the experimental results. This would seem a good idea when considering the variance one may get in practical experimental results, however with numerically calculated experimental results no variance will occur for the exact same set of design points. For this reason, Qu, et al. [42] suggest the minimisation of bias error due to the choice of response surface as a better criterion.

[†] V. Pareto, a prominent Italian economist, introduced the idea of Pareto optimality at the end of the 19th century. To define the notion of domination let $\mathbf{f}=(f_1,\ldots,f_n)$ and $\mathbf{g}=(g_1,\ldots,g_n)$ be two real-valued vectors of *n* elements; **f** is partially less than **g** ($\mathbf{f} <_p \mathbf{g}$) if:

$$\forall i \in (1,\ldots,n); f_x \leq g_i \ldots \& \exists i : f_i < g_i$$

If $\mathbf{f} <_p \mathbf{g}$, we say that **f** dominates **g.** Consequently, a feasible solution $\mathbf{x}^*$ is said to be a Pareto optimal if and only if another **x** does not exist such that $\mathbf{f}(\mathbf{x}) <_p \mathbf{f}(\mathbf{x}^*)$. [41]

The following four sections give a brief description of the available response surfaces in LS-OPT. (NB. Kriging Interpolation is available in alpha phase only)

2.6.2.1 Linear Approximations

Linear approximation is inherently the most economical approximation method within a single optimisation iteration. However, the method can be less stable and will often require more optimisation steps to converge, which could ultimately mean more function evaluations are required. The number of points required per optimisation step when using the *D*-Optimal design approach, can be calculated by "1.5*(n+1)+1" for 'n' variables. This may seem like too many points at first, but the multiplication by 1.5, or 50% over sampling, goes a long way to providing one with more certainty about the appropriateness of your response surface. Furthermore, the over sampling filters some of the noise due to the numerical evaluation of the responses surface.

2.6.2.2 Quadratic Approximations

Since many phenomena behave quadratically with respect to their variables, it is often far more sensible to use quadratic approximation techniques. In a *D*-Optimal design, the number of points required per optimisation iteration is, "1.5*(n+1)(n+2)/2 +1". Intuitively one would require more points to construct a quadratic surface than a linear surface, but if the bias error is reduced by using a quadratic response surface, one can expect that convergence may occur sooner than with a linear approximation. In turn, one may require fewer function evaluations over the span of the optimisation.

Both Linear and Quadratic approximation methods in LS-OPT make use of the Successive Response Surface Method (SRSM) [43]. This method effectively relocates and resizes the subspace of interest for each optimisation iteration, in an attempt to "zoom-in" to the region of the optimum design. This results in a more effective use of the function value evaluation to gain knowledge only near the region of interest. However, since a new response surface is constructed, previous data collected in prior optimisation iterations are lost and the new surface loses any previous global validity.

2.6.2.3 Neural Network Metamodels

One of the main points of interest when considering neural networks (NN) and Kriging methods as compared with polynomial response surfaces (e.g., linear and quadratic) is the ability to update the response surface with appended data from subsequent optimisation iterations. This gives improved global validity as well as higher definition in the region of interest when combined with the domain reduction approach of SRSM.

On a basic level, a NN consists of input and outputs that are linked by the neural network's neurons which compute the outputs from the inputs. The NN is defined by parameters like the inter-neuron connection strengths and biases on both the input and output side of the neurons. These parameters are determined in the *learning* process, in which a training algorithm directs the parameters to a state where the error in approximation is minimised. For a more definitive description of Neural Networks see e.g. reference 44.

2.6.2.3 Kriging Interpolation

Kriging is a spatial interpolation technique originally developed by the South African geostatistition D G Krige, in an attempt to more accurately predict subterranean ore reserves. Kriging interpolation behaves similarly to NN in that it has the ability to update itself once more data is appended. The method uses a stochastic correlation to relate a known polynomial function to the unknown function of interest. A complete description can be seen in Simpson [45]. The method's main advantage is its accuracy, since it interpolates the available data points. In comparative studies, Kriging interpolation has demonstrated similar performance characteristics to the previously mentioned metamodelling techniques, but has shown to be less robust [46] and more sensitive to noise [47]. The method is however still under development by LSTC and is only available indirectly and in an alpha phase within LS-OPT.

2.7 Multidisciplinary Design Optimisation

Throughout this chapter, references are made to journal publication and other sources that describe the work that has been done within that specific field. However, this study aims to examine a number of the fields simultaneously.

Due to the diversity of the problem in question, it is felt that a design technique be analysed that encompasses as many as possible of the most suitable methods. This idea leads the study to the field of Multidisciplinary Design Optimisation (MDO). MDO, as the name suggests, is the optimisation of a given design where multiple disciplines are considered for the purposes of establishing design performance.

Within the MDO framework a number of formulations exist. The appropriate formulation depends on factors like the degree of coupling between the disciplines and the ratio of shared to total design variables [48]. A paper by Giesing and Barthelemy [49] provides an industry perspective on MDO applications and needs. The standard method is to evaluate all disciplines simultaneously in one integrated objective and constraint set by applying an optimiser to the multidisciplinary analysis (MDA), similar to single-discipline optimisation. Figure 2.14 below is an adaptation of Figure 2.12 above that shows the similarity with single-discipline optimisation (SDO). The major difference between SDO and MDO is that two or more solvers may exist in parallel. Furthermore, all variables are not necessarily used by both solvers, this may be since certain variables do not form part of the model used for a discipline or that the influence the variable has on the results used from the discipline are insignificant. For this reason, variables are isolated into sub-sets used only by the specific discipline. This method of MDO is numerically the most correct, but in industry it may not always be possible to analyse a set of design with multiple disciplines simultaneously or even at the same physical location. It is for this reason that other formulations exist. Fortunately for the purposes of this study, both disciplines, CFD and FEM, will be available simultaneously.

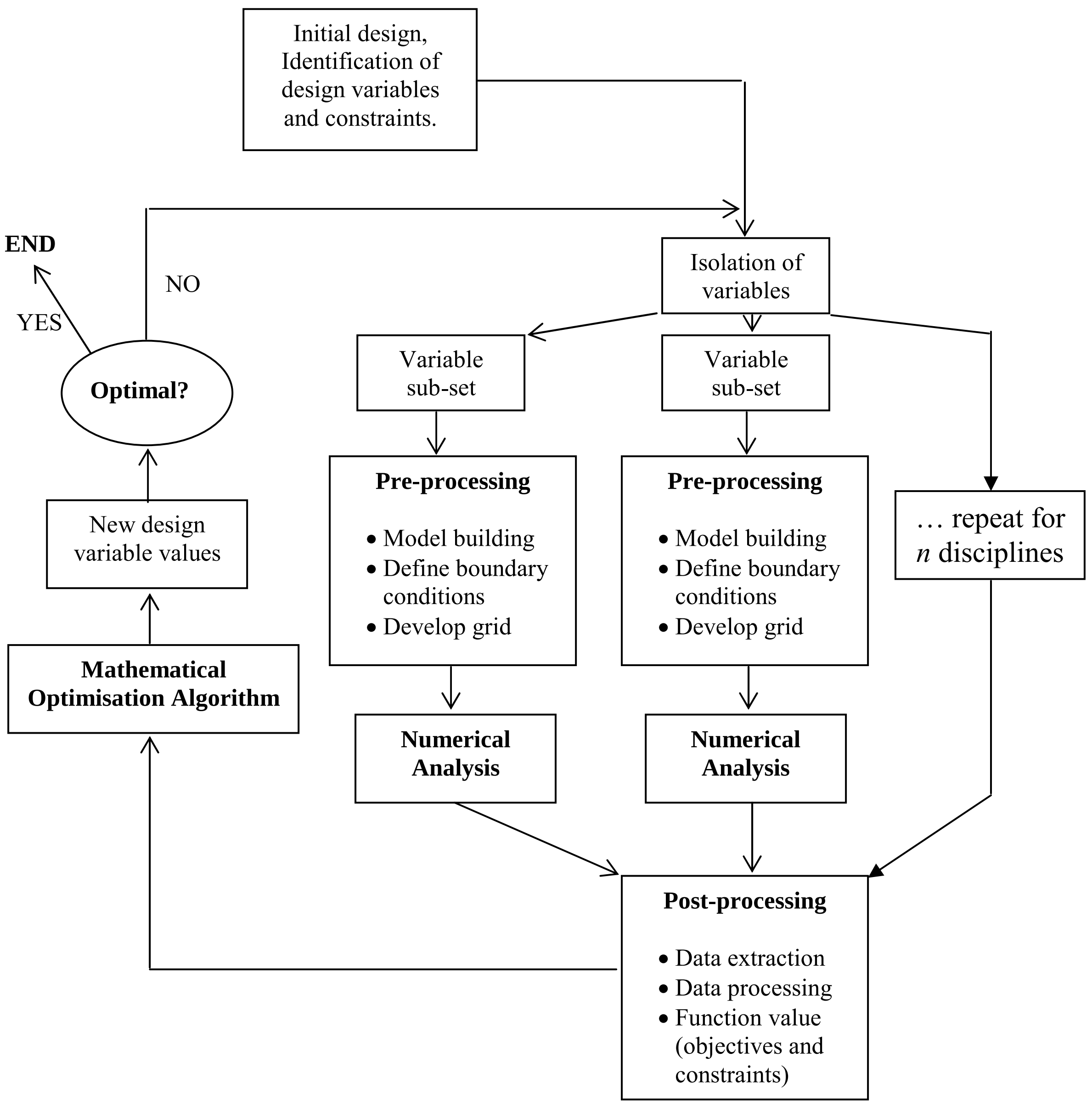

Initial design, Identification of design variables and constraints.
END
NO
YES
Optimal?
Isolation of variables
Variable sub-set
Variable sub-set
Pre-processing
• Model building
• Define boundary conditions
• Develop grid
Pre-processing
• Model building
• Define boundary conditions
• Develop grid
… repeat for n disciplines
New design variable values
Mathematical Optimisation Algorithm
Numerical Analysis
Numerical Analysis
Post-processing
• Data extraction
• Data processing
• Function value (objectives and constraints)

Figure 2.14: MDO cycle for *n* disciplines

2.8 Conclusion

Sloshing is a real and present challenge, one which has troubled engineers over the last century in one form or another. This chapter has one outstanding message; there are a wide number of techniques that are available for use in the design of liquid tanks, a number of which are interdependent. Depending on the application, it may be required of the engineer to consider a range of design criteria and standards that are pertinent. This study will examine a number of techniques and issues surrounding liquid container design, both from an optimisation and validation perspective.

CHAPTER 3: Modelling of Sloshing

3.1 Introduction

This chapter details the modelling methods used for the sloshing event. A description of the automated grid generation procedure and related issues is provided. An overview of the validation procedures follows and an analysis of the results from the validation within the context of the sloshing modelling will conclude the chapter. The aim of this chapter is to determine the feasibility of using CFD to evaluate sloshing performance as part of a liquid container design cycle.

3.2 Computational Fluid Dynamics

Due to availability, the commercial CFD code Fluent v6.x [19] is used throughout the study for the modelling of the sloshing event. Fluent's grid generator or pre-processor is Gambit, and the section that follows provides a description of the automated use of this pre-processor. Automation of the pre-processor is necessary for the optimisation procedures that follow in chapters 4 and 6.

3.2.1 Grid Generation

Once the geometry and topology of a tank and its damping devices are established, it is necessary to recreate the fluid domain in the pre-processor. In three dimensions, the damping devices may include baffles with holes, or as in reference 14 vertical cylinders. In two dimensions, one can only consider horizontal circles or baffles with or without slots. Figure 3.1 and Figure 3.2 illustrate typical 3D and 2D geometries encountered during this study, respectively. In the context of CFD, it is always advantageous to simplify a given geometry, in that it will reduce computational expense. To simplify the 3D geometry, a symmetry plane can be defined as illustrated in Figure 3.1. 3D geometries may in turn be further simplified to 2D, although features like holes can no longer be captured and side wall effects will also not be modelled. 2D geometries are intuitively computationally less expensive, however in all CFD analyses one must be wary of over-simplifying geometry to the point where the analysis is no longer meaningful.

Figure 3.1 shows a section through the mesh of the 3D geometry, comprised of hexahedral cells only. Although the CFD code is capable of handling tetrahedral cells, accuracy and solution times when using hexahedral cells are far superior. Hexahedral cells in the circular holes are achieved by using an iron-cross formation, that subdivides the circle into five four-sided sections. The 2D geometry is far simpler from the perspective of meshing since it easily lends itself to a fully-structured Cartesian mesh. The implication of 3D versus 2D can be best appreciated when one considers that a typical 3D sloshing analysis will take approximately 48 hours (400 000 cells), while a 2D analysis will take approximate 4 hours (25 000 cells) for the same period of real time solved on the same computer (2GHz P4 Linux workstation). All operations in Gambit can be performed in a command line and in turn through a journal file that when executed will automatically generate the mesh. Appendix A provides a sample 2D journal file.

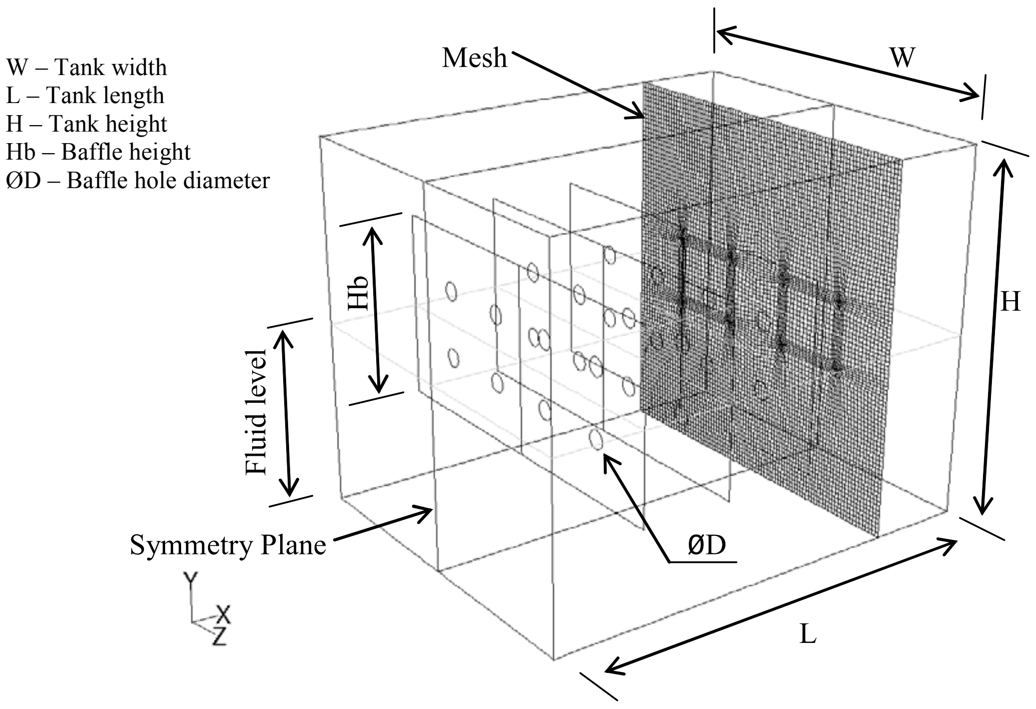
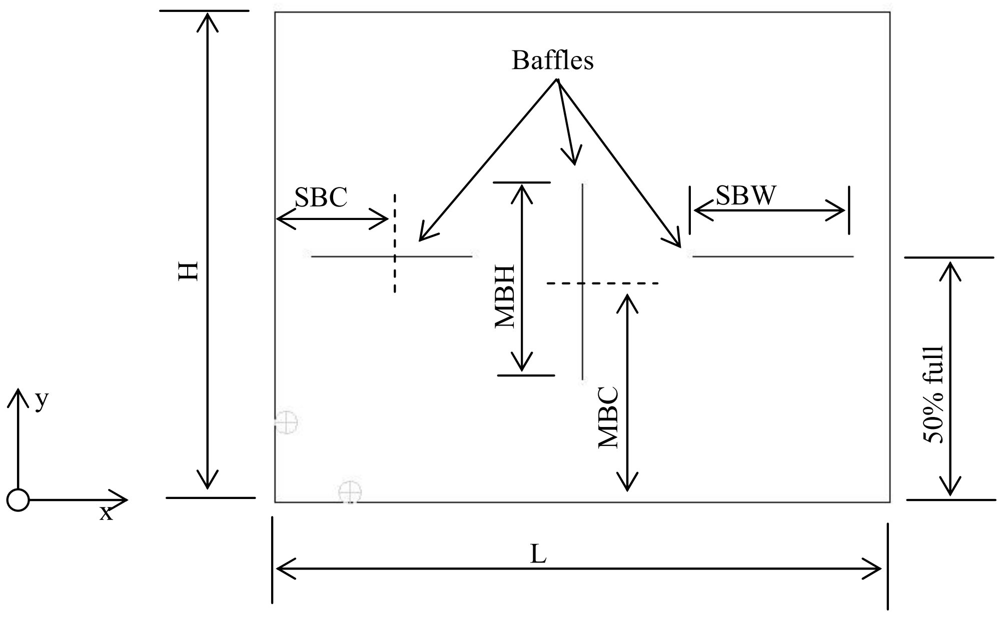


Figure 3.1: Typical 3D geometry and mesh



H – Tank height
L – Tank length
SBC – Side baffle centroid
SBW – Side baffle width
MBH – Mid baffle height
MBC – Mid baffle centroid

Figure 3.2: Typical 2D geometry

3.2.2 CFD Model Setup

Once the mesh has been generated in Gambit, it can be exported as a mesh file and imported into the CFD code Fluent. At this point the model must be scaled and set up for the flow assumptions that are appropriate for the case. This section will describe the number of settings that are used in the CFD model and where appropriate discuss the implications of the assumptions made.

Firstly, this type of analysis is transient and therefore requires an unsteady formulation of the CFD model. Unsteady simulations are traditionally more time consuming and require that a time step size be provided. The time step size must be sufficiently small to ensure the stability of the code and a time-accurate solution. Typical values for this study range from 1 to 2.5 ms and usually equates to approximately 10 sub-iterations per time step for stability in the solution of the most non-linear wave motions.

Since the liquid container is partially filled, two phases exist in the flow field, liquid and gas. The Volume of Fluid (Section 2.4.3.4) model is used to monitor the motion of the free surface. This model is commonly used for the modelling of sloshing [31]. The formulation is simple and relatively inexpensive, and provides good results in cases that involve large free surfaces. All material properties and conservation equations are altered to consider all terms as volume fraction weighted summations. Within the solver, once the flow field has been initialised with liquid throughout, a region is defined that corresponds to the initial location of the gaseous phase. A volume fraction of unity for the gas is then patched over this region.

In this study, the two phases are water and air. Since there is a free surface, surface tension must also be considered for the water-air interface. The surface tension coefficient (γ) was considered to be 0.073N/m at 20˚C [50].

Since the case in study has no inlet or outlet boundaries, a load curve is used as a user-defined boundary condition. An acceleration curve is imposed on the model in the form of a momentum source term. A user-defined c-code converts acceleration data into momentum source (see Section 2.4.3.2) for the *i*-th cell through the formulation given in equation 3.1 below (illustrates formation for *x*-direction), where S_i^m is the momentum source for the *i*-th cell, *x* is displacement in the local *x*-direction, *t* is real time, and ρ is the volume fraction weighted density in the *i*-th grid cell. Appendix B provides a sample c-code that would do this conversion if provided with a text file containing an acceleration versus time signal as two columns of data.

$$S_i^m = \frac{\partial x}{\partial t}\rho_i \tag{3.1}$$

As in references 31 and 51, turbulence is considered very low and if transitional flow exists, it is confined to short duration, small region events. For this reason, it is considered more appropriate to assume a laminar flow field. The validation section of this chapter illustrates the difference in results when assuming turbulent flow. It is important to remember that when assuming turbulent flow, the viscosity within the Navier-Stokes formulation will increase in all cells, including those involved in strictly laminar flow (See Section 2.4.3.4).

As with the pre-processor Gambit, all settings and commands can be done through a text user interface. A sample journal file that sets up the 2D CFD run can be seen in Appendix C.

3.3 Experimental Validation

Since a major part of this study revolves around the utilisation of CFD as a technique to model sloshing, it would be appropriate to perform some level of validation. The aim of the validation is to evaluate the overall performance of the CFD code with its chosen settings, as a tool to determine the level of sloshing in a liquid container.

3.3.1 Experimental Setup

The experimental modelling is divided into two phases. The first and preliminary phase involved a small Perspex or Plexiglass container (H = 200mm, W = 200mm, L = 250mm), mounted on a set of inclined rails. The second phase was an extension of the first phase and involved a larger Perspex container (H = 400mm, W = 400mm, L = 500mm), mounted on the loading bed of a 1-ton truck. In both cases, a digital video camera was aligned with the side of the tank to monitor the motion of coloured water in the partially-filled container.

The Perspex tank for the first phase is seated horizontally on a trolley that runs down an inclined track (see Figure 3.3 below). The frame consists of a 5m-long double track, tracks 385mm apart, with a drop in elevation of 1.7m. The trolley is fitted with ball-bearing wheels and runs freely along the rail. Attached to both the front and rear of the trolley is a rope and shock cord. The shock cord represents a method of accelerating the trolley from standstill and decelerating it before it reaches the end of the rail. The single rope completes a cycle through four pulleys and runs in a one-way cam cleat. The cam cleat is mounted rigidly to the frame and prevents the tank from moving back up the ramp under the force applied by the decelerating shock cord attached to the rear of the trolley. The idea is that the tank will accelerate down the rail, and then decelerate to a standstill as it comes into frame for the digital camera, mounted on a tripod, that is located near the end of the rail (See Figure 3.4). This will provide a digital video of the sloshing as the coloured water returns to its horizontal state of static equilibrium.

To link the experimental model with the CFD model in the context of the momentum source term as discussed in Section 3.2.2, an acceleration signal is measured on the Perspex tank. A 10.36mV/g Shear translational accelerometer and a PL202 analyser, shown in Figure 3.5, are used to measure the acceleration. The data are processed in Matlab v6 [52] and also involves the filtration through a low-pass Butterworth filter. The filtration at 5Hz is necessary to ensure the stability of the CFD code, when applying the signal as a load curve. Figure 3.6 below shows both the unfiltered raw data from the accelerometer (mV), and a filtered acceleration signal (m/s^2).

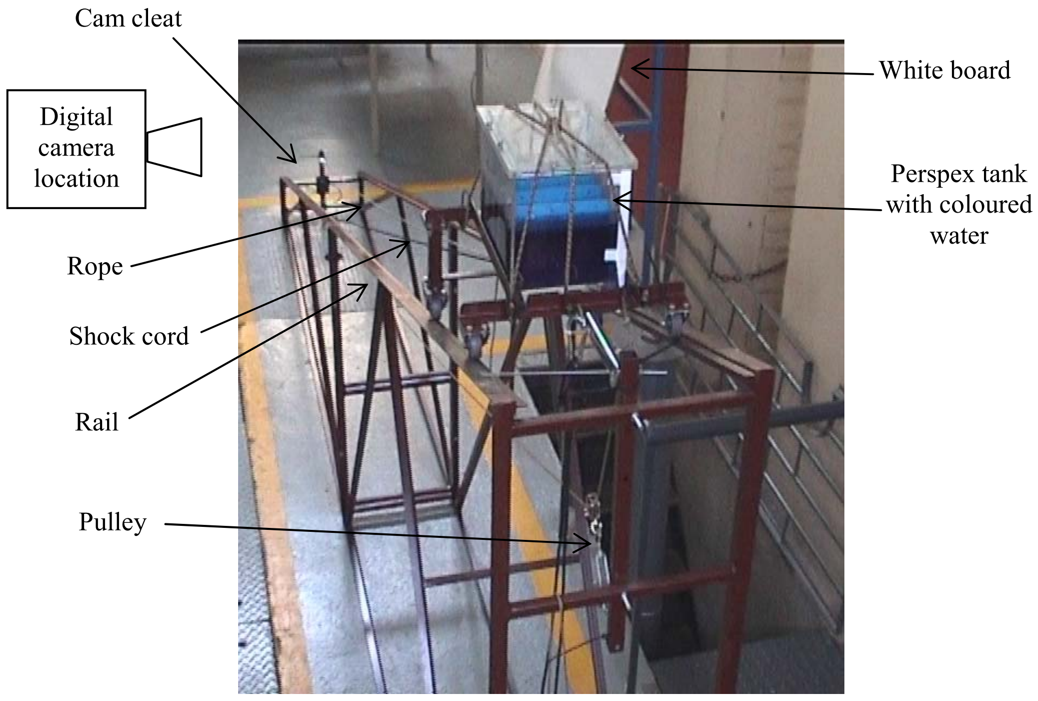


Figure 3.3: Phase 1 experimental setup

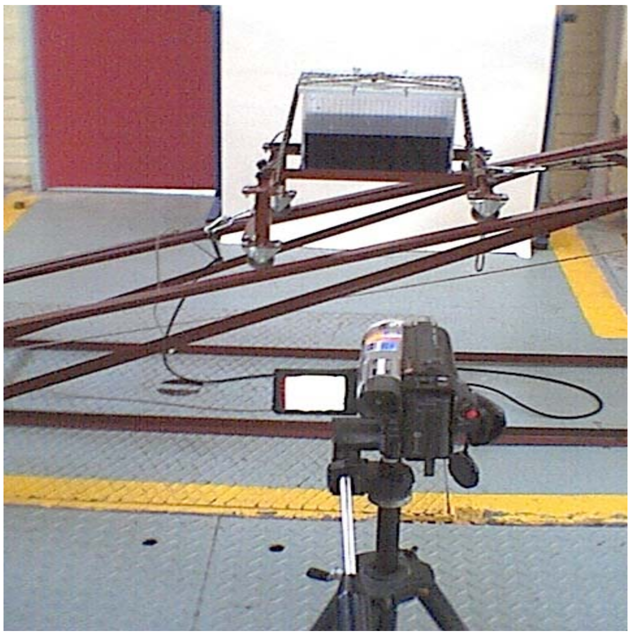


Figure 3.4: Location of camera at end of rail.

Figure 3.5: Acceleration measurement equipment

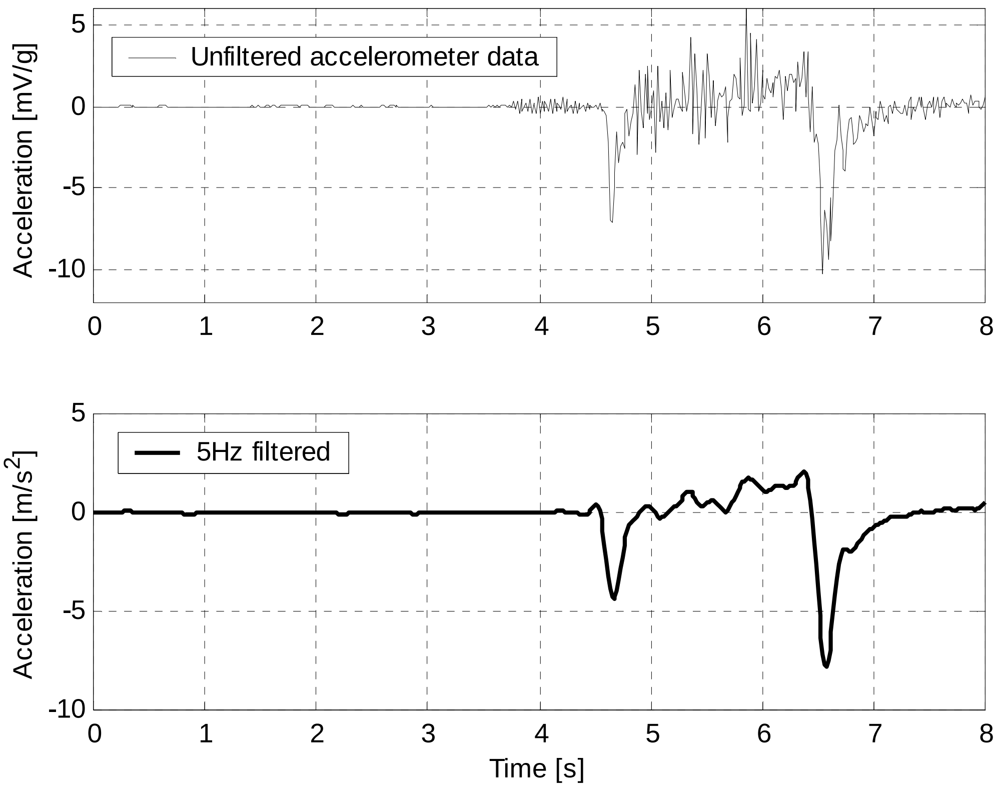


Figure 3.6: Acceleration data for experimental phase 1

It should be noted that although the filtered signal is significantly smoother, some signal detail and amplitude is lost as a consequence of filtration at this frequency.

The second phase of experimental modelling is an adaptation of the first phase to remedy some issues encountered. Figure 3.7 below provides a schematic representation of the setup as well as a digital photograph of the tank after instrumentation. Table 3.1 below gives an overview of the equipment used. The vehicle is accelerated from standstill to 40km/h and then decelerated back to standstill on a near-level tarmac road. The laptop stores both the pressure and acceleration data provided to it through its parallel port from the Spider data logger. The Spider data logger receives voltages on two serial ports, that represent the outputs from the accelerometer and the pressure sensor, both sampled at 100Hz. The pressure sensor is mounted at various points on the liquid container through holes drilled in the Perspex side walls. Both the pressure sensor and the accelerometer require dedicated and rated voltage supplies. The digital camera stores short videos of the motion of the fluid in the tank during the vehicle's acceleration and braking manoeuvre. Each configuration of pressure sensor location and baffle setup is accompanied by its own video footage, acceleration data and pressure data, as it is impossible to exactly replicate the acceleration and braking manoeuvre.

The data from the laptop are further processed by converting the voltage data to m/s^2 and Pascal in the case of the acceleration and pressure data respectively. The processing is done in Matlab v6 [52] and also involves the filtration of the data through a low-pass Butterworth filter. However care must be taken not to remove too much content or amplitude, such that the CFD simulation will no longer adequately represents the test. Figure 3.8 illustrates the unfiltered data compared with data filtered at 1Hz and 5Hz for the phase-2 experimental setup. The unfiltered data is clearly very noisy and discontinuous, while the 1Hz filtered signal suffers from a loss of amplitude and detail. A filter frequency of 5Hz provided the best compromise in that it ensured a flat signal at the start of the test. Using the noisy signal in a CFD simulation caused bubbling and higher frequency sloshing that was not observed experimentally.

Table 3.1: Equipment used in phase-2 experimental setup

Equipment	Input	Output
Acer P1 laptop	220V and Parallel port	Stored CSV pressure and acceleration data
Spider 8-port data logger	220V and serial ports	Parallel port
Voltage Inverter	12V	220V
Sony Digital Video Camera	220V and image/light	Stored video footage
Shear 98mV/g Translational Accelerometer	9V	98mV/g and acceleration force
WIKA 100mBar diaphragm pressure sensor	20Vand induced pressure	4-20mA converted to 2-10V
2-channel Voltage supply	220V	20V (adjustable)
Accelerometer Power Supply	9V battery	-

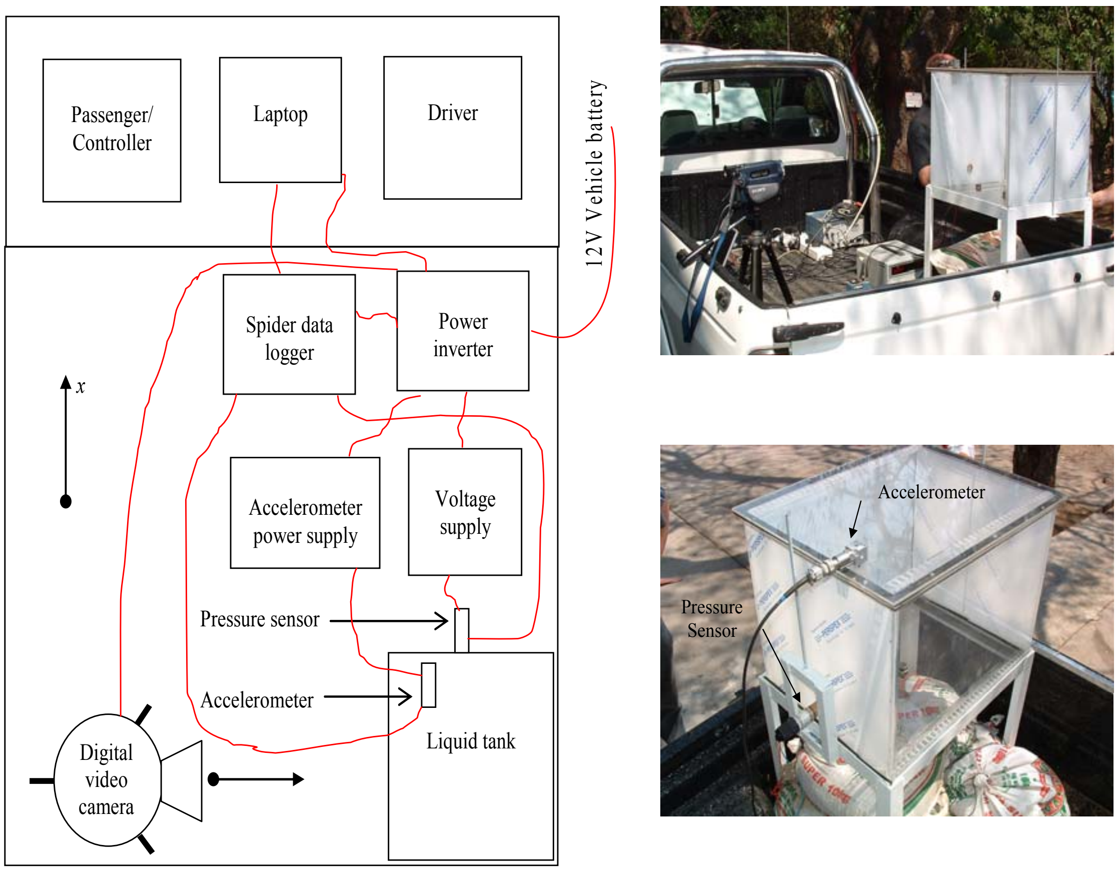


Figure 3.7: Phase-2 experimental setup

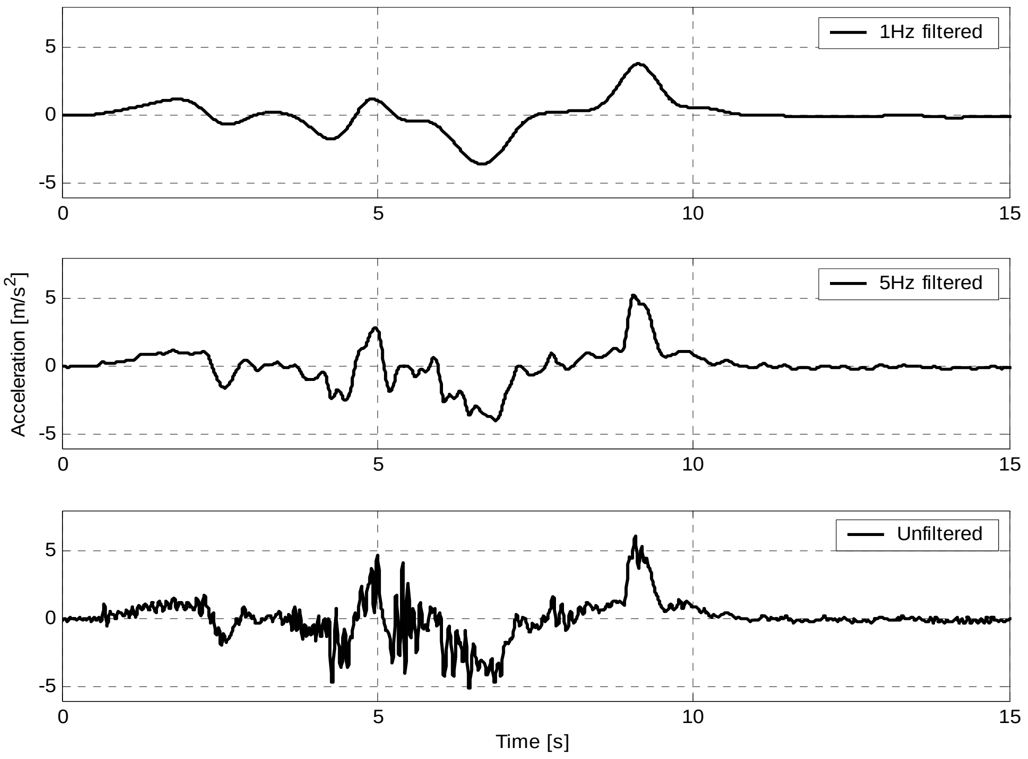


Figure 3.8: Acceleration curves for phase 2 experimental setup

The accelerometer used was supplied with a calibration certificate (Appendix D), and the WIKA pressure sensor was calibrated to static conditions. More detail on the calibration of pressure sensor can be seen in Appendix E.

3.3.2 Comparison of Results

This section provides the comparative results from the experimental and CFD models. The results presented are for both the 1st and 2nd phases of validation, with the second phase data providing qualitative and quantitative perspectives.

3.3.2.1 Phase 1 Experimental Validation

In order to validate the CFD results with the experimental setup, a computer model is recreated with the same dimensions and load curve as that measured in the experiment. This model is then solved, with digital image files created so that the computed wave nature can be compared to the experimental wave nature. The digital video clip taken during the experiment, showing the tank sloshing, can be linked to the acceleration curve at the time where the acceleration returns to zero. The frame rate of the video camera is 25 frames per second, giving a frame period of 0.04s. The video clip may now be viewed frame by frame and compared with the corresponding CFD results.

The first configuration considered is a baffled case with baffle height 80mm and hole diameter of 15mm (Hb = 80mm, ØD = 15mm in Figure 3.1). Figure 3.9 below shows the form of the tank with baffle as generated in Gambit. Figure 3.10 below shows a comparison of free-surface states for this baffled case. The CFD model is a 3D model with a symmetry plane as illustrated, and a laminar flow field assumption.

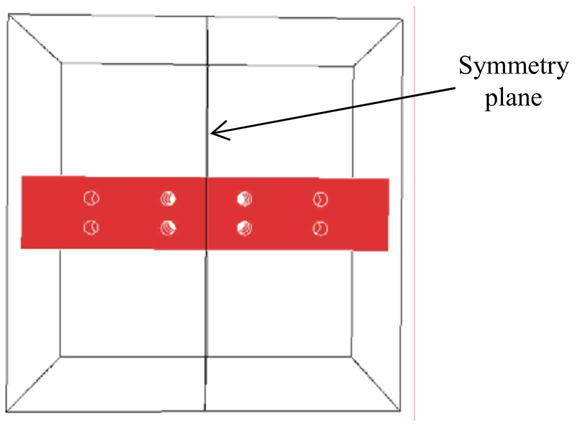


Figure 3.9: Baffled validation case (Hb = 80mm, D = Ø15mm)

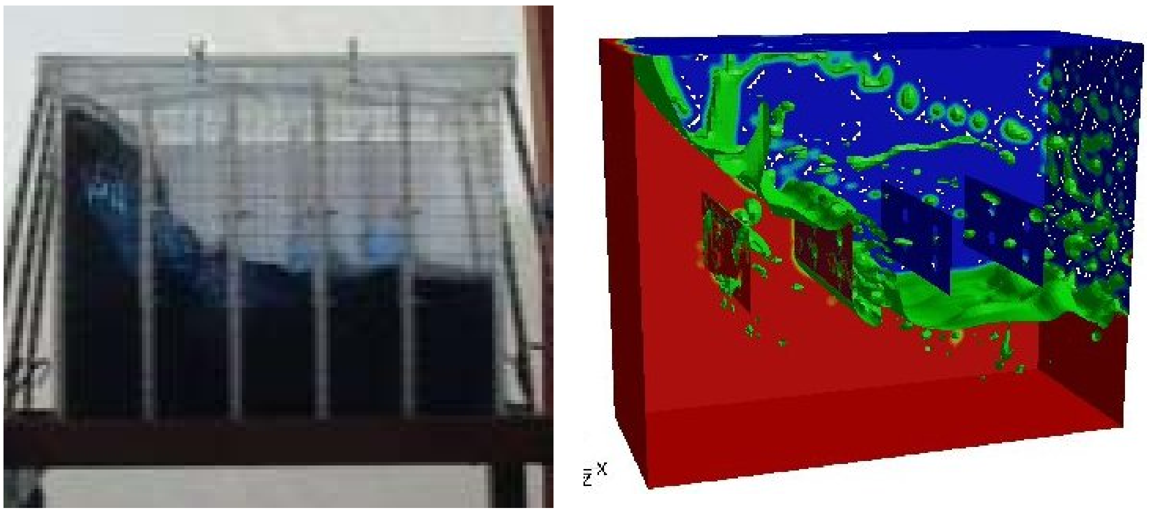


Figure 3.10: Sample of comparative free-surface states for baffled case (Experimental vs. CFD)

One can see that the CFD code is very capable of handling all the non-linearities of the sloshing, although the detail of the free surface is not correctly captured. However, the exact behaviour of all the splashes on the free surface is not repeatable and to try to match it would be futile.

Figure 3.11 below provides a frame for frame comparison of the wave motion for the baffled tank. The video camera provides an image of the coloured water that can be compared with a plot of contours of density from the CFD. As can be seen, the general wave motion and free surface behaviour is quite similar with some variations in amplitude. Variations in amplitude may be a result of the filtration of the acceleration signal as discussed in section 3.3.1. The general motion of the fluid is somewhat chaotic, in part due to the presence of the baffles. This chaotic nature of the wave motions makes it more difficult to make comparisons of definitive events during the sloshing.

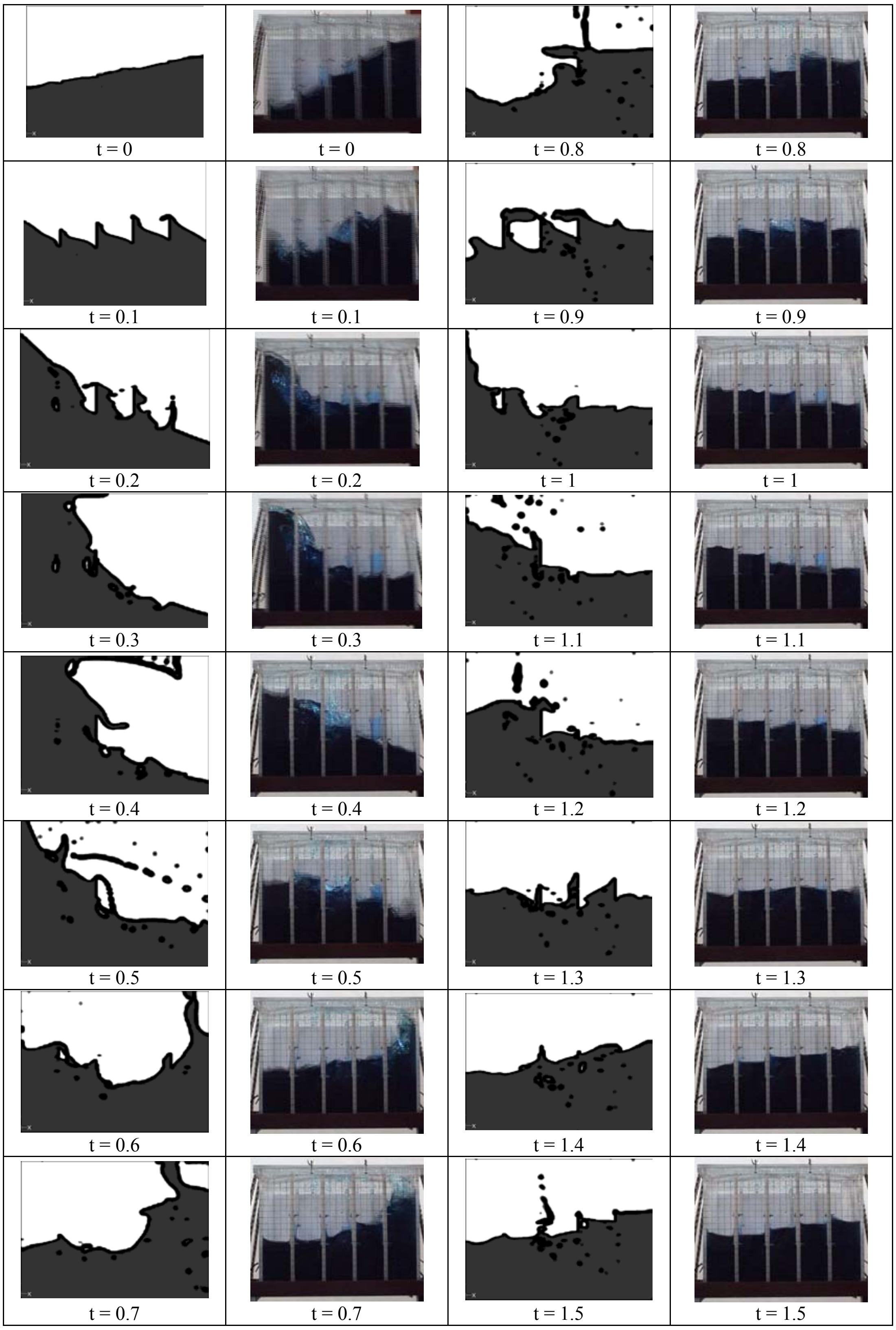


Figure 3.11: Wave comparison for baffled tank (LxWxH,Hb,ØD) = (250x200x200,80,15)mm: Experimental vs. CFD model at time, t [sec]

A further configuration considered is the same tank as above, but without any baffles. It is hoped that more definitive events may occur when there are no baffles to brake up the free surface. Figure 3.12 below illustrates a sample of a comparison of the free-surface states for the un-baffled case.

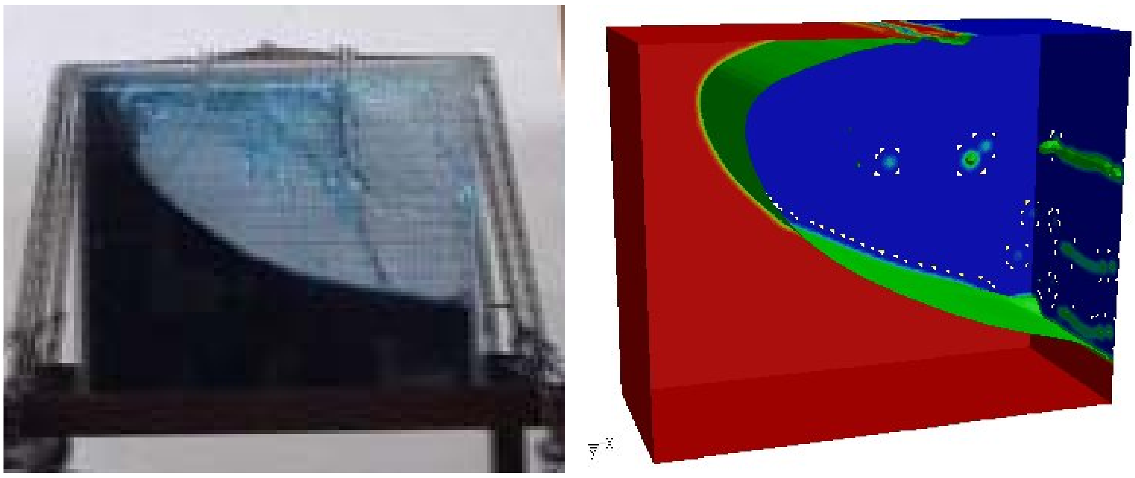


Figure 3.12: Sample of free surface states for un-baffled case (Experimental vs. CFD)

The laminar behaviour of the fluid on the roof of the container leads one to reconsider the laminar flow field assumption. This observation stimulated the consideration of a turbulence model for the validation. A first attempt at using a turbulence model involved the use of the k-ε model [19], however this did not provide results with any major difference to those seen in the laminar simulations. In particular, the separation of the fluid from the roof of the container when the free surface starts to fold over itself does not occur as soon as in the experimental model. A further model, Wilcox's k-ω model [20], is thought to predict earlier separation and provided slightly different results to the laminar simulations. Figure 3.13 below provides a frame-for-frame comparison of the un-baffled case for both laminar and turbulent assumptions. The t = 0.2sec frame clearly illustrates the earlier separation, however the same level of chaos in the free surface at the time of separation is still not evident. It is thought that the noisy three-dimensional loads experienced by the experimental model are what induce the additional chaos. Although the frames shown do not correspond exactly in time, they do exhibit similar wave forms. Discrepancies observed may be attributed to:

1. The cam cleat slips a different amount every run.
2. The CFD acceleration curve represents only one dimension of the accelerations felt by the experimental model.
3. The filtering process rounds the peaks of the signals, and so detail and amplitude is lost.
4. The baffle holders intrude into the flow and induce energy losses not included in the CFD model.
5. The flow field may experience some turbulence in certain areas but cannot be modelled as partially turbulent and partially laminar within the CFD model. As the VOF model does not support Large Eddy Simulations in Fluent, this factor cannot currently be evaluated.

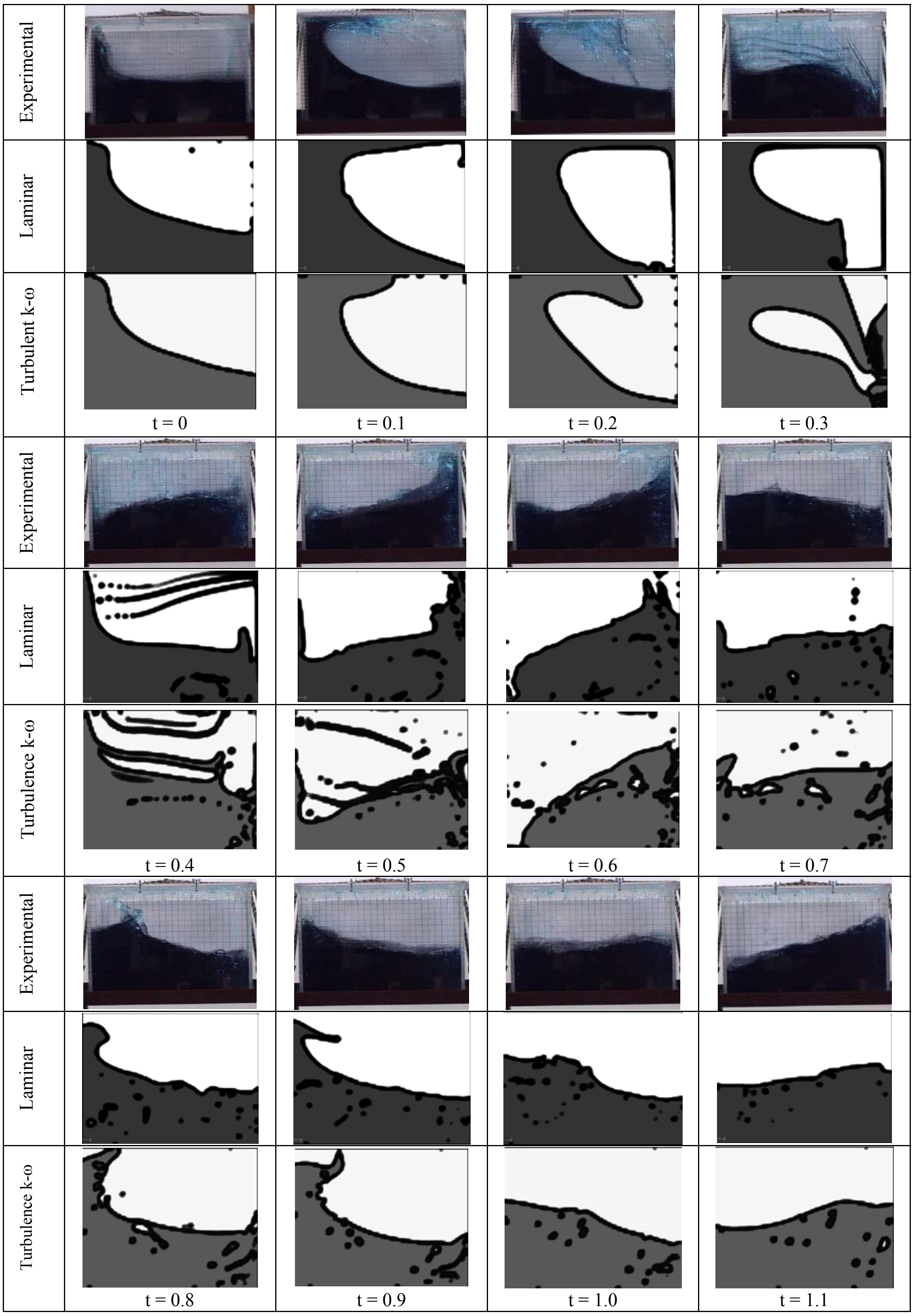


Figure 3.13: Wave comparison for an un-baffled case; Experiment, laminar flow and k-ω turbulence model at time, t [sec]: (LxWxH) = (250x200x200)mm

3.3.2.2 Phase 2 Experimental Validation

Phase two of the validation study attempts to improve the quality of the experimental setup used in phase one. The setup for phase two was discussed in section 3.3.1, but some of the major differences include:

1. An increased tank size to allow for pressure measurement and to reduce the effects of fittings inside the tank.
2. The experiment is done on the back of a truck so that the entire event can be analysed.
3. A pressure sensor is used to provide quantitative data.
4. Each run has a corresponding load curve since the motion of the vehicle is less repeatable than in phase 1.

The following section provides an overview of the comparative results for the phase 2 container (LxH; 500x400mm in Figure 3.2) with and without baffles. With reference to the phase 2 container, three pressure points are considered. Figure 3.14 indicates the location of the three pressure points (P1, P2 and P3) considered. All points are on the centre plane of the container. Since the different pressure points did not give much further insight, only pressure point 1 will be considered in the text for comparison.

Figure 3.15 shows a comparison of the liquid motion of the baffled and un-baffled cases for both the CFD and experimental model. The comparison is of frames from the digital video and a pressure contour plot in the liquid from the CFD simulations. The pressure contour plot provides us with additional data about the liquid phase that was not seen in the first validation phase. All frames are referenced to a specific sloshing event that occurs at the end the vehicle's deceleration. Points to take note of include the fact that experimental wave amplitudes are in general higher than CFD wave amplitudes, especially for baffled cases, although wave behaviour is quite similar. This may be a result of the filtering of the acceleration signal that does remove some peak accelerations, as can be seen in Figure 3.8 above. Disagreement is evident near in Figure 3.15 near 8 seconds. The figure illustrates the recovery of this disagreement as the low frequency acceleration experienced by the tank is reduced as the vehicle comes to rest. The fluid motion that follows represents natural oscillatory modes and are thus unaffected by transient input signal. The final two slides show excellent agreement in this regime. The disagreement is confirmed in Figure 3.16 where further comparative data are provided in the form of gauge pressure curves extracted from the CFD and experimental setup and plotted over each other.

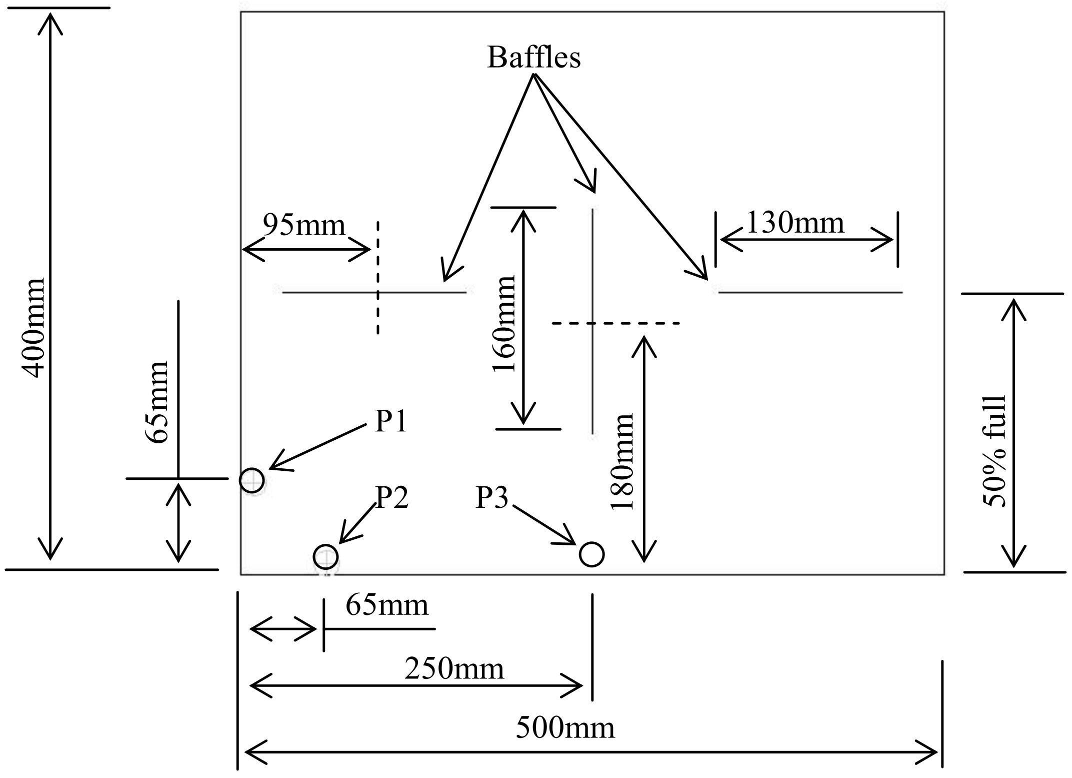


Figure 3.14: Configuration of phase two container and pressure point locations

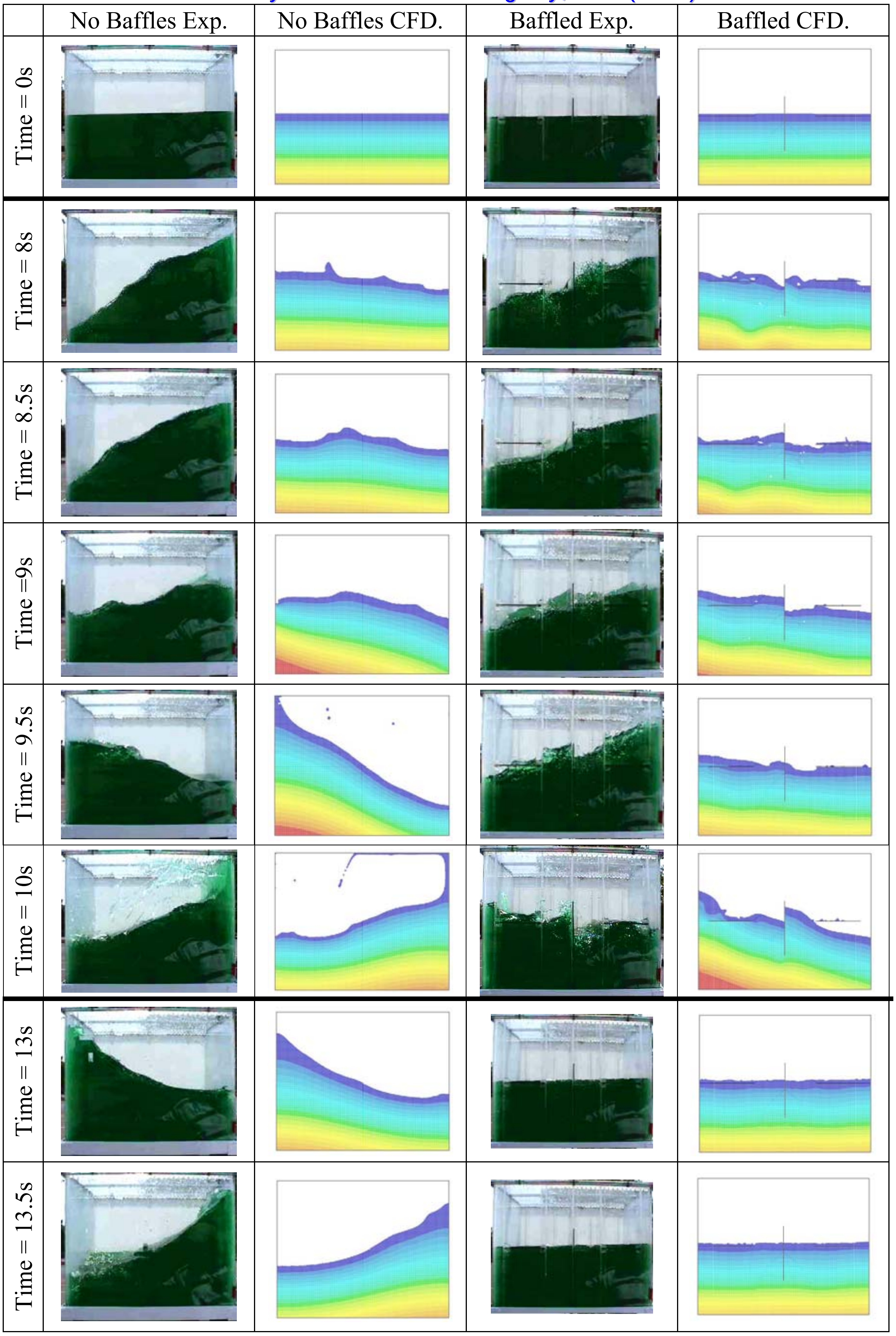
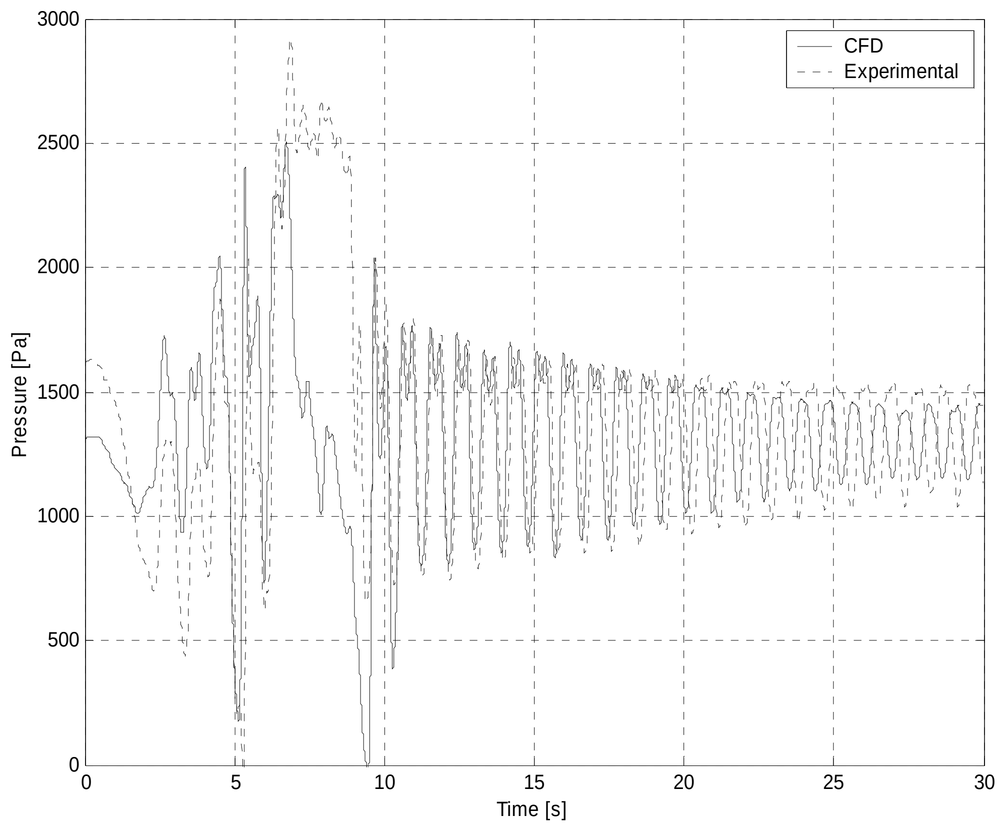
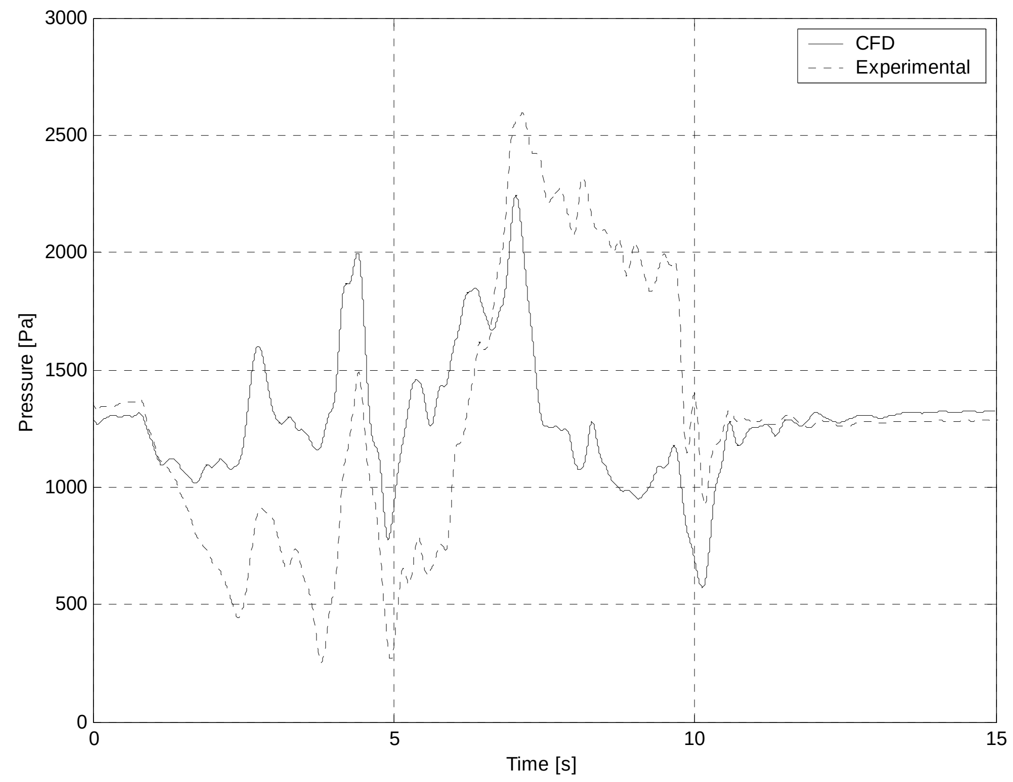


Figure 3.15: Comparative frames of liquid motion for CFD and experimental models (50% fill level)

The experimental pressure signal is converted according to a previously performed static calibration test. All data, both CFD and experimental, are again low-pass filtered at 5Hz. Variations in initial pressure levels (too high a pressure measured by the diaphragm of the transducer) is thought to be due to dithering, an effect caused by higher frequency excitations induced by the vibrations of the vehicle's moving mechanical parts (e.g., engine, gearbox). The baffled case clearly illustrates the very similar flow trend but differing amplitudes between the experimental and CFD results. This may be in part due to the deficiencies of the piezo-crystal linear accelerometer at low frequencies. The no-baffles case does not suffer as much from the variation in amplitudes since the waves are more often constrained by the lid of the container, i.e., the hydrostatic head cannot increase beyond the height of the tank. The water in the CFD model forms a lower slope against the wall than in the experiment, implying that the hydrostatic pressure in the experiment is maintained longer at the maximum head value than in the CFD model for the 7-10 sec time frame.



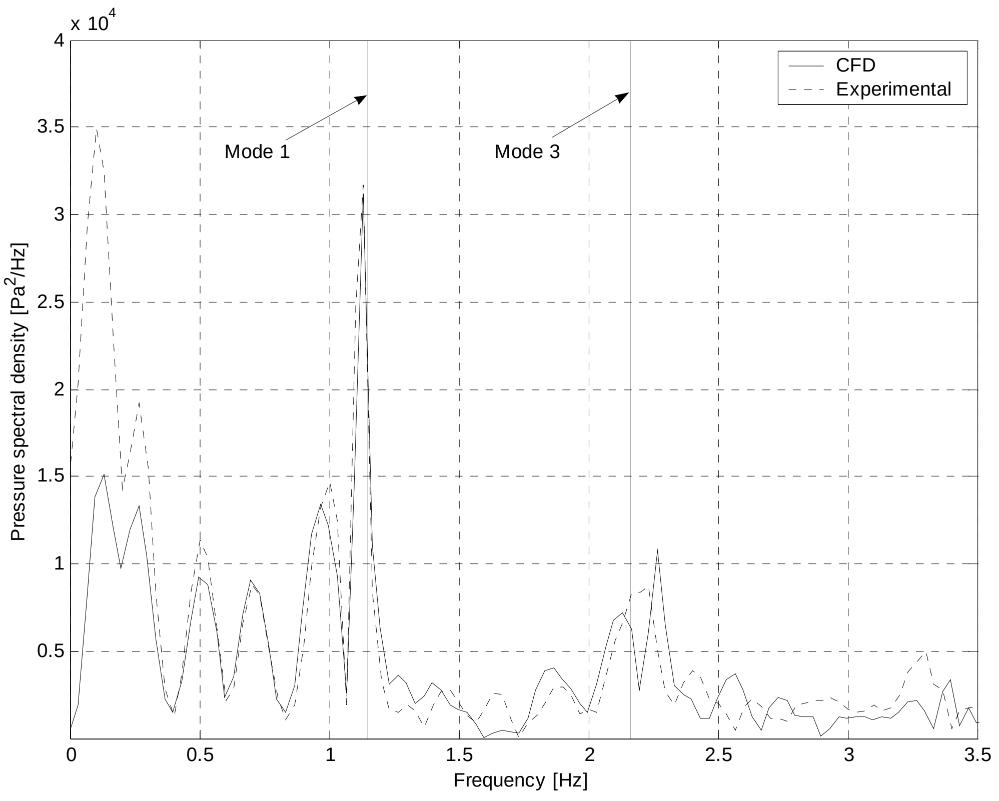
Pressure point P1 (No baffles case)



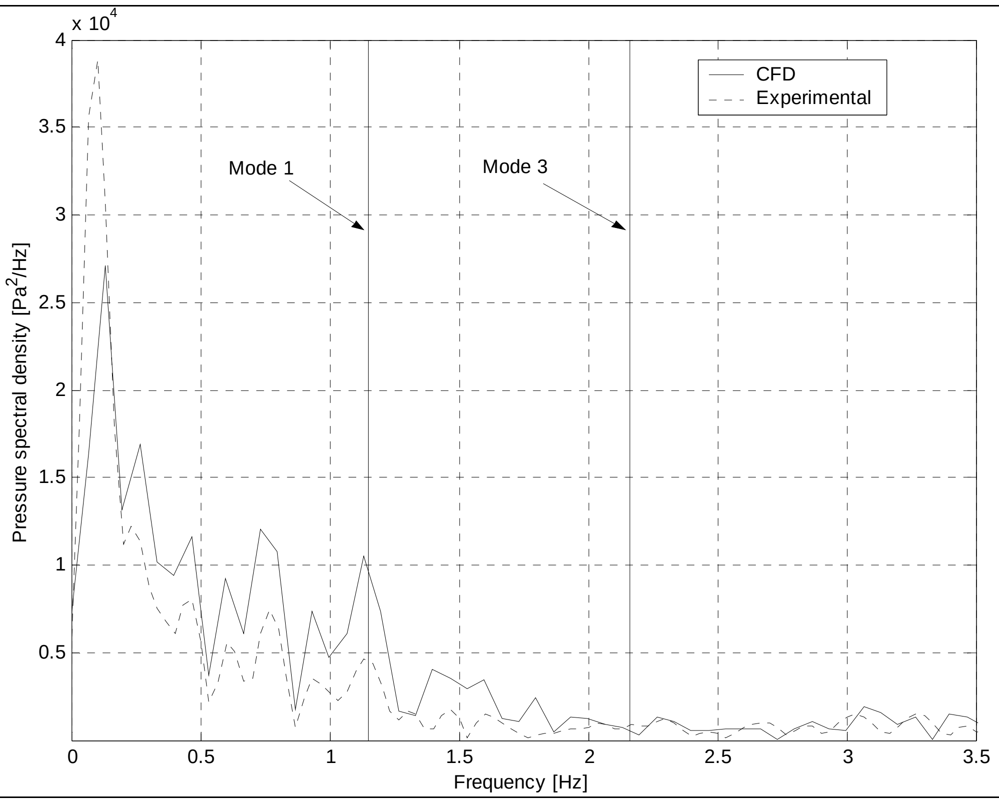
Pressure point P1 (Baffled case)

Figure 3.16: Comparative gauge pressure plots for the experimental model and the CFD

The above data can be further processed and compared from a pressure spectral density perspective. All pressure signals are passed through a Fast Fourier Transform and converted from the time domain to the frequency domain. Figure 3.17 below illustrates these data for the four cases in question (CFD and Experimental for baffles and no baffles), with spectral density (Pa^2/Hz) on the vertical axis and frequency on the horizontal axis. Also shown are the 1st two odd oscillatory modal frequencies based on linear flow theory [14]. The no-baffle case shows excellent agreement between the CFD and experimental model, while the baffled case once again shows agreement in the flow trend but a variation in amplitude. The 1st two odd oscillatory model frequencies are very closely captured, although the damping for the baffled case has removed the 2nd odd mode content. It should be noted that the magnitude of the low-pressure section of the signal will be higher for a lower amplitude wave. Both plots show the error in the 8 to 10 second range as low frequency (<0.5Hz) variations in amplitude.



Pressure point P1 (No baffle case)



Pressure point P1 (Baffled case)

Figure 3.17: Comparative amplitude/frequency domain plots of pressure signals

3.3.3 Conclusion

Within the spectrum of this study, the results achieved are generally very acceptable. The validation study provides sufficient insight into the validity of the numerical model as well as some level of insight into the phenomenon of sloshing itself. The un-baffled case also provides additional and more interesting data about the oscillatory modes. Certainly within a design perspective, an improved numerical model would seemingly translate to an improved physical design.

CHAPTER 4: Optimisation for Sloshing

4.1 Introduction

This chapter covers all work done on the optimisation of sloshing, including the definition of the optimisation problem within the context of sloshing and how the software is set up to achieve automation of the optimisation process. Analysis for sloshing is done by the commercial CFD code Fluent v6.x [19], in the manner described in Chapter 3. The chapter includes results for all the single discipline optimisation for sloshing. The optimisation for sloshing includes both the use of 3D and 2D analyses of sloshing, as well as a comparison of different optimisation methodologies using less expensive 2D CFD analyses.

4.2 Definition of Objective Function

One of the major challenges in design optimisation for sloshing is the quantitative evaluation of an objective function. Implications of sloshing include the undesirable acoustical effect experienced in a vehicle fuel tank and issues related to dynamic feedback from oscillating liquids in aeroplanes or liquid transport containers. Ideally, one would like to remove any oscillatory motion, but also reduce the level of free-surface break up. Equation 4.1 below provides the formulation of a Total Deviation Value (TDV) that provides a single value for the level of sloshing that occurred over a period *t*. The value is essentially a numerical integration in time of the deviation of the free surface from its initial position of rest. As illustrated, the value is normalised by the number of computational or discretisation cells involved in the free surface, so that the number of cells encompassing the free surface do not affect the TDV.

$$TDV = \int_0^{t} \left(\sum_{xyz} \frac{\left|\left(surface\ height - initial\ level\right)\right|}{number\ of\ cells\ on\ free\ surface} \right) dt \qquad (4.1)$$

Figure 4.1 below provides an example of the deviation of the free surface versus time, where the TDV represents the area under the curve. The example represents a case without baffles, filled to 70% of capacity and exposed to a constant deceleration that corresponds to the tank decelerating from 60 to 0 km/h in 2 seconds. Figure 4.1 also provides images of the position of the free surface at various moments, for comparison with the corresponding calculated free-surface deviation at that time. The comparison clearly illustrates that the break-up of the free surface around 0.75 seconds causes an increase in the deviation value. Once the surface quietens, as it approaches its position of rest for the constant deceleration, the deviation levels off at the value corresponding to the resulting free-surface inclination. In a case with no baffles, the time to reach a state of equilibrium is quite long, due to the increase in free-surface area as the liquid oscillates. Figure 4.2 below illustrates 3 baffled cases and their corresponding deviations with time. Case 1 shows the worst case resulting from highly oscillatory motion, while case 2 shows an improvement, but free-surface break-up causes a peak near 0.7 seconds. Case 3 shows the best result with the lowest TDV. All three lines will converge eventually to the same value once the free surface has settled.

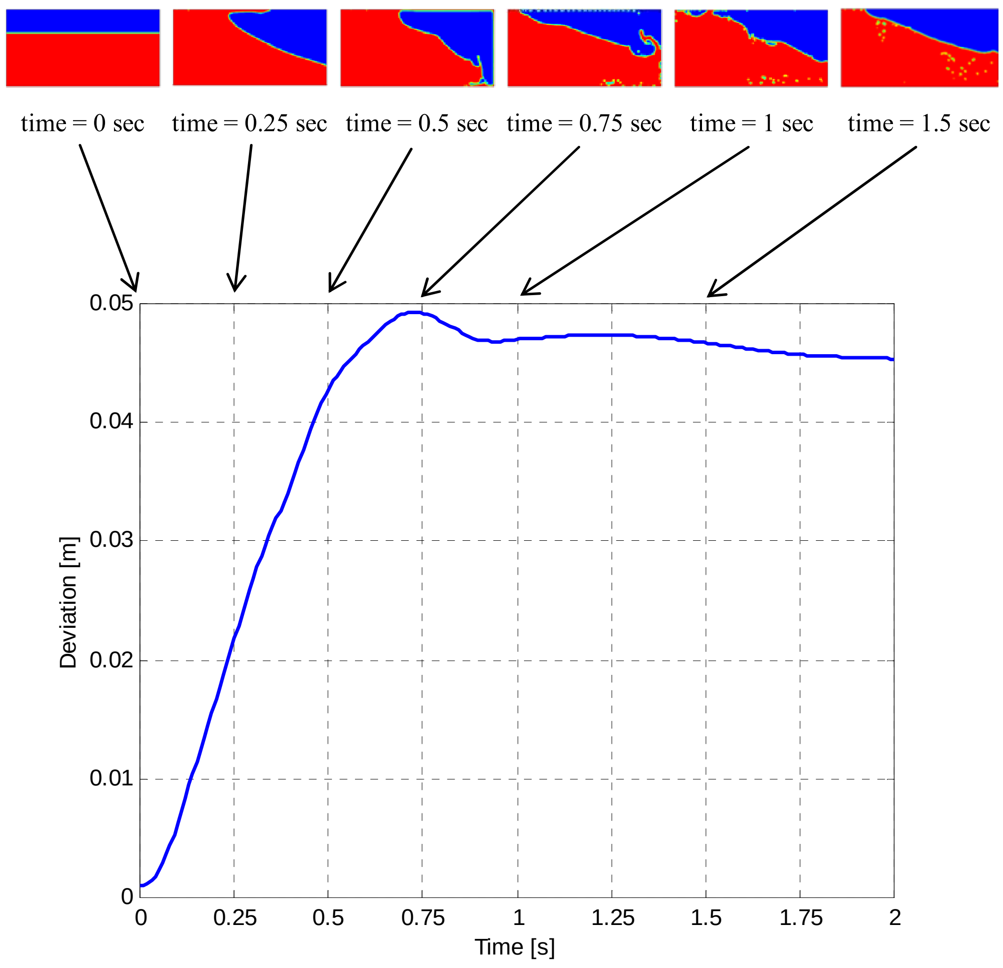


Figure 4.1: Example of free surface deviation versus time

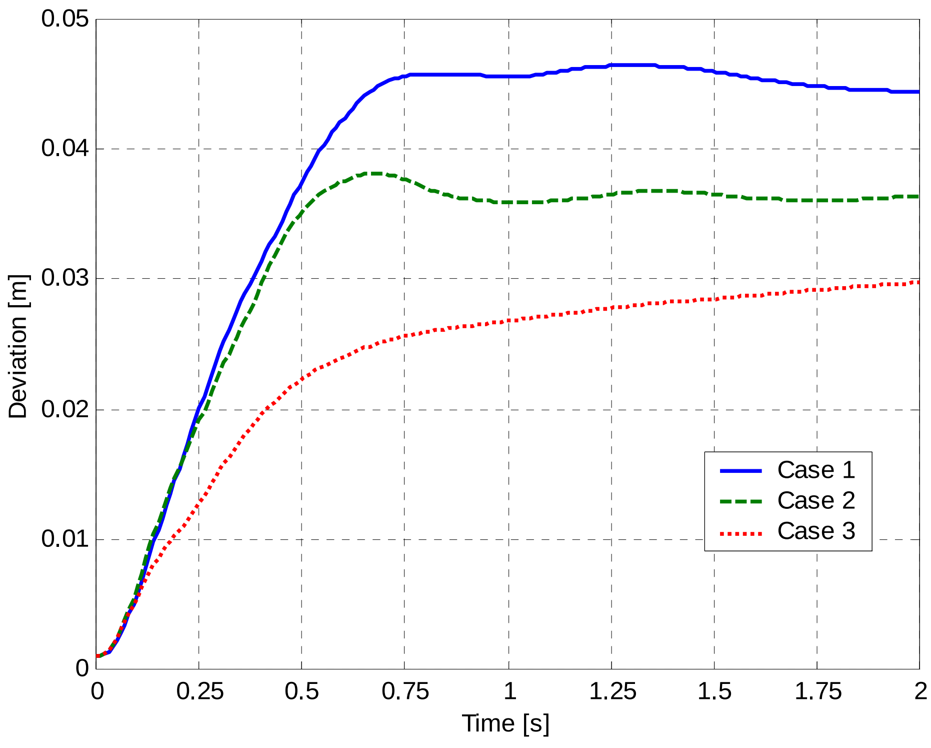


Figure 4.2: Free-surface deviation for 3 baffled cases

In summary, a low TDV results from a case where the free surface approaches its position of equilibrium (for the instantaneous acceleration) slowly, without any break up of the free surface, and without any oscillatory motion.

4.3 Optimisation Problem Setup

The setup of the optimisation problem describes the steps that are taken to ensure the partial or full automation of the optimisation procedure. This section describes the setup for all the optimisation for sloshing problems that are examined. All the cases examined in this section are performed according to the flow chart shown in Figure 2.12.

4.3.1 LS-OPT 3-D Sloshing Case Optimisation

The case examined here is a full 3-D representation of a partially-filled (50%) liquid container with zero-thickness baffles containing holes. Figure 4.3 below (Repeated from Figure 3.1) shows the form of the 3-D model of dimensions WxHxL = 400x400x500mm. The model is subject to a constant deceleration from 60 to 0 km/h in 5 seconds followed by zero deceleration for a further 2 seconds, i.e., the total transient time simulated is 7 seconds.

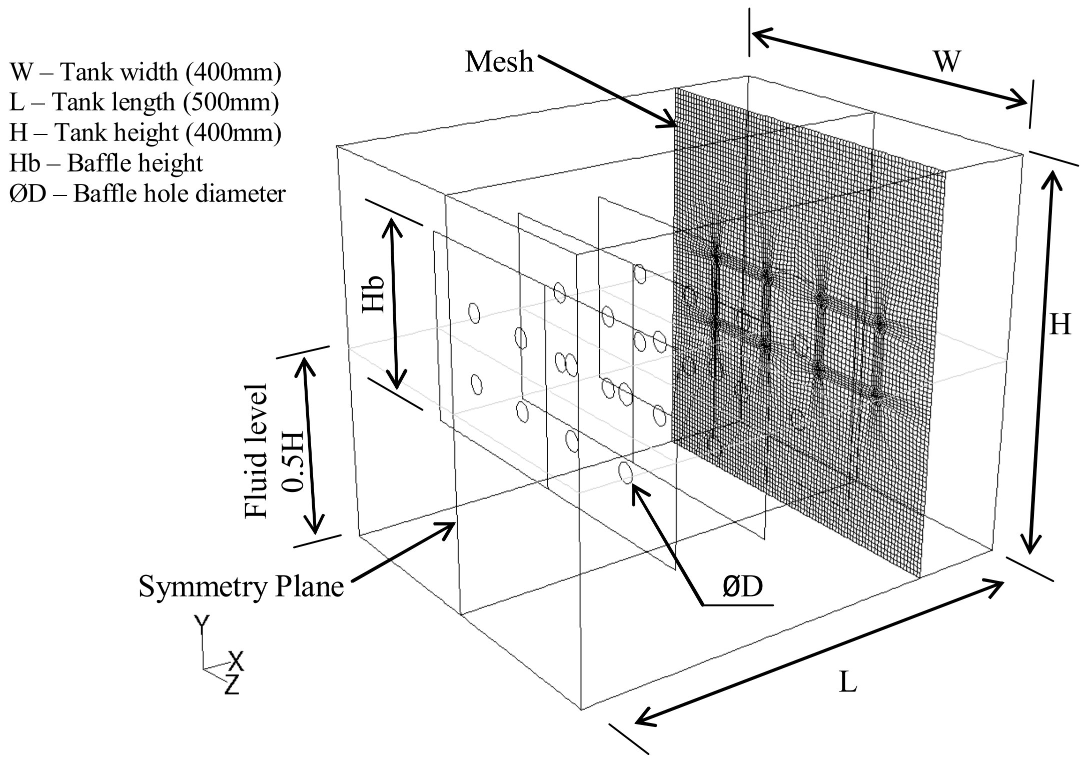


Figure 4.3: Geometry of 3-D sloshing case

This optimisation case involves the use of LS-OPT as the optimisation software, assuming linear successive response surfaces in one case and a Kriging meta-model in another (Appendix F provides the input command file used by LS-OPT). The user will initialise the design points to be analysed for each optimisation iteration. Pre-processing is done using Gambit with a journal file for the automated construction of the mesh (Appendix G provides the Gambit journal file used). A further journal file (Appendix H) is used in the automated setup of the model in Fluent, which models half of the geometry base on the symmetry plane shown in Figure 4.3. Post processing of the data generated by Fluent to extract the TDV is done by an executable file compiled from the C source code shown in Appendix I.

To complete a single Fluent analysis of a 3-D sloshing event takes approximately 48 hours on the available workstations (2GHz P4 Linux Workstations). Due to this long simulation time, only two design variables are considered in the optimisation. The optimisation problem is defined according to Equation 4.2 below:

$$\textit{Variables}\ \mathbf{x} = [x_1, x_2]^T \qquad (4.2)$$

x_1 = Hb = Baffle height

x_2 = ØD = Hole diameter

Objective:

Minimize {f(**x**) = TDV}

Subject to:

Inequality constraints-

$$g_1(\mathbf{x}) = 10\text{mm} - \frac{x_1 - 4x_2}{5} < 0$$

$$g_2(\mathbf{x}) = \frac{x_1 - 2x_2}{3} - \frac{x_2}{2} - 8\text{mm} < 0$$

$$g_3(\mathbf{x}) = 400x_1 - 8\pi . x_2^2 - 95000\text{mm}^2 < 0$$

Side constraints-

$g_4(\mathbf{x})$: x_1 (Hb) (80;380)mm

$g_5(\mathbf{x})$: x_2 (ØD) (15;80)mm

Inequality constraints g_1 and g_2 are necessary to ensure that the model is geometrically feasible, e.g., that the holes do not get too large for the baffles, etc. The side constraints g_4 and g_5 represent the limits on the variables. Inequality constraint g_3 is put in place in anticipation of the design going toward large baffles with small holes. In this case one would like to make the problem more interesting by restricting the amount of material used (related to mass of baffles or cost of production). In this case arbitrarily chosen as 95 000mm^2. This case formed part of a study done in collaboration with R Dieterich in 2002 [53].

4.3.2 2-D Sloshing Cases Setup

Due to the expense of full 3-D CFD analyses, further analysis involved a 2-D simplification of the liquid container. A number of cases are examined, all involving one of two topological layouts that will be referred to as design 1 and design 2. Figure 4.4 and Figure 4.5 below show the geometries used in designs 1 and 2 respectively. All cases are subject to the same load derived from a constant deceleration from 60 to 0 km/h in 2 seconds (Appendix J provides a sample of the User-Defined Function (UDF) c-code used for a predefined acceleration). As in the 3-D case, pre-processing is done by Gambit with the use of a journal file (See sample 2-D Gambit journal in Appendix A) and the Fluent setup is done with its respective journal file (See sample 2-D Fluent journal in Appendix C). Appendix K provides the 2-D-adapted c source code for compiling the TDV extractor program.

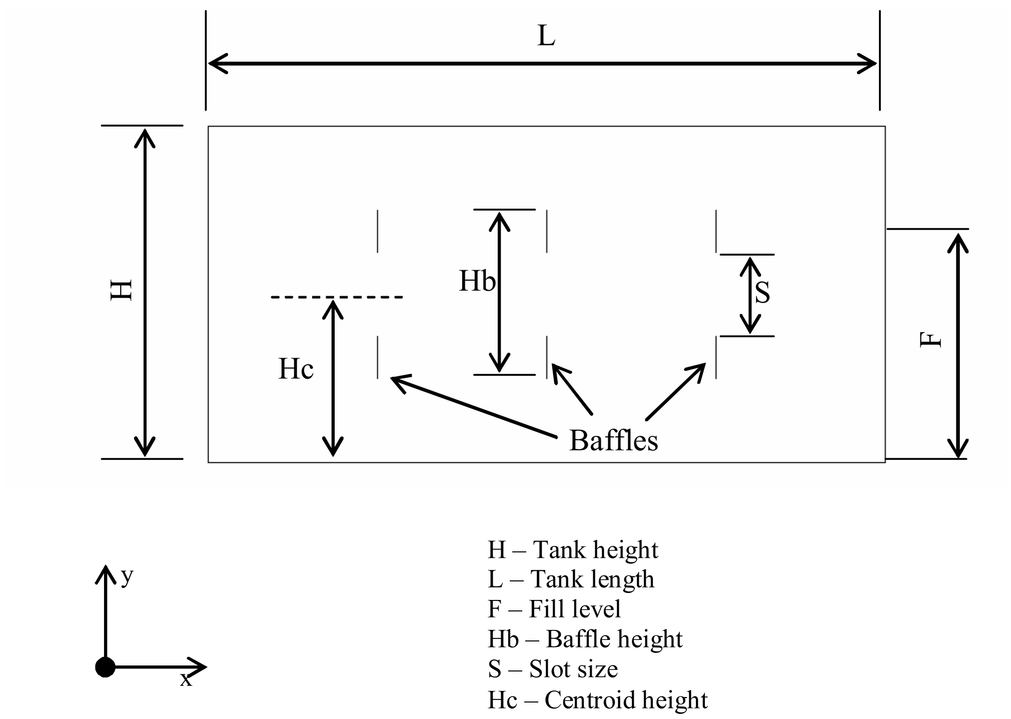


Figure 4.4: Geometry of 2-D container: Design-1

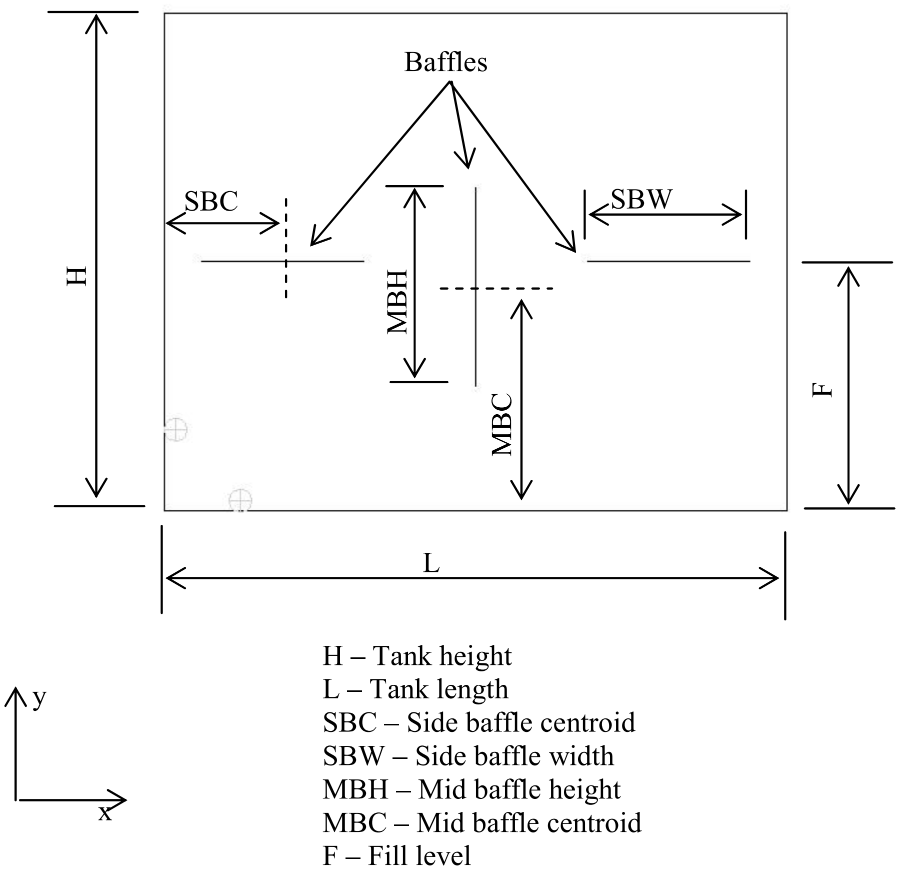


Figure 4.5: Geometry of 2-D container: Design-2

Table 4.1 below provides the general formulation of the 2-D optimisation cases considered. The abbreviations for the design variables are consistent with those used in Figure 4.4 and Figure 4.5 above. The sections that follow provide more in depth descriptions of the individual cases.

Table 4.1: Definition of optimisation cases (2D sloshing)

Case	Design type (HxL)	Fill	Optimisation method	Design variables	Constraints	Evaluations per iteration
1	**1** (200x400mm)	0.7H	Linear SRSM	$\mathbf{x}_{(1\text{-}3)}$= (Hc,Hb,S)	-Geometrical	7
2	**1** (200x400mm)	0.7H	Quadratic SRSM	$\mathbf{x}_{(1\text{-}3)}$= (Hc,Hb,S)	-Geometrical	16
3	**1** (200x400mm)	0.7H	Neural Network	$\mathbf{x}_{(1\text{-}3)}$= (Hc,Hb,S)	-Geometrical	129
4	**1** (200x400mm)	0.7H	Dynamic-Q	$\mathbf{x}_{(1\text{-}3)}$= (Hc,Hb,S)	-Geometrical	4
5	**2** (200x400mm)	0.7H	Quadratic SRSM	$\mathbf{x}_{(1\text{-}4)}$= (MBC,SBC, MBH,SBW)	-Geometrical	23
6	**2b** (400x500mm)	0.7H	Linear SRSM	$\mathbf{x}_{(1\text{-}4)}$= (MBC,SBC, MBH,SBW)	-Geometrical	8
7	**2** (200x400mm)	x_5*H (variable)	Quadratic SRSM	$\mathbf{x}_{(1\text{-}5)}$= (F,MBC,SBC, MBH,SBW)	-Geometrical -Fill level	32

4.3.2.1 Linear LS-OPT Design 1 (Case 1)

This case makes use of the Linear SRSM within the LS-OPT framework. Appendix L provides the LS-OPT command file for this case. The optimisation problem is defined as in Equation 4.3 below.

$$\textit{Variables } \mathbf{x} = [x_1, x_2, x_3]^T \qquad (4.3)$$

x_1 = Hc = Baffle centroid height

x_2 = Hb = Baffle height

x_3 = S = Slot size

Objective:

Minimize $\{f(\mathbf{x}) = TDV\}$

Subject to:

Inequality constraints-

$$g_1(\mathbf{x}) = -(190 - 0.5*x_2) + x_1 < 0$$
$$g_2(\mathbf{x}) = (10 + 0.5*x_2) - x_1 < 0$$
$$g_3(\mathbf{x}) = x_3 - 0.8*x_2 < 0$$

Side constraints-

$g_4(\mathbf{x})$: x_1 (Hc) (10;190)mm

$g_5(\mathbf{x})$: x_2 (Hb) (20;180)mm

$g_6(\mathbf{x})$: x_3 (S) (10;140)mm

Inequality constraints g_1 and g_2 ensure that the baffle does not come too close to either the roof or the floor of the container respectively. Inequality constraint g_3 ensures that the slot in the baffle does not become greater than 80% of the baffle height. The side constraints represent the limits of the variables. A linear SRSM is the simplest response surface method that makes use of 50% over-sampling provided by LS-OPT, in that it requires the least analyses per optimisation iteration.

4.3.2.2 Quadratic LS-OPT Design 1 (Case 2)

This case makes use of the quadratic successive response surface method within the LS-OPT framework. Appendix M provides the LS-OPT command file. The problem is defined exactly as in equation 4.3 above. Although the Quadratic SRSM requires more analyses per optimisation iteration, its performance should be evaluated against that of the Linear SRSM due to a possible bias or modelling error reduction.

4.3.2.3 Neural Network LS-OPT Design 1 (Case 3)

This case makes use of a single neural network meta-model within the LS-OPT framework. Appendix N provides the LS-OPT command file. In an attempt to construct a response surface representing a larger (more global) region of the 3-D design space, the user will provide all the points to be used for the single optimisation iteration, as well as their results. 129 analysis results are provided as obtained during the case 1, *4.3.2.2 quadratic LS-OPT design 1* optimisation analysis. The problem is defined exactly as in equation 4.3 above. The neural network result will provide an alternative answer to the Linear and Quadratic SRSMs from the perspective of a more global optimisation technique.

4.3.2.4 Dynamic-Q TDO Design 1 (Case 4)

This case makes use of the Dynamic-Q method within the TDO framework. Each analysis is initialised by the user. The use of TDO and Dynamic-Q is motivated by the desire to compare LS-OPT's meta-modelling methods with a gradient-based method as provided by TDO. The problem is defined exactly as in equation 4.3 of case 1, *4.3.2.1 Linear LS-OPT design 1* above.

4.3.2.5 Quadratic LS-OPT Design 2 (Case 5)

This case makes use of the quadratic successive response surface method within the LS-OPT framework. Appendix O provides the LS-OPT command file. The design 2 topology is motivated by the suggestion made during Rodriguez's studies of 1952 [1] that proposes locating flow dampers in the regions of highest velocity. The results for TDV should be compared with those obtained in the previous four optimisation analyses. The problem is defined as in Equation 4.4 below.

$$\textit{Variables}\ \mathbf{x} = [x_1, x_2, x_3, x_4]^T \qquad (4.4)$$

x_1 = MBC = mid baffle centroid

x_2 = SBC = side baffle centroid

x_3 = MBH = mid baffle height

x_4 = SBW = side baffle width

Objective:

$$\min \{f(\mathbf{x}) = \text{TDV}\}$$

Subject to:

Inequality constraints-

$$g_1(\mathbf{x}) = - x_2 + 0.5*x_4 + 10\text{mm} < 0$$
$$g_2(\mathbf{x}) = x_2 + 0.5*x_4 – 190\text{mm} < 0$$
$$g_3(\mathbf{x}) = x_1 + 0.5*x_3 – 190\text{mm} < 0$$
$$g_4(\mathbf{x}) = - x_1 + 0.5*x_3 + 10\text{mm} < 0$$

Side constraints-

$g_5(\mathbf{x})$: x_1 (MBC) (15;185)mm

$g_6(\mathbf{x})$: x_2 (SBC) (15;185)mm

$g_7(\mathbf{x})$: x_3 (MBH) (10;180)mm

$g_8(\mathbf{x})$: x_4 (SBW) (10;180)mm

As before, all constraints are required for geometric feasibility.

4.3.2.6 Linear LS-OPT Design 2b (Case 6)

This case makes use of the linear successive response surface method within the LS-OPT framework. Appendix P provides the LS-OPT command file. The motivation for this analysis will only become clearer during the MDO analysis of chapter 6, where larger containers (corresponding to those used during validation phase 2) are analysed. The problem is defined much as in case 5, *section 4.3.2.5 Quadratic LS-*

OPT design 2 above except for the following constraints (Equation 4.5) that change due to the change in the dimensions of the container:

Inequality constraints- (4.5)

$$g_1(\mathbf{x}) = - x_2 + 0.5*x_4 + 10mm < 0$$
$$g_2(\mathbf{x}) = x_2 + 0.5*x_4 - 190mm < 0$$
$$g_3(\mathbf{x}) = x_1 + 0.5*x_3 – 340mm < 0$$
$$g_4(\mathbf{x}) = - x_1 + 0.5*x_3 + 40mm < 0$$

Side constraints-

$$g_5(\mathbf{x}): x_1 \text{ (MBC) } (60;320)\text{mm}$$
$$g_6(\mathbf{x}): x_2 \text{ (SBC) } (15;185)\text{mm}$$
$$g_7(\mathbf{x}): x_3 \text{ (MBH) } (40;320)\text{mm}$$
$$g_8(\mathbf{x}): x_4 \text{ (SBW) } (10;180)\text{mm}$$

4.3.2.7 Quadratic LS-OPT Saddle Design 2 (Case 7)

This case makes use of the quadratic successive response surface method within the LS-OPT framework. Appendix Q provides the LS-OPT command file. The motivation for this study is to analyse the effect of the fill level (H), previously assumed as 70% full, on the performance of the baffles. The setup will attempt to maximise TDV w.r.t. the fill level while minimising TDV w.r.t. the remaining variables, to try and establish the worst case. The problem is defined as follows in equation 4.6 below, with the definition making use of a saddle point analysis.

Variables $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$ (4.6)

$$x_1 = F = \text{fill level}$$
$$x_2 = MBC = \text{mid baffle centroid}$$
$$x_3 = SBC = \text{side baffle centroid}$$
$$x_4 = MBH = \text{mid baffle height}$$

x_5 = SBW = side baffle width

Objective:

$$\underset{x_{i,i=1-4}}{\min}\left\{\underset{x_5}{\max}\left(f(\mathbf{x}) = TDV\right)\right\}$$

Subject to:

Inequality constraints-

$g_1(\mathbf{x}) = - x_2 + 0.5*x_4 + 10\text{mm} < 0$

$g_2(\mathbf{x}) = x_2 + 0.5*x_4 - 190\text{mm} < 0$

$g_3(\mathbf{x}) = x_1 + 0.5*x_3 - 190\text{mm} < 0$

$g_4(\mathbf{x}) = - x_1 + 0.5*x_3 + 10\text{mm} < 0$

Side constraints-

$g_5(\mathbf{x})$: x_1 (MBC) (15;185)mm

$g_6(\mathbf{x})$: x_2 (SBC) (15;185)mm

$g_7(\mathbf{x})$: x_3 (MBH) (10;180)mm

$g_8(\mathbf{x})$: x_4 (SBW) (10;180)mm

$g_9(\mathbf{x})$: x_5 (F) (10;190)mm

As before, all constraints are required for geometric feasibility.

4.4 Optimisation Results

The following section provides the results of the optimisation analyses described in section 4.3 above. Results will include the improvement of the design w.r.t. its starting value as well as the progression of all the variables and objective during the optimisation process. The various optimisation methods will be compared with each other on the basis of their results and how economically they are attained.

4.4.1 LS-OPT 3-D Sloshing Case Optimisation

As stated in section 4.3.1 this case formed part of a study conducted in collaboration with R Dieterich in 2002 [52]. As stated before, this study involved a Linear SRSM approach and a Kriging meta-model approach. Table 4.2 below gives the final results for both approaches. Figure 4.6 below illustrates the progress of the Linear RSM optimisation run. The figure does show that the solution converged after one iteration, however the result of the Kriging optimisation method suggests that the local optimum lies elsewhere. This confirms that further linear SRSM optimisation iterations are necessary to locate the true local optimum.

Table 4.2: 3-D sloshing LS-OPT optimisation results

	Starting value	Converged linear SRSM result	CFD computed	Converged Kriging model result	CFD computed
Hb (x_1) [mm]	300	80	80	120.9	120.9
ØD (x_2) [mm]	40	15	15	27.7	27.7
***f*10^4* [mm^2.s]**	27.10	20.03	21.31	19.02	**19.87**

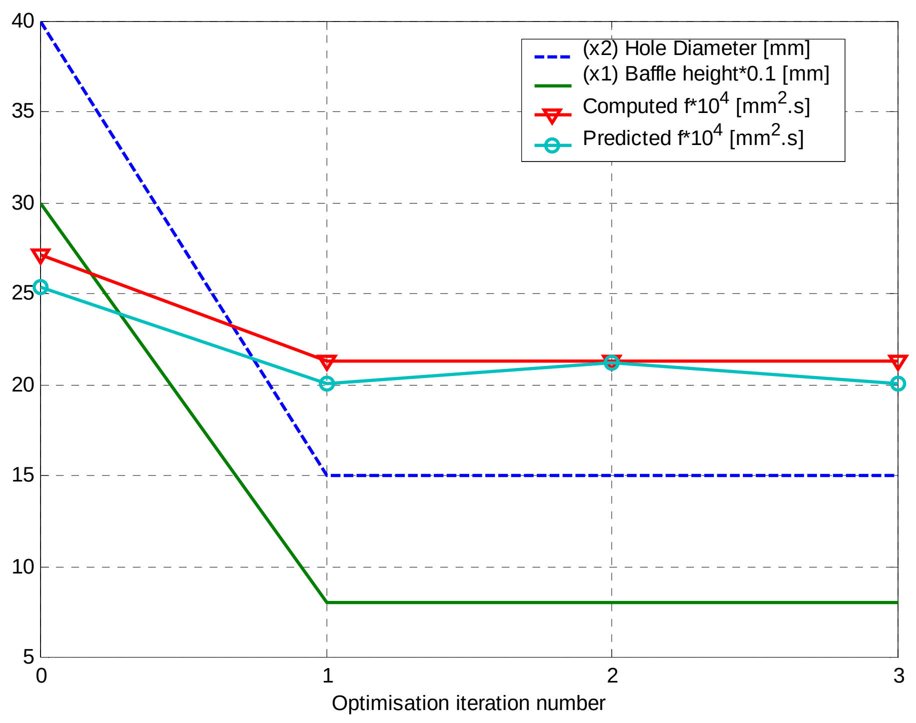


Figure 4.6: Linear SRSM optimisation history (3D sloshing)

The Kriging model is constructed from the points listed in Table 4.3 below, some of which come from the Linear RSM optimisation run. The resulting Kriging surface can be seen in Figure 4.7 below. Since this is an optimisation case with only 2 variables, it is quite simple to visualise a response surface like the one created by the Kriging interpolation method. The advantage of this method is that one can get a more global perspective on the response in question. As one can see an improved optimum existed slightly further along the active second inequality constraint (gap between hole and baffle edge). (Linear optimum shown for comparison on Figure 4.7)

Table 4.3: Points used for Kriging meta-model

Hb [mm]	80	139.15	147.5	170	170	187.7	200	200
ØD [mm]	15	15	35.29	15	35	15	15	20
$f*10^4$ [mm^2.s]	21.31	23.15	19.68	23.99	20.88	24.53	24.80	23.63

Hb [mm]	230	236.6	300	339.3	339.4	380	380
ØD [mm]	59.7	15	40	15	64	15	80
$f*10^4$ [mm^2.s]	24.66	25.54	27.13	25.26	31.07	20.47	34.17

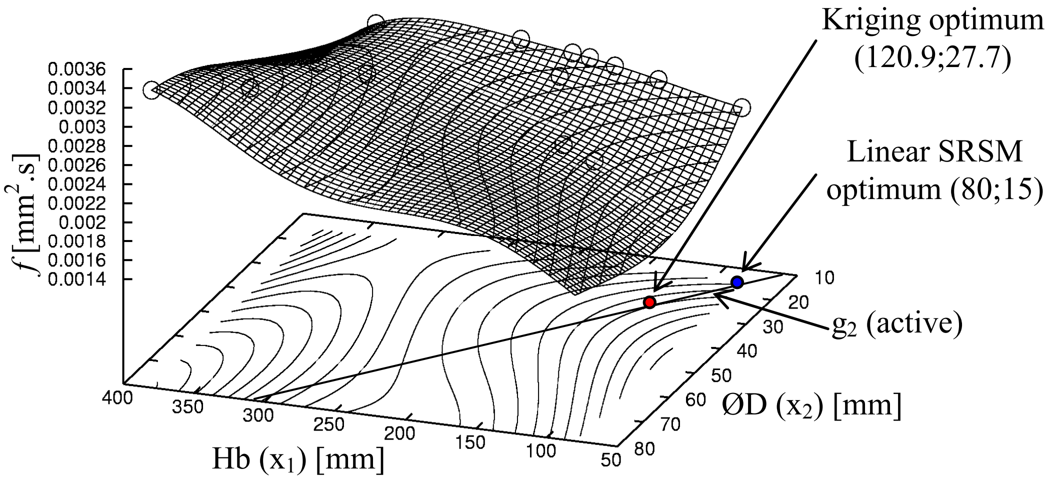


Figure 4.7: Kriging surface of 3-D sloshing case (with permission J Haarhoff)

The behaviour of the objective is clearly non-linear in the region of the optimum, which would explain the inability of the Linear RSM to locate it early in the optimisation process. However, it should be said that the Linear RSM would most likely have eventually found the global optimum once the design sub-region had been sufficiently reduced. This case makes a strong argument for the use of perhaps a

Quadratic RSM or a global meta-model like Kriging which is able to handle nonlinearities and thus reduce bias error.

Upon inspecting the objective response in Figure 4.7 one can make some conclusions about the effectiveness of baffles. Large baffles with small holes perform relatively poorly due to an effect best described as compartmentalisation. In effect the liquid container is merely subdivided into a number of smaller containers that have very little flow damping ability. The idea is to reduce the energy in the liquid, and since the highest velocities are seen near the free surface, this is where the flow damping should be done. This idea is confirmed with the result of the optimisation being small baffles with medium size holes. The small baffles allow flow past them so as not to compartmentalise the container, but still induces flow losses as the liquid passes over or under the baffle. The medium size holes continue this principle since the smallest holes will most likely not allow much flow through them and as a result not induce much damping. The larger holes will in turn allow too much liquid through an also not provide as much damping.

As a final comment on this optimisation case, the Linear RSM method required 15 function evaluations to attain a 21.3% improvement in the objective (from an arbitrary starting design), while the Kriging method attained a 26.7% improvement with the same number of function evaluations. It would also appear that the Kriging optimum is near the local minimum for this case, but would require more function evaluation (specifically near the area of the local optimum) to provide an accurate representation of the response.

4.4.2 2-D Sloshing Optimisation Results

4.4.2.1 Linear LS-OPT Design 1 (Case 1)

Table 4.4 below shows the final results for the Linear RSM optimisation of a design 1 2-D container. The result for TDV represents an improvement of 37.8% and required 50 function evaluations in 7 optimisation iterations. Figure 4.8 below shows the progress of the optimisation versus optimisation iteration number. Although this plot represents the results obtained from 70 function evaluations (10 iterations), the optimum was found after the 7th iteration. Only one inequality constraint is active at the optimum and that is $g_2(\mathbf{x})$, which prevents the baffle from moving too low. It is interesting to note that the side constraint for the hole size is also active (Minimum slot size S (x_3) = 10mm). This indicates that the 2-D assumption applied to the model, resulting in a slot instead of a hole, provides very little flow damping. Furthermore, the design trend suggests a medium size baffle, the upper half of which tries to intercept the high velocities of the free surface. In effect, the lower half of the baffle is probably somewhat inactive. For the purposes of later comparison it should also be stated that the total length of baffle used (for all the baffles) is 373.8mm.

Table 4.4: Final results for Linear RSM design 1

	Starting value	Converged linear RSM result
Hc (x_1) [mm]	100	77.28
Hb (x_2) [mm]	100	134.6
S (x_3) [mm]	50	10
***TDV* *(f*10^4)* [m.s]**	36.46	**22.66**

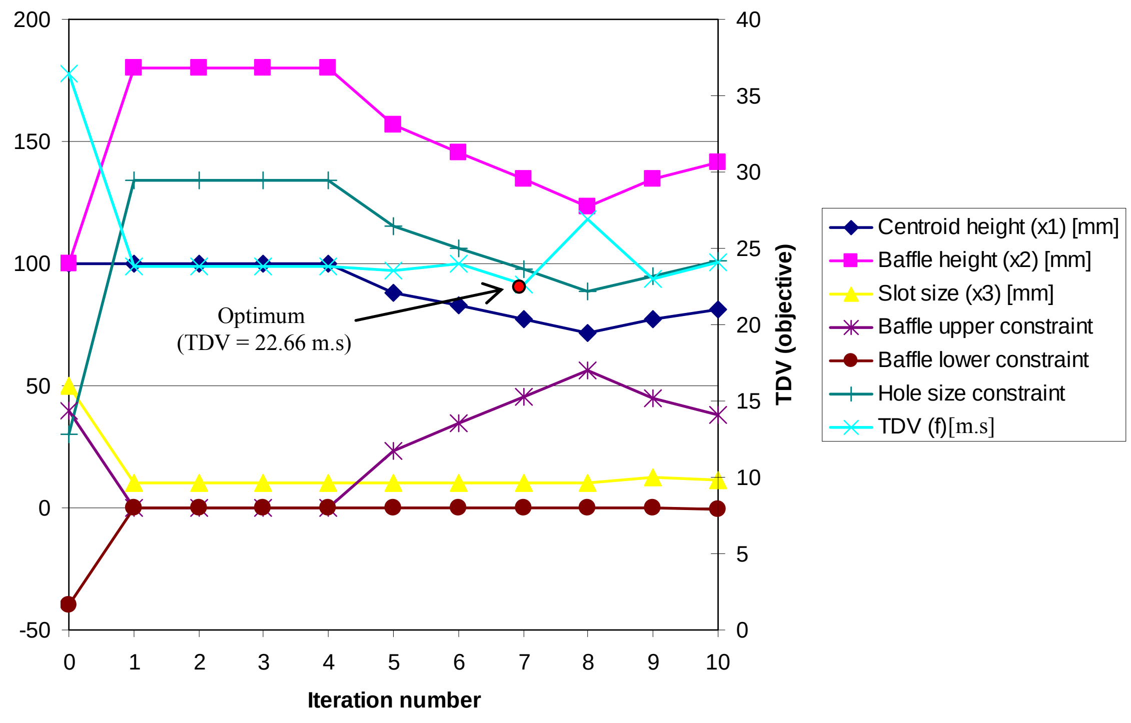


Figure 4.8: Optimisation history for Linear RSM design 1

When examining the 1st four optimisation iterations in Figure 4.8 above, it is clear that the non-linear behaviour of the problem only becomes evident to the optimisation algorithm once the design subspace has been sufficiently reduced in size. Figure 4.9 below gives a good illustration of the domain reduction for the variable x_1 (Baffle centroid). The result does however suggest a significant bias error due to the linear response assumption when applied to the initial large domain.

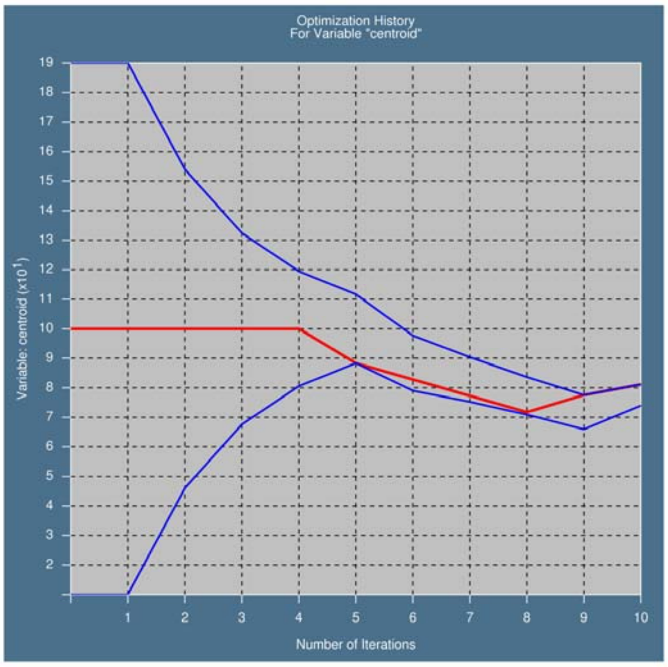


Figure 4.9: Design domain/subspace reduction for Baffle centroid (x_1)

4.4.2.2 Quadratic LS-OPT Design 1 (Case 2)

Table 4.5 below shows the final results for the Quadratic RSM optimisation of a design 1 2-D container. The result for TDV represents an improvement of 36.2% and required 128 function evaluations in 8 optimisation iterations for convergence. Figure 4.10 below shows the progress of the optimisation versus optimisation iteration number. The results are very similar to those obtained in the Linear RSM optimisation, although the Quadratic RSM run required 45.3% more function evaluations. What is clear from Figure 4.10 is that the Quadratic response assumption exhibited far less bias error side effects. The variables start converging toward the optimum almost immediately but approached it more slowly than the Linear RSM optimisation run.

Table 4.5: Final results for Quadratic RSM design 1

	Starting value	Converged linear RSM result
Hc (x_1) [mm]	100	76.68
Hb (x_2) [mm]	100	133.6
S (x_3) [mm]	50	10
***TDV* (*$f*10^4$*) [m.s]**	36.46	**23.26**

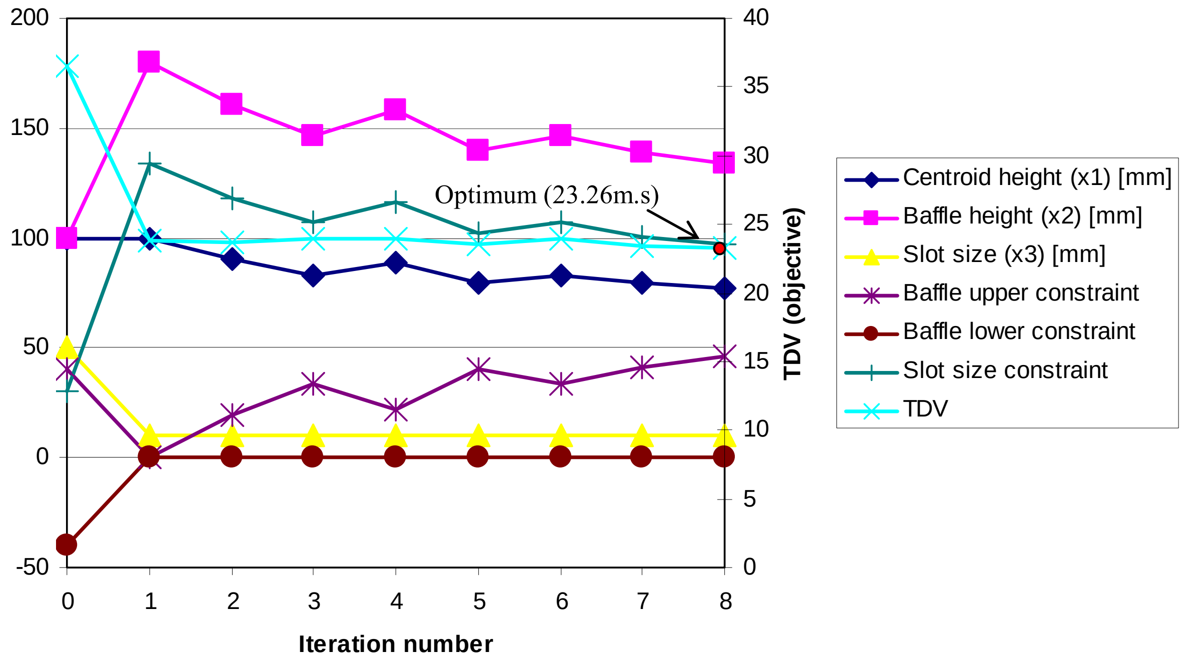


Figure 4.10: Optimisation history for Quadratic RSM design 1

An interesting analysis of the results is provided when examining a trade off between responses or variables. In particular, a trade-off study was performed between the objective TDV and the length of baffle used. The latter could be equated to the cost of the materials used. Figure 4.11 below illustrates the trade-off curves generated for each optimisation iteration. The trade-off curve (approximated Pareto-optimal front) is generated by linking a specified number of points that are the predicted optimum results for the objective (TDV) for the response value in question (baffle length in this case) from the data received during a particular iteration. Some interesting observation can be made about trade-off curves. The curve provides the observer with the opportunity to decide what level of performance is required and how much material he/she is willing to use. From this perspective, the 1st three curves provide the most meaningful data. Due to the domain subspace reduction, the curves that follow don't provide much of a global perspective and are only valid in the immediate region of the subdomain optimum. Considering the trade-off curve illustrated for iteration 2, to achieve a TDV = 36 m.s, then 125 mm of baffle is required. If however the TDV needs to be reduced to TDV = 24 m.s, then 500 mm of baffle is required.

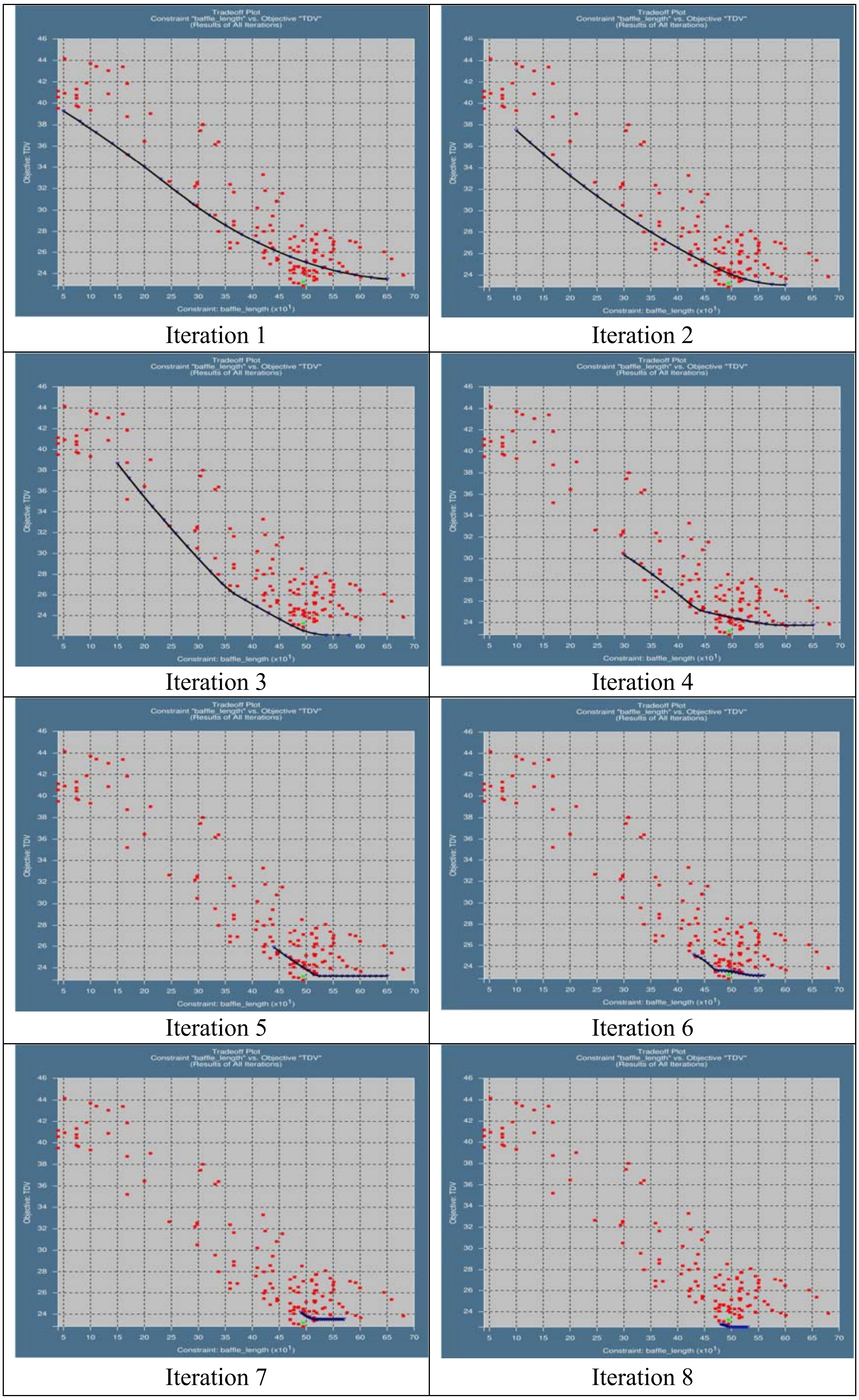


Figure 4.11: Trade-off progress for Quadratic RSM design 1

The final trade-off curve (iteration 8) gives more accurate data in terms of the absolute optimum, but when considering the potential use of trade-off curves is perhaps unnecessary to consider. In fact, if your objective is to do a trade-off study, it is only necessary to complete two or three optimisation iterations.

However since this is an optimisation study, the points of interest include that the maximum length of baffle available does not provide the absolute optimum.

4.4.2.3 Neural Network LS-OPT Design 1 (Case 3)

Table 4.6 below shows the final results for the Neural Network optimisation of a design 1 2-D container. The result for TDV represents an improvement of 27.6% and utilised 128 function evaluations in a single optimisation iteration. The total length of baffle used is 349.5mm.

Table 4.6: Final results for Neural Network Design 1

	Starting value	Converged linear RSM result
Hc (x_1) [mm]	100	73.24
Hb (x_2) [mm]	100	126.5
S (x_3) [mm]	50	10
***TDV* (*$f*10^4$*) [m.s]**	36.46	**26.41**

4.4.2.4 Dynamic-Q TDO Design 1 (Case 4)

Table 4.7 below shows the final results for the TDO Dynamic-Q optimisation of a design 1 2-D container. The result for TDV represents an improvement of 36.4% and utilised 40 function evaluations in 10 optimisation iterations. Figure 4.12 below shows the progress of the optimisation versus optimisation iteration number. Due to the differing nature of the Dynamic-Q method (i.e., spherically quadratic subproblem with move limits), the progress of the algorithm is clearly different in nature. The results are consistent with those seen previously with the methods available in LS-OPT. The total length of baffle used is 378mm.

Table 4.7: Final results for Dynamic-Q Design 1

	Starting value	Converged linear RSM result
Hc (x_1) [mm]	100	78
Hb (x_2) [mm]	100	136.0
S (x_3) [mm]	50	10
***TDV* *($f*10^4$)* [m.s]**	36.46	**23.18**[*]

[*] The results for TDV during the optimisation process where calculated using Fluent v5, but were confirmed with Fluent v6, with the results from the former being consistently lower (22.88 optimum versus 23.18).

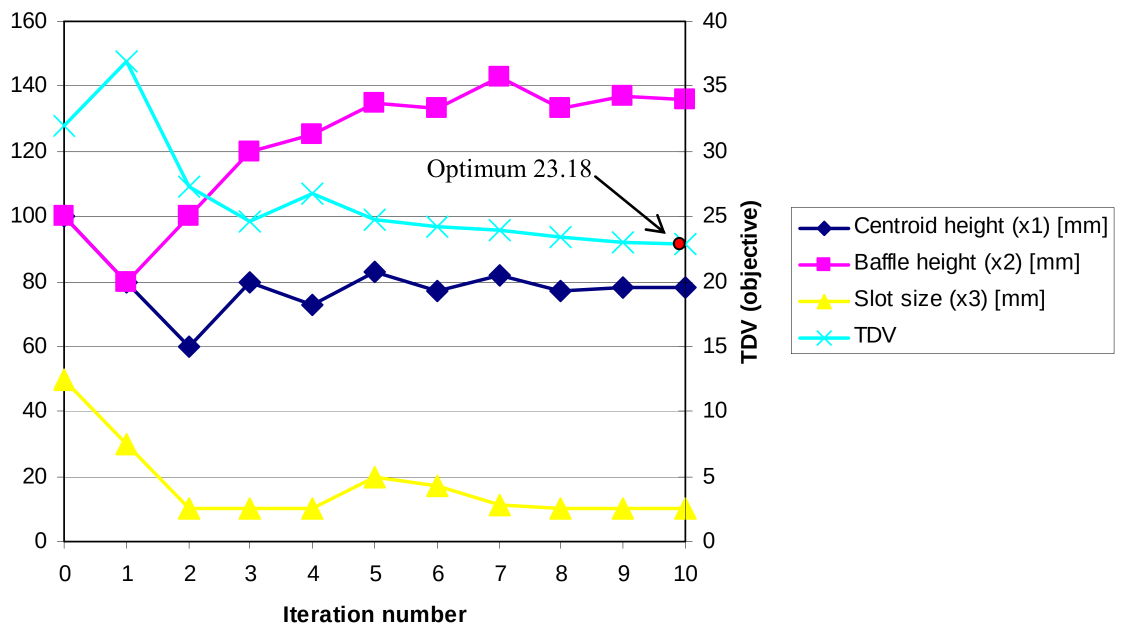


Figure 4.12: Optimisation history for Dynamic-Q design 1

Although the method requires significantly less function evaluations, it should be mentioned that a significant level of user intervention was necessary. Since the method is a gradient-based method it is susceptible to becoming unstable when noise enters the gradient calculations. A remedy to this instability is to impose move limits for the variables to prevent the divergence of the algorithm and to prevent it from overshooting local minima. The choice of the size of these move limits is not typically intuitive beforehand. A small move limit may also result in a very long convergence time. A further value that needs specification is the perturbation size for the finite difference gradient calculation. The incorrect choice of this value may result in the algorithm being adversely affected by the noise exhibited in most engineering problems. Ultimately, response surface methods illustrated significantly more robustness in the problem of optimisation for sloshing.

4.4.2.5 Quadratic LS-OPT Design 2 (Case 5)

Table 4.8 below shows the final results for the Quadratic RSM optimisation of a design 2 2-D container. The result for TDV represents an improvement of 16.7% and utilised 184 function evaluations in 8 optimisation iterations. Figure 4.13 below shows the progress of the optimisation versus optimisation iteration number. The total length of baffle used is 418.8mm. The only active inequality constraint for this design is $g_4(\mathbf{x})$, which prevents the middle baffle from moving too low.

Table 4.8: Final results for Quadratic RSM design 2

	Starting value	Converged linear RSM result
MBC (x_1) [mm]	100	80.8
SBC (x_2) [mm]	100	79.29
MBH (x_3) [mm]	100	141.6
SBW (x_4) [mm]	100	138.6
***TDV* (*f*10^4*) [m.s]**	27.03	**22.52**

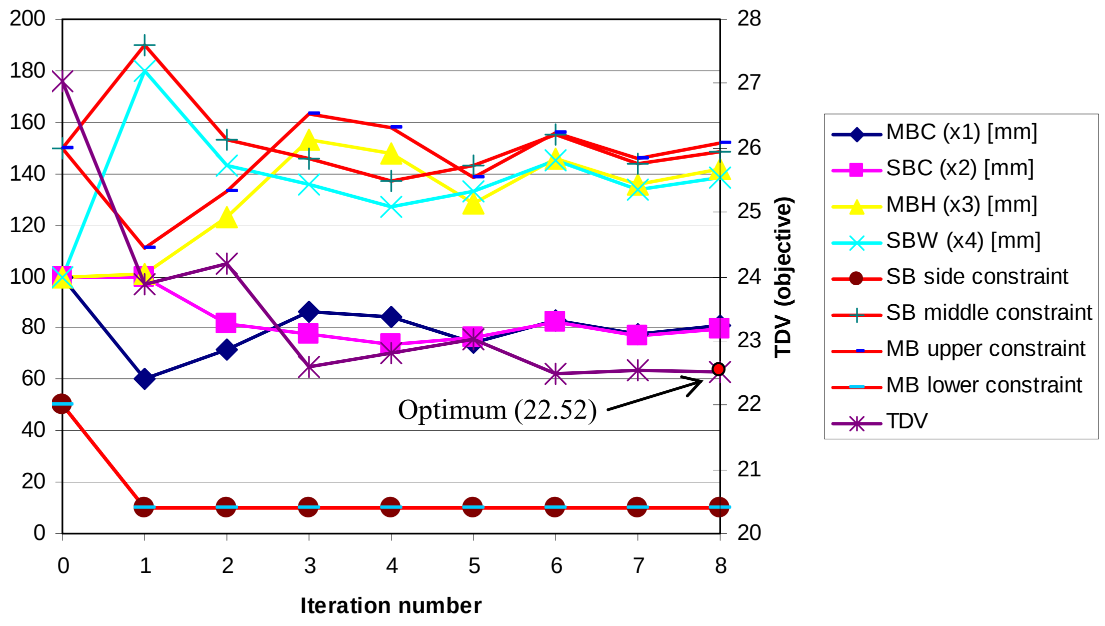


Figure 4.13: Optimisation history for Quadratic RSM design 2

The final result does provide a slight improvement for the optimum when compared to the design 1 container. However, the small improvement in TDV relative to the starting value does suggest a better overall design (one vertical and two horizontal baffles versus vertical slotted baffles).

4.4.2.6 Linear LS-OPT Design 2b (Case 6)

Table 4.9 below shows the final results for the Linear RSM optimisation of a design 2b 2-D container. The result for TDV represents an improvement of 40.3% and utilised 81 function evaluations in 10 optimisation iterations. Figure 4.14 below shows the progress of the optimisation versus optimisation iteration number. The total length of baffle used is 519mm. These results will be considered in conjunction with results in Chapter 5, when examining MDO of the liquid container in Chapter 6. The reason for repeating the 2D sloshing analysis for this case is to obtain a new TDV value for a higher container (400mm as apposed to 200mm). TDV is influenced by the height of the container since more space above the initial free-surface level will allow more deviation of the active free surface during the sloshing event from its initial location. Appendix R shows comparative frames, during the sloshing analysis, for the base case and the final design.

Table 4.9: Final results for Linear RSM design 2b

	Starting value	Converged linear RSM result
MBC (x_2) [mm]	100	162
SBC (x_3) [mm]	100	78
MBH (x_4) [mm]	100	244
SBW (x_5) [mm]	100	136
***TDV* *($f*10^4$)* [m.s]**	49.17	**29.37**

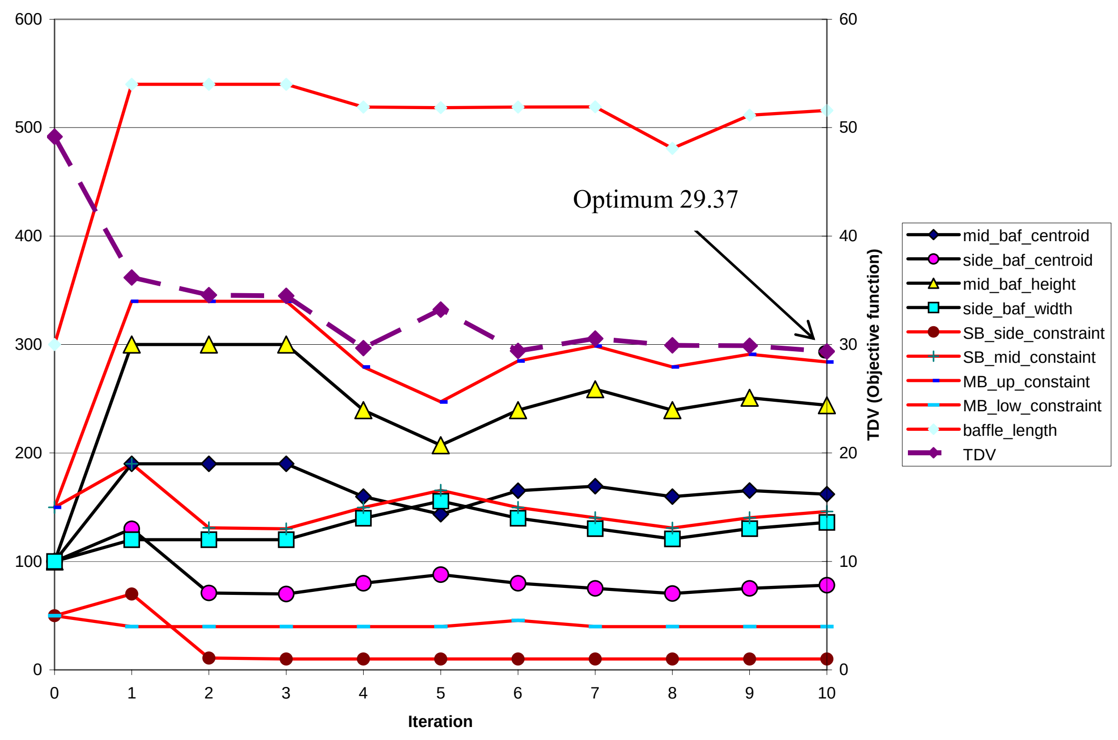


Figure 4.14: Optimisation history for Linear RSM Design 2b

4.4.2.7 Quadratic LS-OPT Saddle Point Design 2 (Case 7)

Table 4.10 below shows the final results for the Quadratic RSM saddle-point optimisation of a design 2 2-D container. The result for TDV represents an improvement of 27.8% and utilised 257 function evaluations in 8 optimisation iterations. Figure 4.15 below shows the progress of the optimisation versus optimisation iteration number. The total length of baffle used is 442.8mm. These optimisation analysis results provide some level of confidence in the final design variable values attained, since the result for TDV represent the worst TDV that would occur for the set of variable values that the optimisation algorithm converged to.

Table 4.10: Final results for Quadratic RSM saddle point Design 2

	Starting value	final linear RSM result
F (x_1) [m]	0.1 (50% full)	0.153 (77% full)
MBC (x_2) [mm]	100	83.7
SBC (x_3) [mm]	100	86
MBH (x_4) [mm]	100	139
SBW (x_5) [mm]	100	151.9
TDV ($f*10^4$) [m.s]	33.56	**24.23**

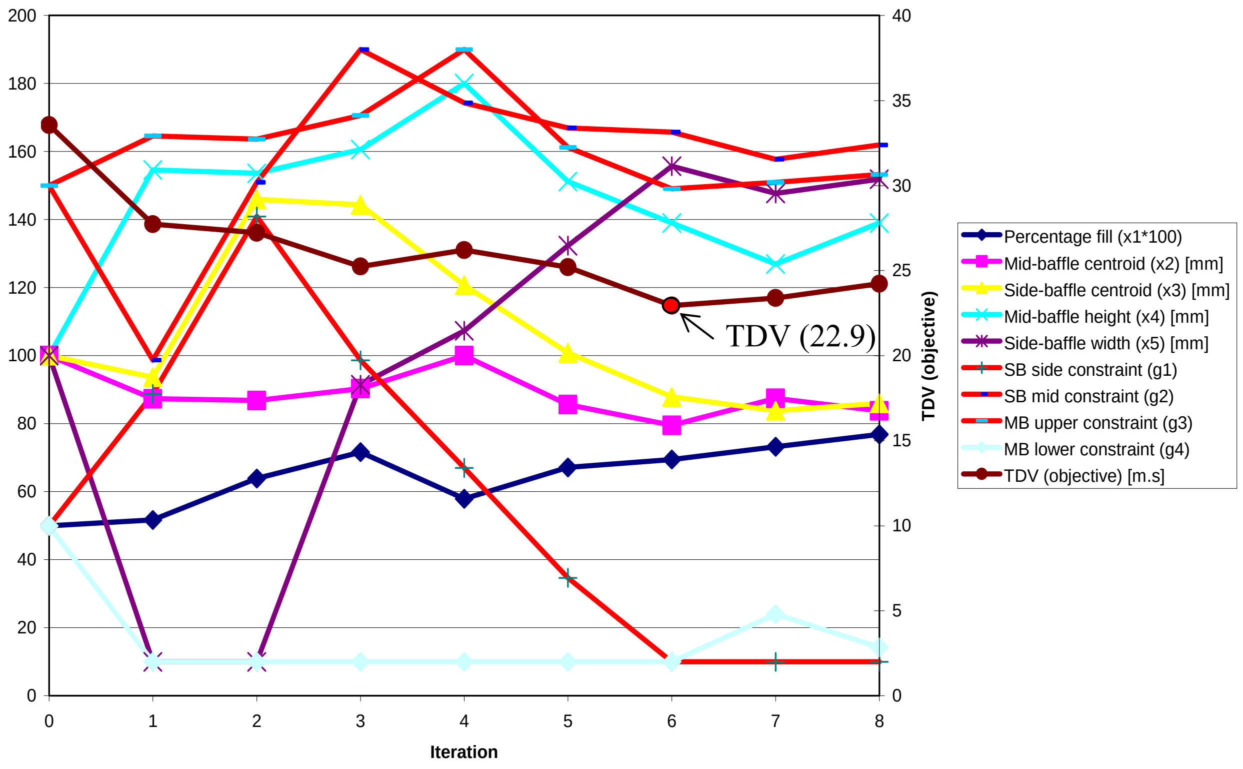


Figure 4.15: Optimisation history for Quadratic RSM saddle point Design 2

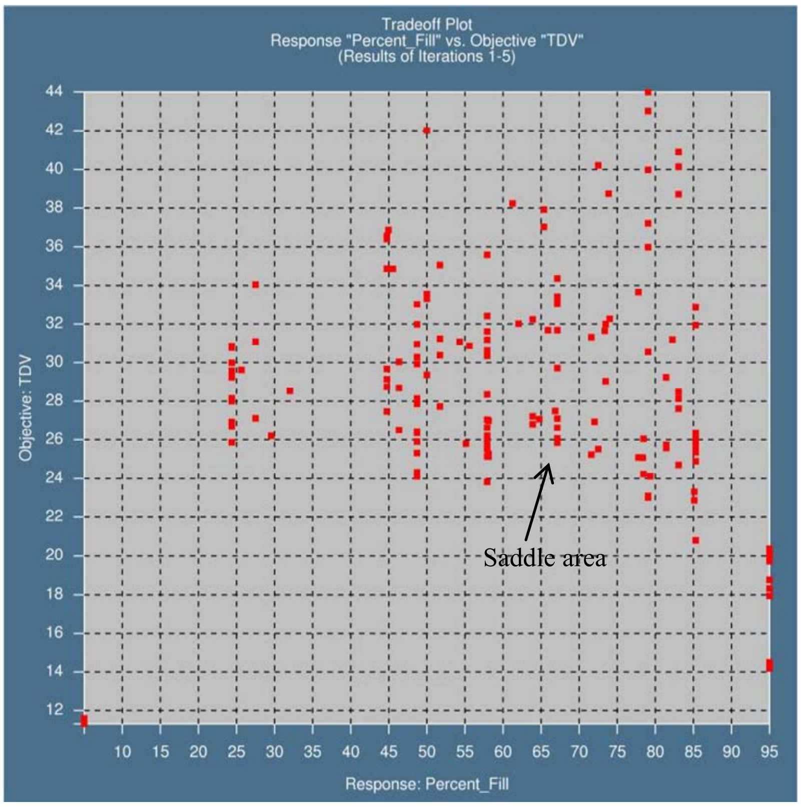


Figure 4.16: Trade-off plot for TDV versus fill level for Quadratic RSM saddle point Design 2

Figure 4.16 above shows all the analysis points plotted as TDV against Percentage fill. The trend clearly illustrates the reduction in sloshing due to extreme fill levels. The data conforms well to what is suggested in the literature, as discussed in section 2.4.2 of this dissertation. The spread of values for a given fill level represents the various designs considered for that fill level, and illustrates the variation in performance of those designs. Figure of between 69 % and 77% full for the worst case TDV in the last 3 optimisation iterations support the previous use of 70% full as a fill level. The implication is that other fill levels would exhibit less sloshing for the same set of design variable values (hence the notation of a saddle point).

4.4.2.8 Summary of 2D optimisation results

Table 4.11 below provides a summary of the 2D-sloshing optimisation results presented in this chapter.

Table 4.11: Summary of 2D-sloshing optimisation results

Case	TDV (start) $f*10^4$ [m.s]	TDV (final) $f*10^4$ [m.s]	TDV best data point $f*10^4$ [m.s] (Iteration)	Total baffle length [mm]	Design variables (Start)	Design variables (final)	Total iterations	Function Evaluations	Active constraints
1	36.46	22.66	22.66 (8.1)	373.8	(Hc,Hb,S) (100,100,50)	(Hc,Hb,S) (77.28,134.6, 10)	10	50	g_2,g_6
2	36.46	23.26	22.9 (8.3)	370.8	(Hc,Hb,S) (100,100,50)	(Hc,Hb,S) (76.68,133.6, 10)	8	128	g_2,g_6
3	36.46	26.41	22.9 (1.115)	349.5	(Hc,Hb,S) (100,100,50)	(Hc,Hb,S) (73.24,126.5, 10)	1	129	g_2,g_6
4	36.46	23.18	23.18 (10)	378	(Hc,Hb,S) (100,100,50)	(Hc,Hb,S) (78,136,10)	10	40	g_2,g_6
5	27.03	22.52	22.52 (7.1)	418.8	(MBC,SBC, MBH,SBW) (100,100, 100,100)	(MBC,SBC, MBH,SBW) (80.8,79.29, 141.6,138.6)	8	184	g_4
6	33.56	29.37	29.37 (11.1)	516	(MBC,SBC, MBH,SBW) (100,100, 100,100)	(MBC,SBC, MBH,SBW) (162,78, 244,136)	10	81	g_4
7	33.56	24.23	-	443	(F,MBC,SBC, MBH,SBW) (100,100, 100,100,0.1)	(F,MBC,SBC, MBH,SBW) (0.153,84,86, 139,152)	8	257	g_4

It is interesting to note that although most optimum values coincide with the best data point for the particular case, the two cases that do not (Case 2 and 3) use the same data set. It should be once again noted that the data points used for the nueral network are supplied by the database of points available from the quadratic SRSM (Case 2) analysis. The suspicion is that in both cases, insufficient resolution of points were available near the optimum and that further optimisation iterations would have in all likelihood resulted in more accurate predictions of the location of the local minimum.

The Linear RSM analyses provided very similar final results to the Quadratic RSM analyses with fewer function evaluations, e.g., 50 (Case 1) and 128 (Case 2) function evaluations for the Linear and Quadratic RSMs respectively for design 1. If one is interested in the optimum result only, one would therefore favour the Linear RSMs. If however one is interested in the trends exhibited for each design problem, one requires a higher level of confidence in the quality of the fit with respect the response in question. Figure 4.17 below provides some insightful data in this respect. The bar graphs provided represent ANalysis Of VAriance (ANOVA) [34] plots for both the Linear and Quadratic RSM analyses of the design 1 liquid container. Both plots are for the 3rd optimisation iteration of their respective analyses. Each bar represents a specific term in the equation which corresponds to the response surface fitted through the data for that optimisation iteration. Intuitively, the Linear fit requires only three (the number of variables) terms, as interaction effects are not considered, while the quadratic fit requires nine terms. The magnitudes of the bars relative to each other illustrate the relative significance of that term in the fit. The magnitude of the red portion of the bar provides a relative indication of the confidence in the value of the coefficient chosen for that term.

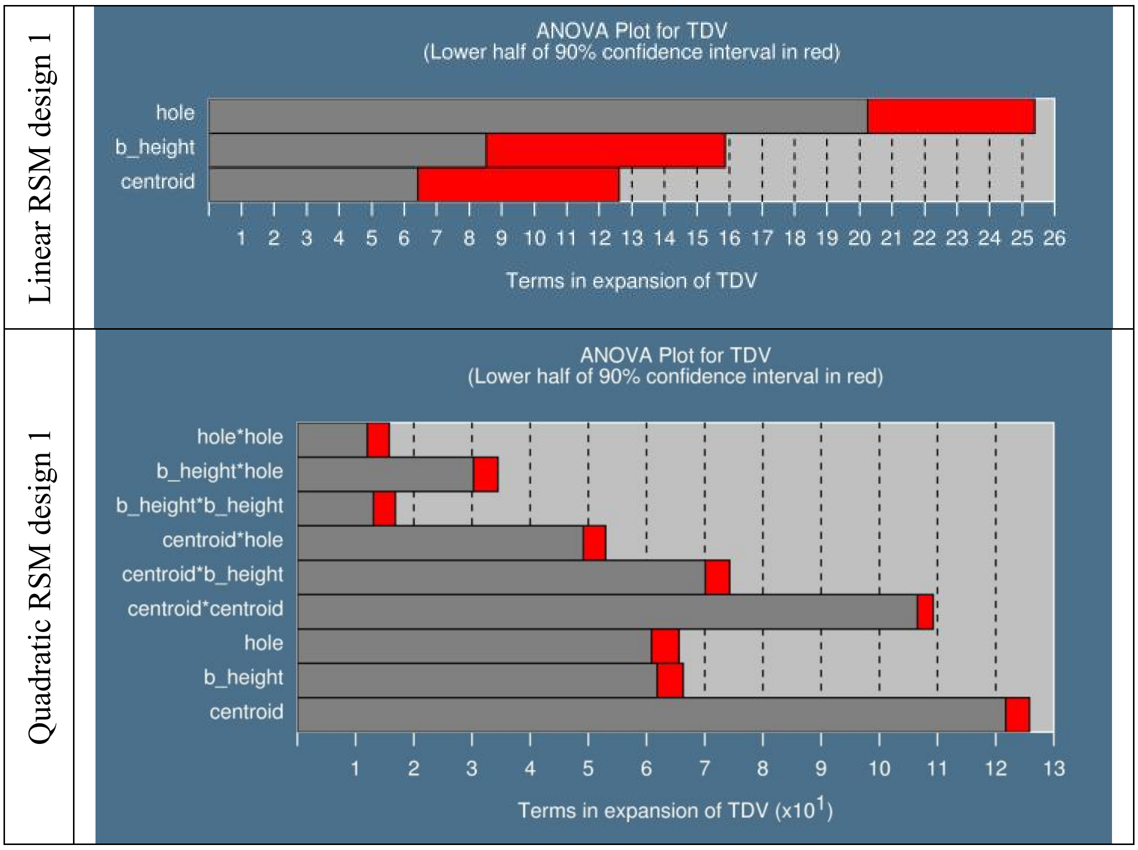


Figure 4.17: Comparative ANOVA plots for Quadratic and Linear RSM analyses

It is clear from the plots that the quadratic fit provides a surface in which the optimisation algorithm has significantly higher confidence. The plot also shows that the quadratic cross or interaction terms are as significant as the linear terms. These data provide strong motivation for the use of a Quadratic RSM if the trends in the performance of the design are of significant interest, e.g., when considering trade offs.

4.5 Conclusion

The main point that can be concluded form this chapter is that RSMs in conjunction with LS-OPT provide a robust and insightful method of numerical design optimisation of liquid containers for sloshing. The over-sampling method used by LS-OPT ensures minimal susceptibility to noise and a good estimation of the response for the specific design subspace.

A final observation from this chapter is that global meta-model methods like Kriging and Neural Networks can provide interesting plots when contemplating global trends for up to two variables.

CHAPTER 5: Optimisation for Impact

5.1 Introduction

This chapter covers both the modelling aspects of an impact analysis and the definition of the optimisation problem. The analysis for impact is done using the commercial non-linear finite element solver LS-DYNA 970 [26]. The chapter includes results of the single discipline optimisation for impact, covering impact analyses of both 2D and 3D geometries.

5.2 Impact Analysis

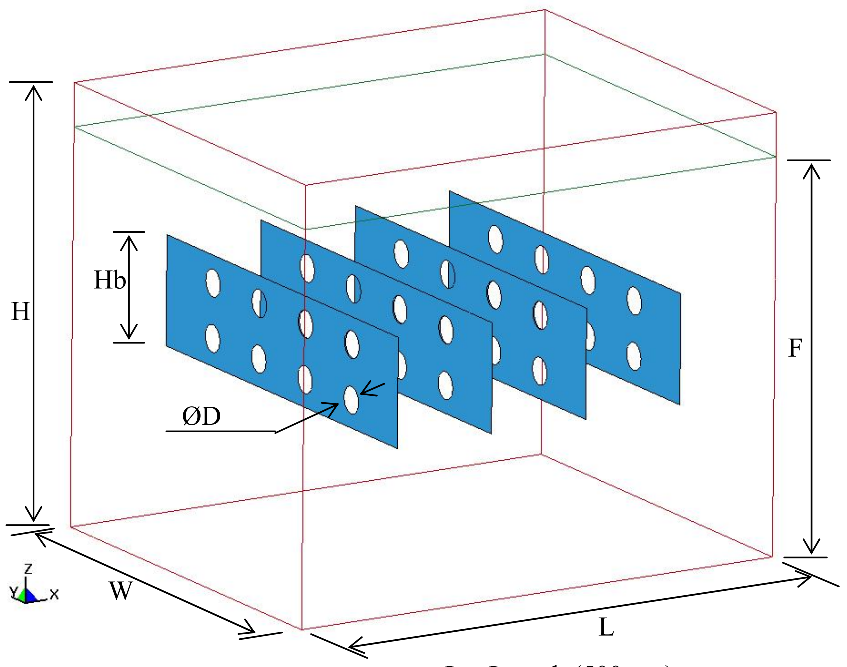
This section discusses the methodology adopted for the modelling of the fluid structure interaction in a liquid container. The analyses will simulate the stresses experienced in the baffles of a liquid container when exposed to accelerations typical of an impact. All procedures are fully automated for the purposes of numerical optimisation.

5.2.1 Mesh generation

Although all analyses are performed by LS-DYNA 970, LSTC (the supplier of LS-DYNA) does not currently provide an adequate parametric pre-processor. The available FEM pre-processor for this study is MSC-Patran [12], a dedicated pre-processor intended for use with MSC FEM solvers e.g., Nastran, Marc, Dytran. Patran does however provide an export function that is compatible with simple models for an older version of LS-DYNA, i.e., v930.

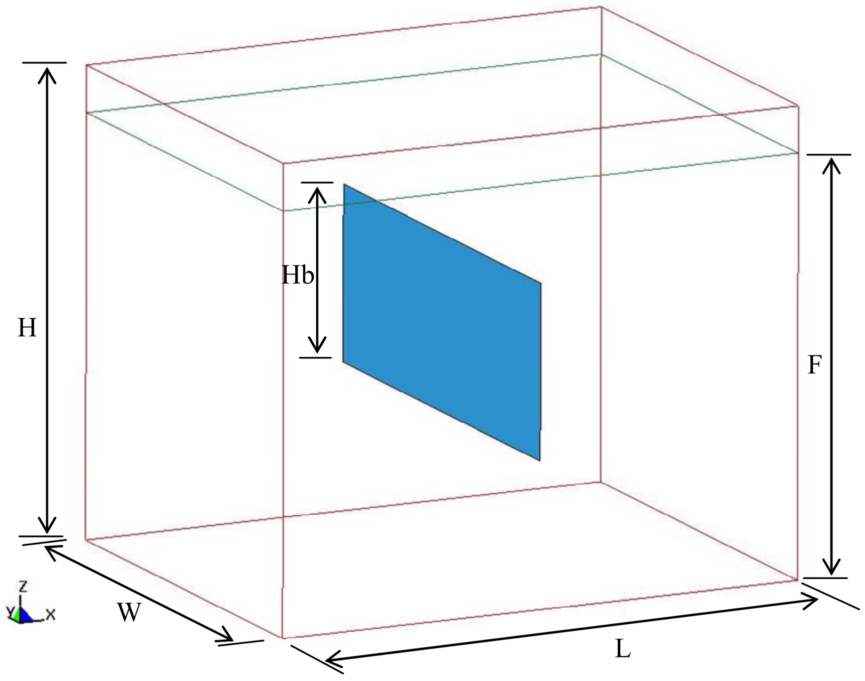
The geometries considered will all be solved using three-dimensional models since two-dimensional simplifications of the geometry are not possible with the available methods. Figure 5.1 and Figure 5.2 below show the forms of the 3D and 2D geometries. The mesh consists of two elements types, brick elements and shell elements. The brick elements are used for the fluid phases (liquid and gas) while the shell elements are used for the walls of the container and the internal baffles. Although the shell elements occupy zero thickness in the model, they will behave according to the user-specified shell thickness. A fill level of 90% is used for all impact analyses. A higher fill level is used because it represents more fluid inertia and in turn induces higher stresses than the lower 70% fill level used in the sloshing analyses.

Figure 5.3 below illustrates the form of the mesh used in the 3D impact analysis. The image is as seen from one end of the container, perpendicular to the baffle surface. The form of the mesh throughout the length of the model remains as seen in this figure. A typical mesh contains approximately 100 000 hexahedral cells.



L – Length (500mm)
W – Tank width (400mm)
H – Tank height (400mm)
Hb – Baffle height
ØD – Hole diameter
F – Fill level (90%)

Figure 5.1: Geometry of 3D liquid container



L – Length (500mm)
W – Tank width (400mm)
H – Tank height (400mm)
Hb – Baffle height
F – Fill level (90%)

Figure 5.2: Geometry of 2D "extruded" liquid container

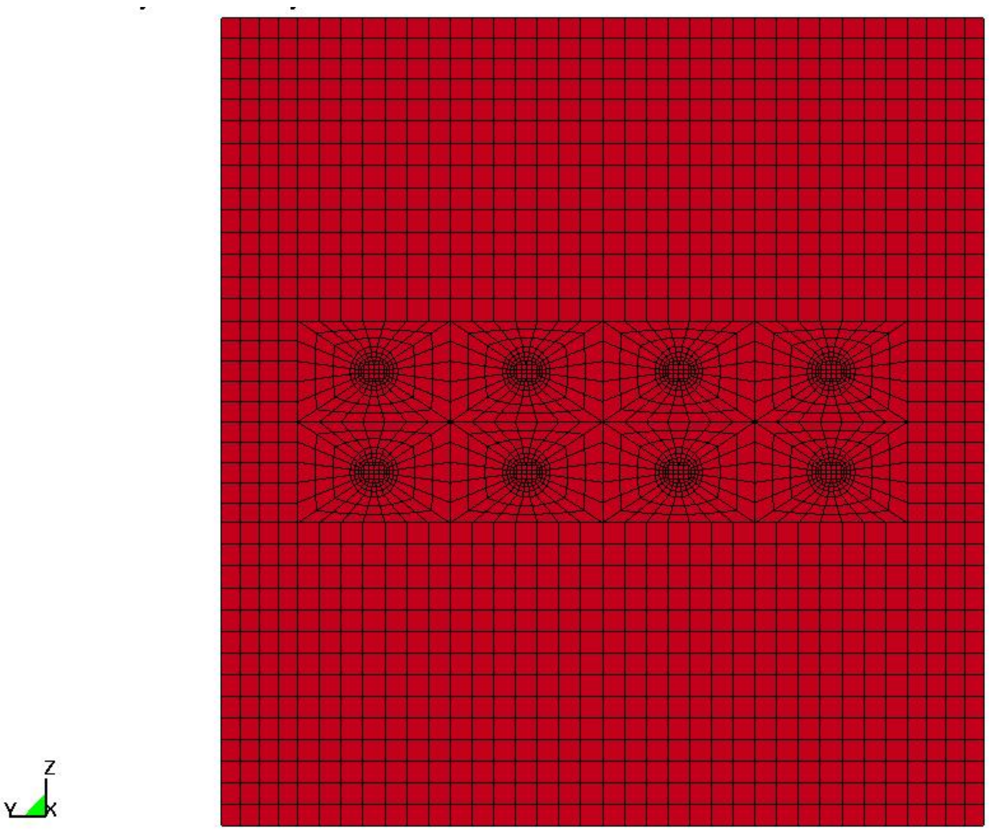


Figure 5.3: Mesh used in LS-DYNA analysis

Since the mesh generation forms part of a typical optimisation procedure, the process needs to be fully automated. As with Gambit before, a text or session file is created that contains a string of command-line commands in Patran Command Language (PCL) that represent Patran actions. By running this session file in batch mode, Patran will generate the mesh according to specified variable values and export the resulting mesh to LS-DYNA v930 format. An example of one such session file can be seen in Appendix S. This file is then cleaned using the scriptable Linux text editor SED that uses a script as seen in Appendix T to remove all but the mesh data.

5.2.2 Model setup

This section describes the setup and assumptions for the impact model. Since LS-DYNA has no graphical user interface in UNIX versions, all settings are loaded to the solver user a text file known as a keyword file. The keyword file contains all models to be included and their corresponding settings, node locations and connectivity data (generated as described above), and the load curves that will be applied during solution.

Since the section of the file that contains the mesh data is generated using Patran, the models and their settings will need to be manually prescribed in the text file. Appendix U provides an example of the section of the keyword that specifies models and model settings. Some of the more significant models used are described below.

The Arbitrary Lagrange-Eulerian (ALE) model is used for cases like the one analysed in this study where fluid motion is modelled in conjunction with structural deformation. Unlike with the Lagrangian formulation, the nodes do not follow the material flow, instead the material flows through a fixed mesh. The structural shell elements are however still treated with the Lagrangian formulation. The ALE model is used together with a fluid-structure coupling algorithm that prescribes the type of interaction that will take place between the materials in the Eulerian mesh and the elements of the Lagrangian structure. In the case of this study a Penalty coupling algorithm is used, as described in section 2.4.4 of this dissertation.

Other significant entries in the keyword file include those that describe the material properties of the various elements used in the analysis. These include the following materials.

The first material type is the rigid material that is used for the walls of the liquid container itself. No stress distribution is solved in these elements. The second material type is the plastic-kinematic material used for the shell elements of the baffles. All stresses and strains are solved in this material and it is allowed to deform plastically according to material type 3 in LS-DYNA [26] that has a bilinear stress-strain curve. Kinematic hardening is used with a tangent modulus of 100 and no strain effects. The third and fourth materials are of type material null, which implies that no stresses or strains are solved for in elements of the material, but the motion of the material is solved for. These materials will be applied to the air and water inside the container respectively. The difference between these two materials is in their respective densities and viscosities. Table 5.1 below provides the properties of the materials used.

Table 5.1: Table of material properties

Density (container) [kg/m^3]	7830
Density (baffles) [kg/m^3]	7830
Young's modulus (baffles) [GPa]	207
Poisson's ratio (baffles)	0.3
Density (air) [kg/m^3]	1.1845
Viscosity (air) [N.s/m^2]	1.84e-5
Density (water) [kg/m^3]	998
Viscosity (water) [N.s/m^2]	0.001

The final entry in the keyword file includes the load curves that are used during the analysis. The most obvious of these is the gravity vector, which is applied to all materials. The second load curve that is applied to the rigid body only, is that which prescribes the motion of the tank during the impact scenario.

For the purposes of this study a condensed version of the impact load curve analysed by Craig, et al. [37] is used in all the impact analyses. Figure 5.4 below shows the form of the load curve used for the analyses. The various stages of acceleration are due to various parts of a vehicle being crushed. The full load represents the acceleration seen by the vehicle during a typical NHTSA full frontal collision. The compressed signal is shortened by an order of magnitude while its amplitude is increased 4 times. Using this form of compression, analysis results using the compressed signal provide similar peak stresses in the baffles. The reason for using the compressed signal is due to the time required to run a full analysis. A full length load curve analysis will run for approximately 14 hours on a 3GHz P4 Linux workstation while the compressed-signal analysis will run for 2 hours on the same machine. When considering numerical optimisation on single-processor machines, within the context of this study, it is impractical to wait 14 hours for a single solution.

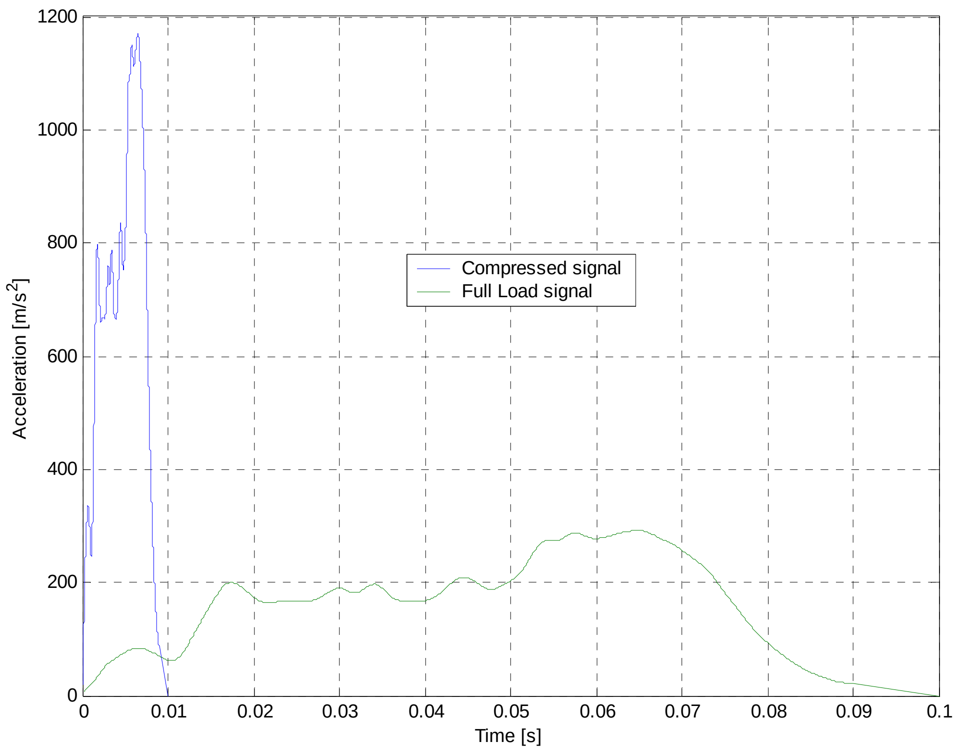


Figure 5.4: LS-DYNA load curves

5.3 Mathematical Optimisation

This section describes the methods used for the optimisation of the liquid container for impact. The section includes the definition of the optimisation problem, the setup of the automated optimisation process, and the results obtained.

5.3.1 Definition of Problem

As discussed in Chapter 2, the integrity of a liquid container is an important factor and can be an issue of law in the case of a vehicle's fuel tank. To demonstrate the integrity of the fuel tank as a constraint for numerical design optimisation, the maximum principal stress in the baffles of the liquid container will be monitored and considered violated if it exceeds a predefined maximum value. The objective of the problem will be to reduce the mass of baffles in the container without sacrificing their integrity as defined.

5.3.2 Problem Setup

Figure 5.5 shows a flow diagram for the cycle followed during the optimisation for impact problems. This method is applied in both analyses that are discussed later in this chapter.

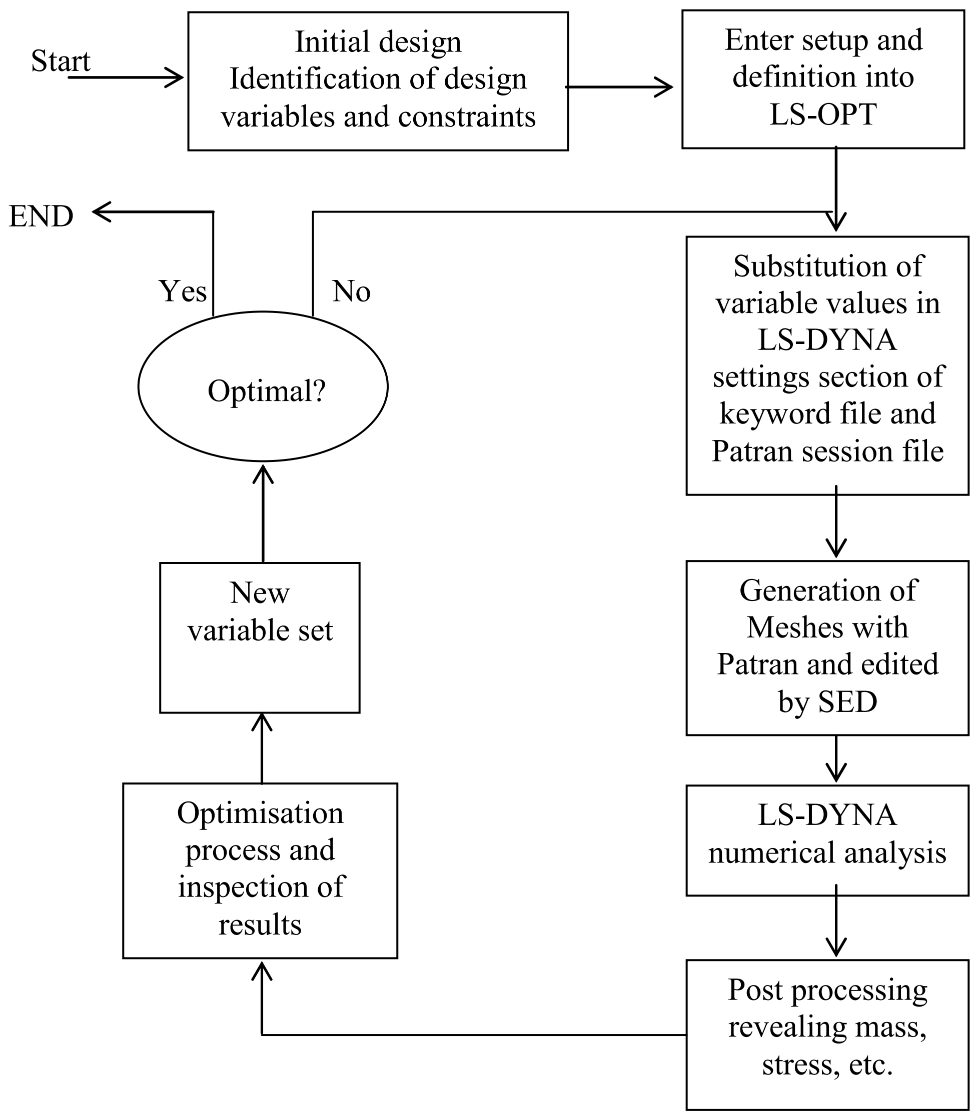

Start
Initial design Identification of design variables and constraints
Enter setup and definition into LS-OPT
END
Yes
No
Optimal?
Substitution of variable values in LS-DYNA settings section of keyword file and Patran session file
New variable set
Generation of Meshes with Patran and edited by SED
Optimisation process and inspection of results
LS-DYNA numerical analysis
Post processing revealing mass, stress, etc.

Figure 5.5: Flowchart for optimisation of impact problems

5.3.2.1 3D Geometry Optimisation Problem Definition

The 3D geometry optimisation was done using the Linear RSM method in LS-OPT (see Appendix V for LS-OPT command file). In accordance with Figure 5.1 above, the problem is defined as in Equation 5.1 below.

$$\textit{Variables } \mathbf{x} = [x_1, x_2, x_3]^T \tag{5.1}$$

x_1 = Hb = Baffle height

x_2 = ØD =Hole diameter

x_3 = Baffle thickness

Objective:

min $f(\mathbf{x})$ = Baffle mass = volume * density

Subject to:

Inequality constraints-

$g_1(\mathbf{x})$ = Max Principal Stress in Baffle < 200e6 Pa

$$g_2(\mathbf{x}) = \frac{x_1}{2} - x_2 > 0.03$$

Side constraints-

$g_3(\mathbf{x})$: x_1 (Hb) (80;300)mm

$g_4(\mathbf{x})$: x_2 (ØD) (15;50)mm

$g_5(\mathbf{x})$: x_3 (Thickness) (1;10)mm

Constraints g_2 to g_5 are all geometrical constraints and ensure the feasibility of the proposed geometry. Inequality constraint g_1 enforces the integrity of the design by ensuring that the maximum principal stress does not exceed the yield strength of the material used for the baffles (200 MPa in this case).

5.3.2.2 2D Geometry Optimisation Problem Definition

The 2D "extruded" geometry optimisation was done using the Linear RSM method in LS-OPT (see Appendix W for LS-OPT command file) and took approximately 1.5 weeks on a 3GHz P4 Linux workstation for the number of iterations shown below. The topology represents a reduced version of that seen in Chapter 4, 2D container design 2. Side baffles are neglected for the impact analysis as very little strain is placed on their integrity. In accordance with Figure 5.2 above, Equation 5.2 below provides the problem definition.

$$\textit{Variables } \mathbf{x} = [x_1, x_2, x_3]^T \tag{5.2}$$

x_1 = MBC = Middle baffle centroid

x_2 = MBH =Middle baffle height

x_3 = Baffle thickness

Objective:

min f(**x**) = Baffle mass = volume * density

Subject to:

Inequality constraints-

$g_1(\mathbf{x})$ = Max Effective V-M Stress in Baffle < 200e6 Pa

$$g_2(\mathbf{x}) = \frac{x_1}{2} + x_2 < 0.34$$

$$g_3(\mathbf{x}) = x_2 - \frac{x_1}{2} > 0.02$$

Side constraints-

$g_4(\mathbf{x})$: x_1 (Hb) (40;320)mm

$g_5(\mathbf{x})$: x_2 (ØD) (60;320)mm

$g_6(\mathbf{x})$: x_3 (Thickness) (1;10)mm

Constraints g_2 to g_6 are all geometrical constraints and ensure the feasibility of the proposed geometry. Again, inequality constraint g_1 enforces the integrity of the design, but instead of maximum principal stress, the maximum effective Von-Mises stress is used. The discussion of results below explains the reason for this choice.

5.3.3 Optimisation Results

The following section provides the results of the optimisation analyses described in section 5.3.2 above. Results will include the improvement of the design w.r.t. its starting value as well as the progression of all the variables and responses during the optimisation process.

5.3.3.1 LS-OPT 3D Impact Case Optimisation

As stated before, this study involves a Linear RSM approach within the LS-OPT framework. Table 5.2 below provides the final results for this case. Figure 5.6 below illustrates the progress of the Linear RSM optimisation run. The optimisation required seven function evaluations per optimisation iteration with a total of 43 evaluations which took approximately 2 weeks to complete on a 3GHz P4 Linux workstation for the number of iterations shown below.

Table 5.2: Final Results for 3D impact case

	Starting value	Converged linear RSM result
Hb (x_1) [mm]	100	90
ØD (x_2) [mm]	25	15
Thickness (x_3) [mm]	2	3.83
Mass*10 f(x)	2.26	**4.04**
Max Stress (g_1) [MPa]	623	**186**

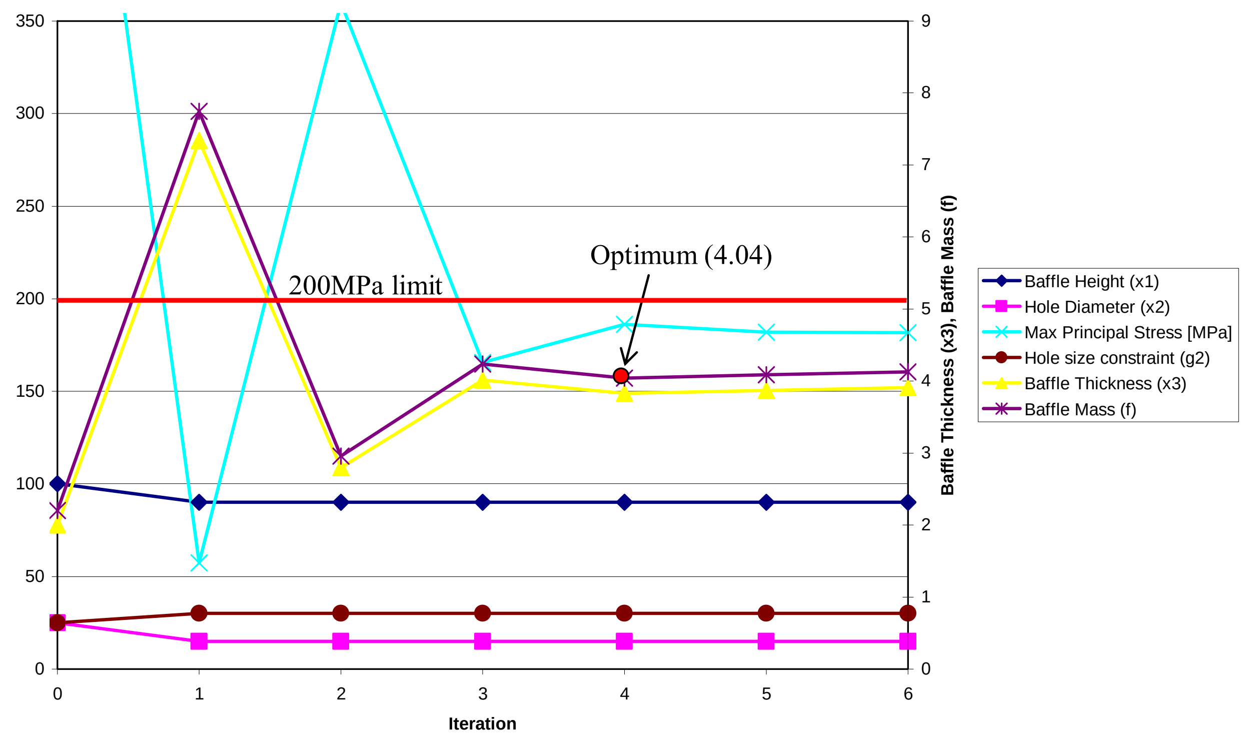


Figure 5.6: Optimisation history for 3D impact case

The first point of interest point is that the objective of mass has in fact increased with respect to the starting design. This is for no reason other than that the starting design violates the first inequality constraint of maximum principal stress by over 400 MPa. Although the first two variables of baffle height and hole diameter converge after the first iteration, the baffle thickness must establish sufficient magnitude to ensure structural integrity. The hole diameter and baffle height combine to achieve the smallest allowable frontal area for the baffle (as per constraint g_2 which restricts the hole from becoming too large relative to the baffle height). This intuitively reduces the level of inertial impact experienced by the baffle. This reduction is however insufficient and the baffle must be further strengthened by increasing its thickness from a starting value of 2mm to a final value of 3.83mm. Inequality constraint g_2 is active, but the limit of g_1 has not been attained. After further investigations it would seem that the principal stress method of establishing structural integrity does not perform as well as the maximum Von-Mises Stress method utilised in the 2D-extruded case of Section 5.3.3.2.

Figure 5.7 below provides an illustration of the 3D impact-case geometry with the variables corresponding to those for the optimum case. The image also provides contours of stress for a discrete moment, 0.008 seconds (Not necessarily peak stress), illustrating stress concentrations near the tank walls.

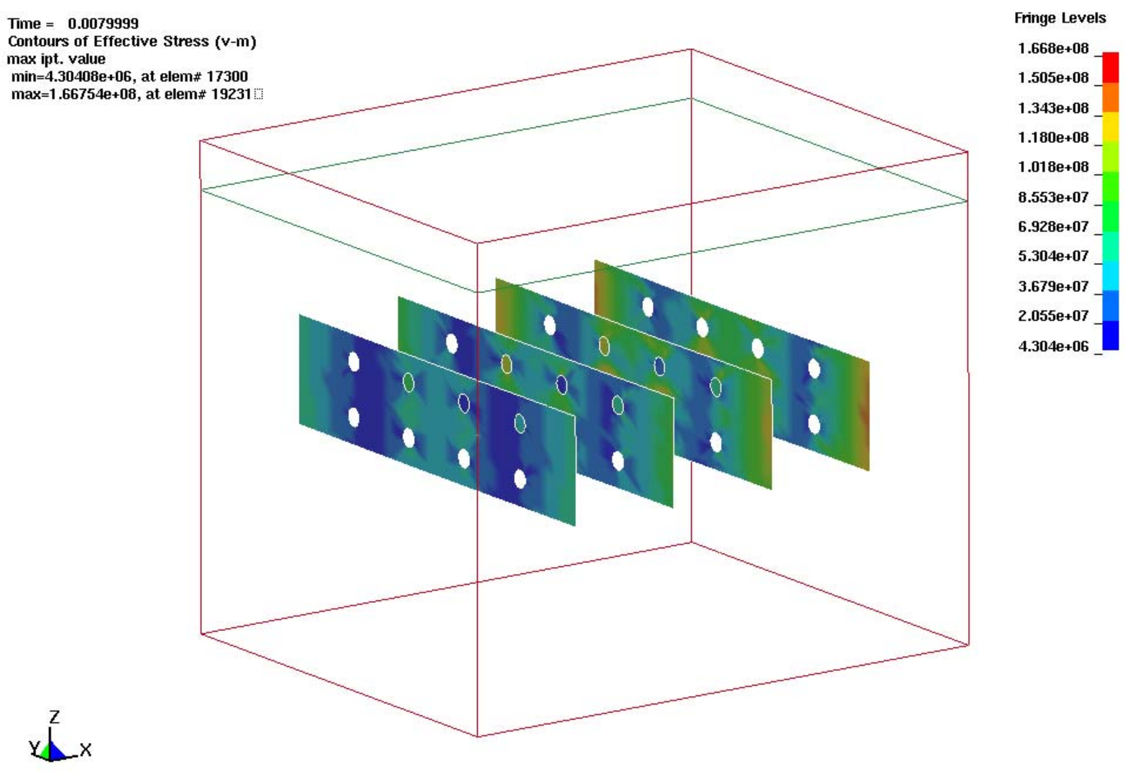


Figure 5.7: Optimum 3D impact-case geometry showing effective Von-Mises stress concentrations

5.3.3.2 Optimisation results for 2D "extruded" case

As stated before, this study involves a Linear RSM approach within the LS-OPT framework. Table 5.3 below provides the final results for this case. Figure 5.8 below illustrates the progress of the Linear RSM optimisation run. The optimisation required seven function evaluations per optimisation iteration with a total of 57 evaluations which took approximately 1.5 weeks to complete on a P4 3GHz Linux workstation.

Table 5.3: Final result for 2D "extruded" impact case

	Starting value	Converged linear RSM result
MBC (x_1) [mm]	100	113.6
MBH (x_2) [mm]	100	40
Thickness(x_3) [mm]	8	6.84
Mass*10 f(x)	30.1	**12.85**
Max Stress (g_1) [MPa]	153.8	198

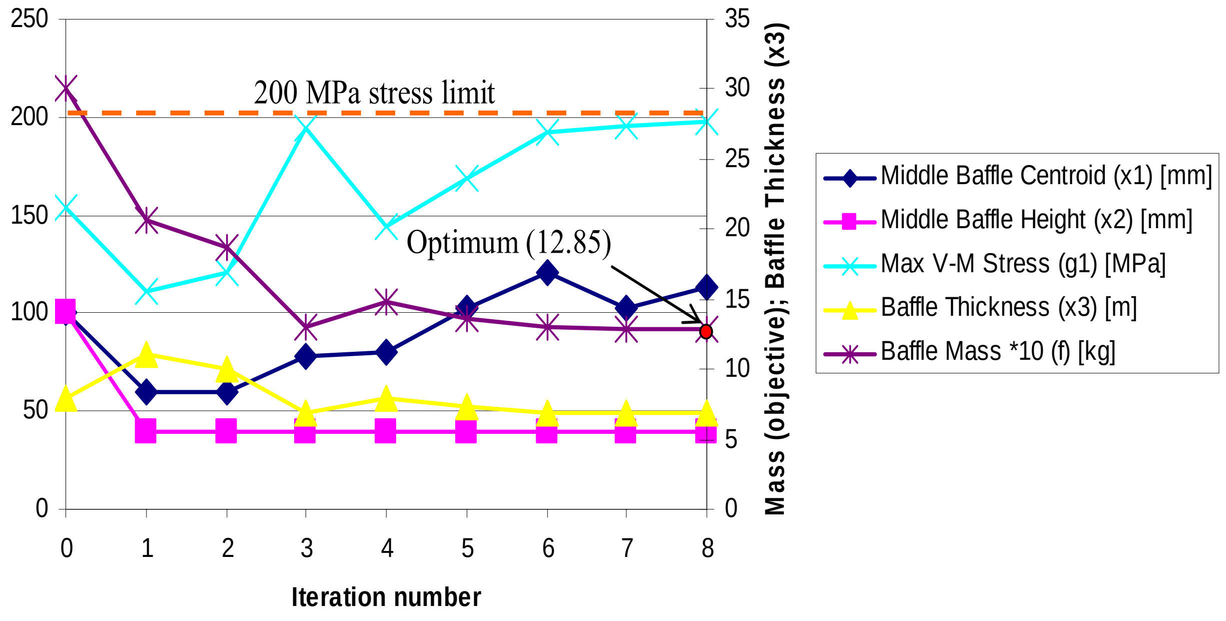


Figure 5.8: Optimisation history for 2D "extruded" impact case

This case provides a contrast to the previous case with an improvement in the objective function of 65%. This is because the maximum stress constraint is not violated but is in fact satisfied for the starting design. The baffle thickness is however significantly larger (6.84mm vs. 3.83mm). It is clear that the maximum Von-Mises effective stress is a stricter measure of stress than the maximum principal stress. It would furthermore seem that the maximum effective Von- Mises stress is a smoother and more reliable measure of the structural performance or integrity of the baffles.

During the optimisation process, the use of the Von- Mises stress demonstrated improved optimisation stability.

In terms of the impact event, the first two variables for baffle height and centroid location have moved to place the baffle as high as possible and make it as small as possible respectively. The explanation for this is firstly that again a smaller frontal area will absorb less of the inertia of the fluid and by being nearer the top of the container allows some of the deflected fluid to move more freely into the small space occupied by air near the top of the tank. The active constraints include the one that restricts the upward movement of the baffle (g_2) and the maximum stress in the tank (g_1).

Figure 5.9 below gives an illustration of the 2D “extruded” impact-case geometry with the variables corresponding to those for the optimum case. The image provides contours of stress for 0.01 seconds (Not necessarily peak stress), illustrating stress concentrations near the tank walls. Also note slightly higher concentrations near the lower half of the baffle indicating that the fluid near the free surface is free to move upward and therefore induce less impact energy onto the baffle in this region.

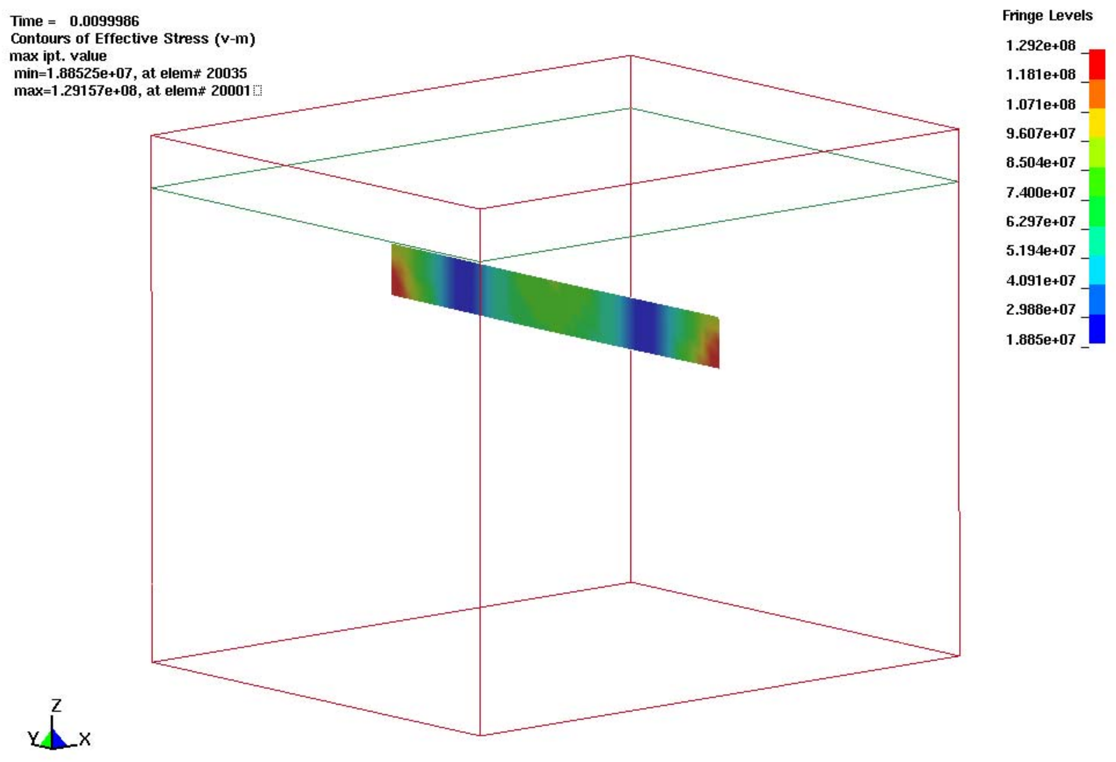


Figure 5.9: Optimum 2D "extruded" impact-case geometry showing stress concentrations

5.4 Conclusion

The results from this chapter illustrate the importance of the frontal area of the baffle with regards to the level of the impact energy it absorbs. Some insight into the impact event is provided however the section does show the need for multidisciplinary analysis. Since baffles are predominantly used for their ability to reduce sloshing, a large reduction in baffle size defeats the purpose of its inclusion in the impact design. The need for structural integrity does however not change and the combination of these challenges leads us to Chapter 6, multidisciplinary optimisation for both sloshing and impact.

CHAPTER 6: Multidisciplinary Design Optimisation for Sloshing and Impact

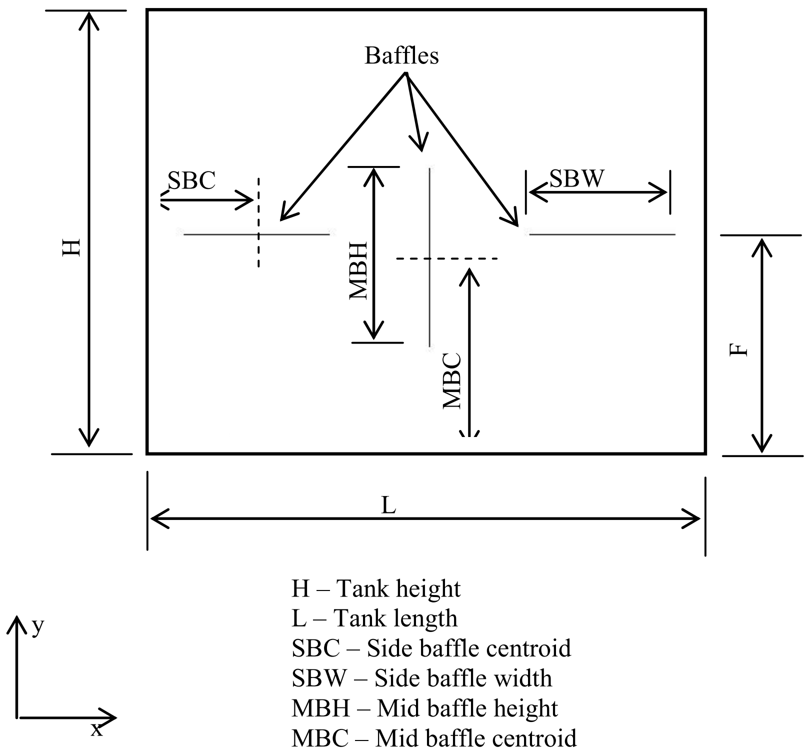
6.1 Introduction

This chapter covers the definition of a Multidisciplinary Design Optimisation (MDO) of liquid containers, considering both sloshing and impact criteria. The chapter covers the formulation and results of the MDO problem.

6.2 Definition of MDO Problem

The MDO problem of the liquid container is defined in such a way so as to consider the objectives discussed and utilised in the optimisation for sloshing section of the study (Chapter 4), as well as the objectives and criteria for integrity used during the optimisation for impact (Chapter 5). The aim is to attain a set of design variables that will provide both good sloshing performance and minimal use of material. The design must naturally also maintain structural integrity during an impact event.

The geometry considered is the same as those considered in sections 4.3.2.6 and 5.3.2.2 and is as shown in Figure 6.1 below.



H – Tank height
L – Tank length
SBC – Side baffle centroid
SBW – Side baffle width
MBH – Mid baffle height
MBC – Mid baffle centroid
F – Fill level

Figure 6.1: Geometry of container for MDO analysis

During the optimisation for sloshing analyses in Chapter 4, this geometry was referred to as design 2b, while in Chapter 5's optimisation for impact the identical geometry is entitled the 2D "extruded" geometry.

6.2.1 Problem Setup

All automated procedures are identical to those used during their respective analyses in Chapters 4 and 5. Both disciplines are now considered simultaneously and therefore require the separation of the variables into those that are shared and those that are considered by only one of the disciplines. The overall setup is best understood when examining it from a flowchart perspective as in Figure 6.2 below. Separate response surfaces are constructed for each discipline and combined in the manner prescribed by the weighting and scaling values (Equation 6.1). By separating the variables, the number of function evaluations required for each response surface is reduced [54]. Criteria for separation include variables that have no impact on the particular numerical solution as well as variables that have a minor or insignificant impact on the numerical solution. This may be done in a number of ways. The first and simplest method is to use one's understanding of the specific phenomena in question and eliminating the variables that are known to have little significance. This can be backed up with a preceding sensitivity study and will provide a more economical, yet meaningful starting point for the optimisation procedure. The second method would be to utilise the information available from the response surface creation that is already part of the mathematical optimisation process. The ANOVA information described in Chapter 4 provides a quantitative measure of the significance of the various variables and can be used to discard less influential variables [54]. This second method will provide a way of reducing the computational expense of each optimisation iteration by discarding less significant variables. Only the first method is employed in this chapter.

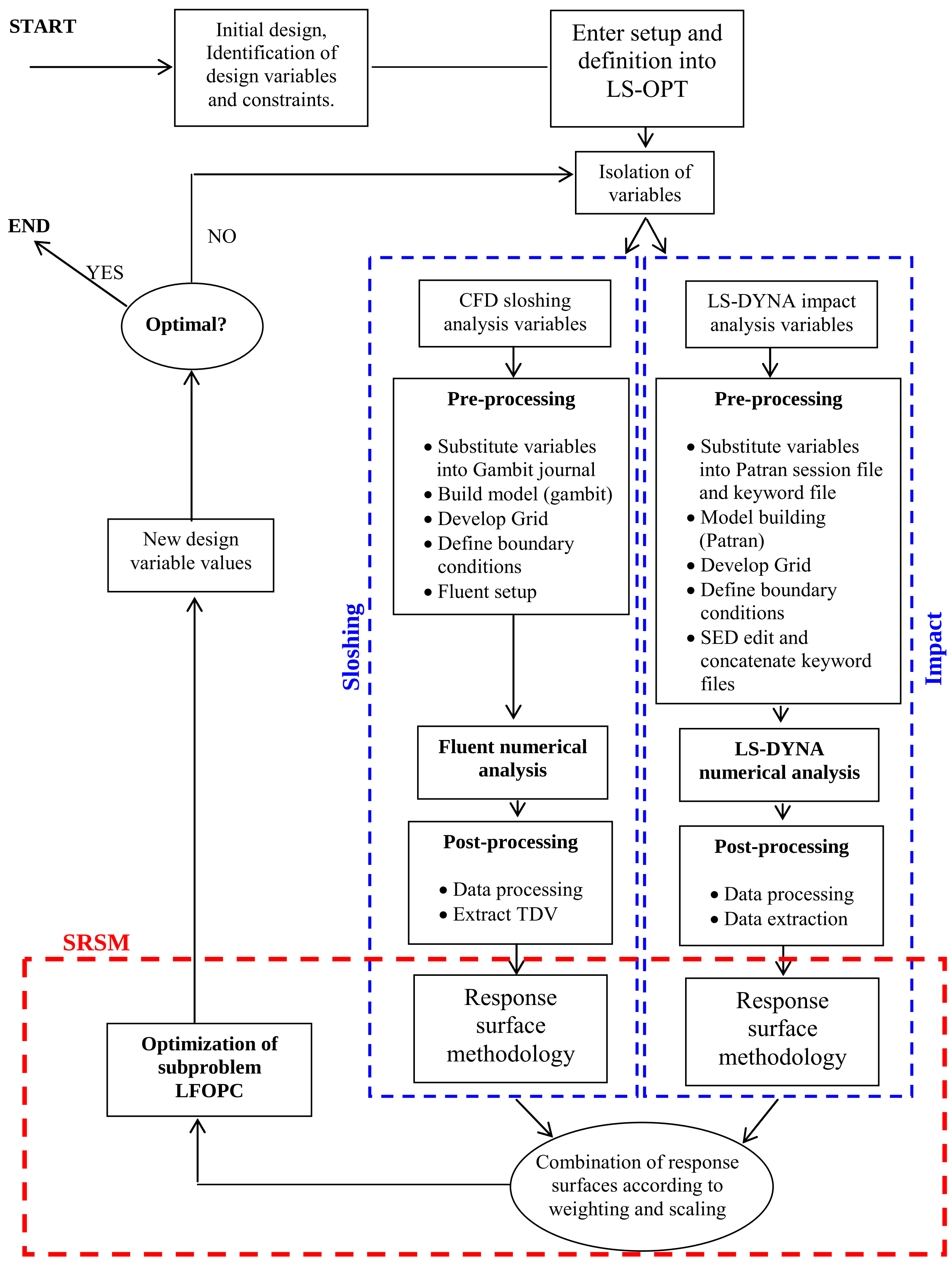

START
Initial design, Identification of design variables and constraints.
Enter setup and definition into LS-OPT
Isolation of variables
END
YES
NO
Optimal?
New design variable values
Sloshing
CFD sloshing analysis variables
Pre-processing
• Substitute variables into Gambit journal
• Build model (gambit)
• Develop Grid
• Define boundary conditions
• Fluent setup
Fluent numerical analysis
Post-processing
• Data processing
• Extract TDV
Response surface methodology
Impact
LS-DYNA impact analysis variables
Pre-processing
• Substitute variables into Patran session file and keyword file
• Model building (Patran)
• Develop Grid
• Define boundary conditions
• SED edit and concatenate keyword files
LS-DYNA numerical analysis
Post-processing
• Data processing
• Data extraction
Response surface methodology
SRSM
Optimization of subproblem LFOPC
Combination of response surfaces according to weighting and scaling

Figure 6.2: Flow diagram for MDO problem for sloshing and impact

6.2.2 Optimisation Problem Definition

The MDO was done using the Linear RSM method in LS-OPT (Command file included in Appendix X). Equation 6.1 below describes the optimisation problem definition, in accordance with the variables seen in Figure 6.1 above.

Multi objective: (6.1)

$$\min\ f(\mathbf{x}) = \sum_j \frac{f_j(x_j)\omega_j}{\alpha_j}$$

where:

$f_1(\mathbf{u})$ = Baffle mass = (Baffle volume) * density

$f_2(\mathbf{v})$ = TDV

weights :

$\omega_1 = 0.5$

$\omega_2 = 0.5$

scales :

$\alpha_1 = 0.1$

$\alpha_2 = 0.001$

Variables:

$\mathbf{x} = (\mathbf{u}, \mathbf{v}) = [x_1, x_2, x_3, x_4, x_5]^T$

x_1 = MBH = Middle baffle height $\{\in \mathbf{u}\ \&\ \mathbf{v}\}$

x_2 = MBC =Middle baffle centroid $\{\in \mathbf{u}\ \&\ \mathbf{v}\}$

x_3 = SBC = Side baffle centroid $\{\in \mathbf{v}\}$

x_4 = SBW = Side baffle width $\{\in \mathbf{v}\}$

x_5 = Baffle thickness $\{\in \mathbf{u}\}$

Subject to:

Inequality constraints-

$g_1(\mathbf{u})$ = Max V-M Stress in Baffle < 200e6 Pa

$g_2(\mathbf{v}) = - x_3 + 0.5*x_4 + 10\text{mm} < 0$

$g_3(\mathbf{v}) = x_3 + 0.5*x_4 - 190\text{mm} < 0$

$g_4(\mathbf{x}) = x_2 + 0.5*x_1 - 340\text{mm} < 0$

$g_5(\mathbf{x}) = - x_2 + 0.5*x_1 + 40\text{mm} < 0$

Side constraints-

$g_5(\mathbf{x})$: x_1 (MBH) (40;320)mm

$g_6(\mathbf{x})$: x_2 (MBC) (60;320)mm

$g_7(\mathbf{x})$: x_3 (SBC) (15;185)mm

$g_8(\mathbf{x})$: x_4 (SBW) (10;180)mm

$g_9(\mathbf{x})$: x_5 (Thickness) (1;15)mm

Constraints g_2 to g_5 are all geometrical constraints and ensure the feasibility of the proposed geometry. Inequality constraint g_1 enforces the integrity of the design by ensuring that the maximum Von-Mises stress does not exceed the yield strength of the material used for the baffles (200 MPa in this case).

As previously mentioned, the variables are separated into those relevant to each numerical analysis discipline. The thickness of the baffles is of no relevance to the CFD analysis since zero thickness is assumed for the baffles. This is justified by the fact that the relatively thin baffles will have little influence on the flow patterns associated with sloshing. The variables that define the side baffles are not used in the impact LS-DYNA analysis. The exclusion of the side baffle variables from the impact analysis is justified by the fact that peak stresses are seen in the middle baffle only, since the side baffles absorb very little of the fluid's inertia during impact. Considering that there is insufficient time during the impact analysis to develop any flow patterns, it is intuitive that the exclusion of the side baffles will not have much influence on the analysis results. By separating the variables, the total number of function evaluations per optimisation iteration is reduced from 20, i.e.,(10+10) for a fully shared scenario to 15, i.e.,(7+8) for the setup considered here. This will equate to approximately 25% improvement in solution time per optimisation iteration that would have clear advantages in a design environment.

6.3 Optimisation Results

The multidisciplinary optimisation results in this section represent 13 optimisation iterations and a total of 5 partially shared variables. 3 variables are used in the impact analysis and 4 variables in the sloshing analysis. A total of 92 impact analysis function evaluations and 105 sloshing analysis function evaluations took approximately 3 weeks to solve on a 3GHz P4 Linux workstation. Figure 6.3 below illustrates the optimisation history of the objectives, variables and responses with respect to optimisation iteration number. The optimisation process exhibits reasonable convergence, and the final design represents an improved solution over the base design. The Von-Mises stress constraint is periodically active from the 9th optimisation iteration.

Table 6.1 below provides the starting an ending values for the variables, objectives and stress constraint.

Table 6.1: MDO analysis results

	Starting value	Final linear RSM result
MBH (x_1) [mm]	100	40
MBC (x_2) [mm]	100	263
SBC (x_3) [mm]	100	98.4
SBW(x_4) [mm]	100	10
Thickness(x_5) [mm]	8	7.46
Mass*10	30.1	**13.97**

$f_1(x)$		
TDV **$f_2(x)*10^4$** **[m.s]**	49.17	**43.97**
Max Stress **(g_1) [MPa]**	153.8	198

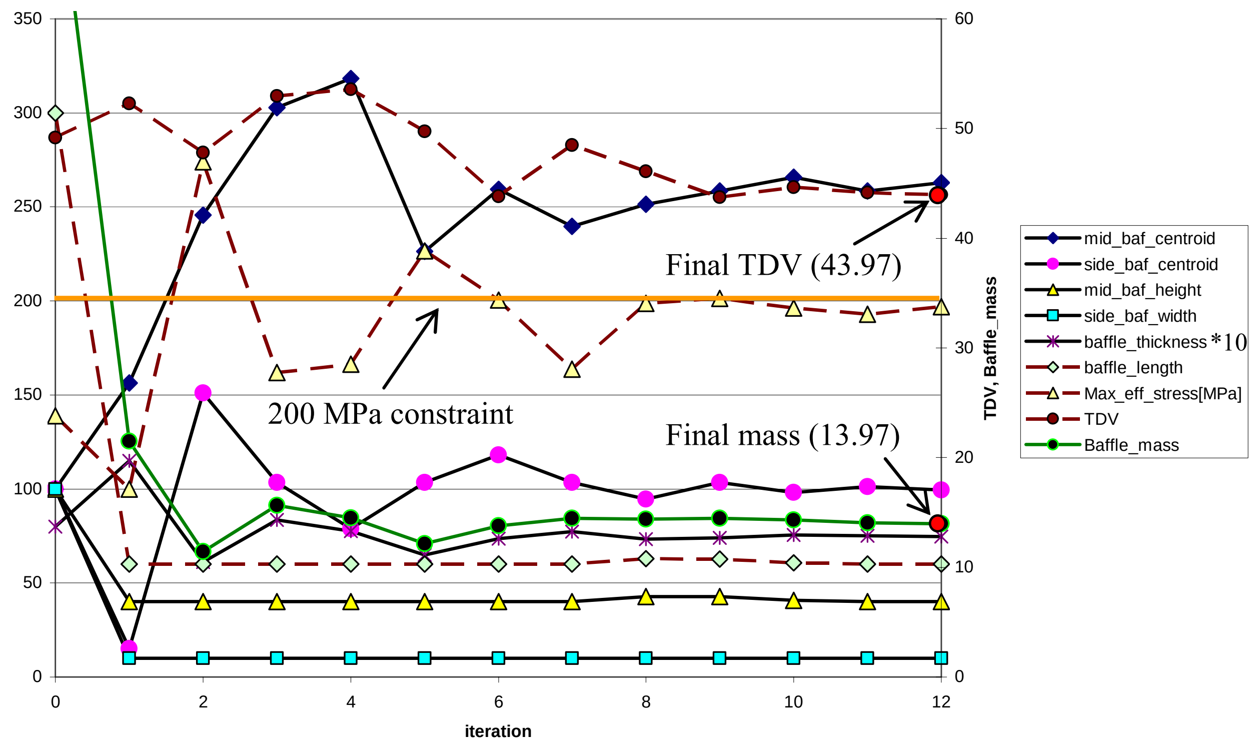


Figure 6.3: Multidisciplinary optimisation history

The results indicate an optimisation routine that has favoured the mass objective. As one would imagine, the baffle thickness is closely linked to the effective stress performance. The linear RSM used results in some oscillation of the design, but a converged solution is still obtained due to the sub-domain reduction scheme employed. The 50% weighting of the two objectives has still resulted in a greater improvement in the mass due to the relative difference between the starting design and a design that satisfies the stress constraint. In contrast, less design improvement is available for sloshing objective. This suggests that the MDO result is subject to the initially chosen weighting between the objectives. This can be further extended to a design environment, where the setup may reflect a desired outcome. The mass of the container may come at a higher premium than the reduction of sloshing. More importantly, the methodology presented in this study allows for a simple method by which the engineer can manipulate the results to obtain a desirable result.

Figure 6.4 below illustrates the form of the final MDO design. Small side baffles suggest that the reduction in sloshing they provide is less than the reduction in mass that can be achieved by making them small. The position of the centre baffle is by no means coincidental. The centre baffle aligns itself with the fluid level (70% for sloshing) as this is the area where the horizontal velocities in the fluid are the highest. Furthermore, since the fluid level at the centre of the liquid container remains relatively constant, the centre baffle is active for most of the sloshing event.

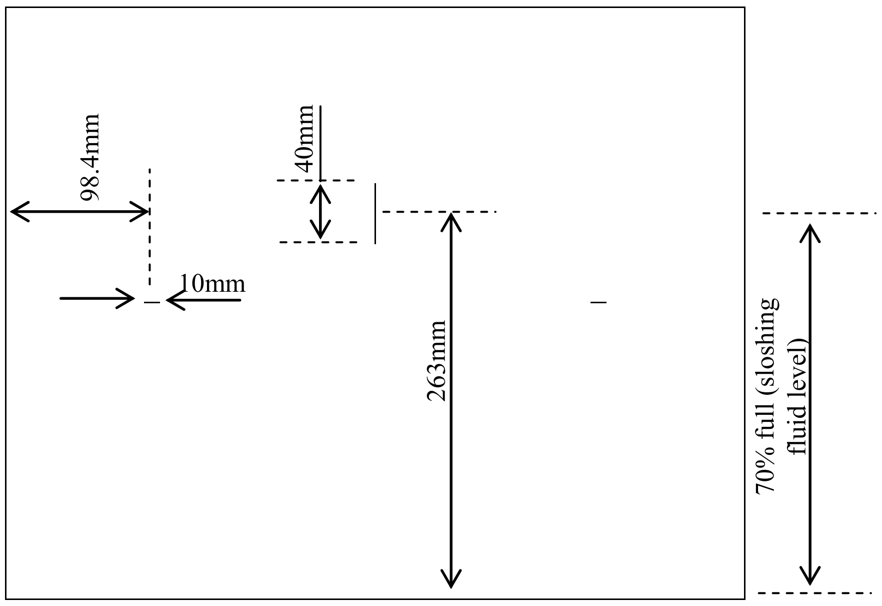


Figure 6.4: Final dimensions of MDO optimum design

6.4 Summary of Results

This section provides an overview of the results achieved for the various single and multidisciplinary optimisation studies on the same geometry. The three cases represent a sloshing only analysis, a mass and stress only analysis and the multidisciplinary mass and sloshing analysis respectively. Table 6.2 below provides the results for the three cases. The two single discipline studies show what improvements can be achieved for baffle and/or TDV. Examining these results in conjunction with the case 3 (MDO) results, it is noted that a compromise between sloshing and baffle mass has been achieved. This indicates some success with regards to the setup of the multidisciplinary design optimisation routine.

Table 6.2: Summary of the optimisation results for the three optimisation scenarios

Case	Objective *f*	**Objective (start)**	**Objective (final)**	Design variables (start)	Design variables (final)	Constraints active
1 (case 6 in Chapter 4)	TDV	TDV = 49.17 [m.s]	TDV = **29.37** [m.s]	$\boldsymbol{x}_{\text{slosh}}$= (MBC,SBC,MBH, SBW) = (100,100,100,100)	$\boldsymbol{x}_{\text{slosh}}$= (MBC,SBC,MBH, SBW) = (162,78,244,136)	SB side, MB lower
2 (2D extruded case in Chapter 4)	Baffle mass	Mass = 30.1[‡] [kg]	Mass = **12.85** [kg]	$\boldsymbol{x}_{\text{impact}}$= (MBC,MBH,Thickness)= (100,100,8)	$\boldsymbol{x}_{\text{impact}}$= (114,40,6.84)	Maximum effective von Mises baffle stress = 198 MPa
3	0.5TDV + 0.5Baffle mass	TDV = 49.17 [m.s]; Mass = 74.88 [kg]	TDV = **43.97** [m.s]; Mass = **13.97** [kg]	$\boldsymbol{x}_{\text{slosh+impact}}$= (MBC,SBC,MBH, SBW,Thickness)= (100,100,100,100,8)	$\boldsymbol{x}_{\text{slosh+impact}}$ (263,98.4,40, 10,7.46)	Maximum effective von Mises baffle stress = 198MPa

The behaviour of MBC is sited as quite interesting in that at first glance this would appear counter intuitive, however, these values should be read in conjunction with the value for MBH (middle baffle height). For the sloshing only optimisation (Case 1), the middle baffle is very large and only the upper edge of the baffle is interacting with the free surface. This gives optimal sloshing performance but uses a lot of material. For the MDO case (Case 3) the upper surface is still interacting with the free surface but the baffle is very small (to reduce mass). The result is

[‡] The starting mass for the impact only case is lower due to an assumed constant side-baffle width of 10mm since it is excluded from the analysis.

that the centroid of the baffle is high. Since sloshing is not a part of the impact analysis (Case 2), I do not believe the stresses to be very sensitive to its location. Figure 5.8 does however suggest that if further iterations were performed the location of the centroid would be higher. Therefore for the MDO case the lack of sensitivity of the stresses to MBC suggest that its location was determined by the sloshing discipline while the baffle's reduced size gave a lower mass.

6.5 Conclusion

This chapter illustrates the setup and results of the combining of the single disciplinary design optimisation techniques used in previous chapters. When considering sloshing only, the optimization algorithm employed effectively reduced the total deviation value used as an objective. When impact only was considered, the baffle mass was effectively reduced until the specified stress constraint was reached. The Multi-disciplinary Design Optimisation results, considering both sloshing and impact, show that a compromise could be found between sloshing behaviour and effective stresses in the baffles due to impact. Other formulations of the MDO problem may be considered, and the results will be highly dependent there on. The specific formulation will depend on the design engineer and results of interest.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This dissertation documents the work covered during the study of numerical design methods in the liquid container design environment. The work covers as many aspects as possible, from analysis and experimental validation to fully automated multidisciplinary design optimisation. The conclusions made during this study are as follows.

Chapter 2 indicated the large array of tools available to an engineer in the liquid container design cycle. An overview of the work done to this point suggests that multidisciplinary combinations of these tools to consider both sloshing and impact would be relatively new work. The limits and boundaries selected for use with these design tools are in some cases governed by legal requirements, as with vehicle fuel tank design.

The chapter that follows covered the modelling of the sloshing phenomena inside a liquid container using Computational Fluid Dynamics. This section provided sufficient insight into the validity of the numerical models as well as some level of insight into the phenomena typical in liquid container sloshing. Discrepancies between the measured and simulated sloshing results were obtained and explained as more attributable to experimental inadequacies (filtration of acceleration content and low-frequency capability of the accelerometer) rather than to simulation (modelling) error. Significantly, within a design perspective, an improved numerical CFD model would seemingly translate to an improved physical design for sloshing.

Chapter 4 documented an extensive look at optimisation for sloshing techniques. The results of the optimisation runs indicate that response surface methods in conjunction with LS-OPT provide a robust and insightful method of numerical design optimisation of liquid containers for sloshing. The chapter illustrated some of the statistical tools that are available for analysing the data available from an optimisation process.

Chapter 5 demonstrated the tools available in a finite element analysis environment to model a liquid container during an impact event and to establish container integrity. From an optimisation perspective, the results of an optimisation process that considers impact only give an as small as possible baffle. This however defeats the purpose of the baffle as a sloshing inhibitor and vindicates the need for a multidisciplinary optimisation process that considers both sloshing and impact.

As a combination of the best practice and most computationally economical design methods seen throughout the study, Chapter 6 presented the multidisciplinary design optimisation process that considers both sloshing and impact. The results of the process are encouraging, as a trade off is found between the requirements for an optimal design for both disciplines. The results suggest that the setup could be easily adapted to accommodate further geometries and circumstances. One of the observations made is that the result is largely influenced by the relative weighting or importance assigned to the respective disciplines. In addition, this implies that since a trade off exists, the desired result can be achieved by altering this discipline weighting.

The results achieved in this study pave the way for further studies in the field. The study may indeed be extended to three dimensions, both within the CFD model and the load curve. An example would be a liquid transporting vehicle performing a sudden turning manoeuvre. Full multi-body system dynamics may be included as a 3rd discipline (e.g., using ADAMS). Practical application of flow damping devices will in general need to accommodate full 3D free-surface motion. Another potential point of interest is the design and evaluation of moving baffles or other dynamic damping devices and the capabilities of CFD codes to handle multi-degree of freedom rigid bodies within the flow domain.

If computational resources exist, a full trade off curve would provide interesting inside into the influence of the weighting of the two disciplines in a multidisciplinary design optimisation process.

As this dissertation went to press, Fluent released an LES model that can work in conjunction with VOF. This will allow for the solution of acoustics and the quantification of sound pressure levels. This can be used at both the experimental verification level, comparing measured sound levels, and as a possible alternative objective.

The structural integrity analyses can be extended to the complete impact simulation of the liquid container, as prescribed by the safety standards. Recent advancements in the simulation of fluid-structure interaction allow for the simultaneous solution of a CFD and FEM solver while coupling the two methods with the transfer of pressure and deformations data (e.g., MPCCI). The merits of utilising the strengths of the two methods (CFD and FEM) in one coupled solution should be evaluated for improved accuracy.

Finally, this study successfully demonstrates the use of multidisciplinary analysis of liquid containers, but the processes illustrated could be applicable to any number of flow problems that invariably have structural design challenges included. Incorporating numerical optimisation with this multidisciplinary approach brings an added level of design cycle and time scale economy that is undoubtedly of benefit to any industrial design process.

REFERENCES

[1] Graham E W, Rodriguez A M, 1952, The characteristics of fuel motion which affect airplane dynamics. *Journal of Applied Mechanics.* 19 (1952) 381-8.

[2] Graham E W, 1951. The forces produced by fuel oscillation in a rectangular tank. *Douglas Aircraft Company Report*. SM-13748.

[3] Abramson H N, 1966, The dynamic behavior of liquids in moving containers, *NASA* SP-106.

[4] Fernando Meseguer-Garrido, 2003, On the sloshing of liquids in parallelepiped-shaped containers. *European Journal of Physics*. 24 (2003) 277-288.

[5] Vincenzo Armenio, Michele La Rocca, 1996, On the analysis of Sloshing of Water in rectangular containers: Numerical study and experimental validation. *Ocean Engineering.* 23 (1996) 705-739.

[6] Alain Cariou, Guido Casella, 1999, Liquid sloshing in ship tanks: A comparative study of numerical simulation. *Marine Structures*. 12 (1999) 183-198.

[7] Bass R L, Bowles E B, Cox P A, 1980, Liquid dynamic loads in LNG cargo tanks. *SNAME Trans*. 88 (1980) 103-126.

[8] Yamamoto S, Kataoka F, Shioda S, Ashitani Y, 1995, Study of impact due to sloshing in midsized LNG carrier. *International Journal Offshore Polar Engineering.* 5 (1995) 10-16.

[9] University of Washington, School of Oceanography. *Linear wave theory lecture notes.* Seattle, USA.

[10] Van Dyke, 1982. The album of fluid motion. *Stanford*, California, USA.

[11] www.nhtsa.dot.gov, web page for National Highway Traffic Safety Administration. *US department of Transport*.

[12] www.mscsoftware.com, web page for *MSC-Software*, CA, USA.

[13] Zhang J, 2003, Non-linear wave theory course notes. ceprofs.tamu.edu/jzhang/ *Texas A&M University*. Texas, USA.

[14] Warnitchai P, Pinhaew T, 1998, Modelling of liquid sloshing in rectangular tanks with flow damping devices, *Engineering Structures*, 20 593-600.

[15] Modi V J, Munshi S R, 1998, An efficient liquid sloshing damper for vibration control, *Journal of Fluids and Structures*, 12 (1998) 1055-1071.

[16] Schotte J S, Ohayon R, 2003, Effect of gravity on a free-free elastic tank partially filled with incompressible liquid, *Journal of Fluids and Structures*, 18 (2003) 215-226.

[17] Faltinsen OM, 1978, A numerical non-linear method of sloshing in tanks with two dimentional flow, *Journal of Ship Research*, 18 (4) 224-241.

[18] Frandsen J B, 2003, Simulation of sloshing motion in fixed and vertically excited containers using a 2-D inviscid σ-transformed finite difference solver, *Journal of Fluids and Structures*, 18 (2003) 197-214.

[19] Fluent.Inc, 2002 , www.fluent.com, *fluent v.6 user manual*, Lebanon, NH.

[20] Wilcox DC, 1998, Turbulence modelling for CFD, *DCW Industries Inc*., La Canada, California.

[21] Hirt C W, Nichols B D, 1981, Volume of Fluid (VOF) method for the dynamics of free boundaries, *Journal of Computational Physics*. 39, 201-225.

[22] Thomas B G, Zhang L, 2001, Mathematical Modelling of Fluid Flow in Continuous Casting, *ISIJ international*, vol. 41, 1181-1193.

[23] Chandrupatla T R, Belegundu A D, 1997, Finite Elements in Engineering, 2nd ED, *Prentice Hall.*

[24] Matthies H G, Steindorf J, 2002, Partitioned but Strongly Coupled Iteration Schemes for Nonlinear Fluid Structure Interaction, *Scientific Computing*, 2002-05.

[25] Hirt C W, Amsden A A, Cook J L, 1974, An Arbitrary Lagrangian-Eularian Computing Method for All Flow Speeds, *Journal of Computational Physics,* 14, 227.

[26] Livermore Software Technology Corporation (LSTC), 2004, www.lstc.com, *LS-DYNA 970 user manual*, Livermore, CA, USA.

[27] Le Sourne H, Couty N, Besnier F, 2003, LS-DYNA Applications in Shipbuilding, *4th European LS-DYNA User Conference*, Plenary Session II, A-11-01-16.

[28] LSTC, 2000, LS-DYNA. Training class in ALE and fluid-structure coupling. *Livermore Software Technology Corporation*. Livermore, USA.

[29] Hansbo P, Hermansson J, Svedberg T, 2003, Nitsche's method combined with space-time finite elements for ALE fluid structure interaction problems, *Chalmers Finite Element Centre*, PREPRINT 2003-14.

[30] Marzougui D, Cing-Dao Kan, Bedewi N E, 1996, Development and Validation of an NCAP simulation using LS-DYNA3D, *FHWA/NHTSA National crash Centre*, The George Washington University, Virginia, USA.

[31] Hadzic I, Mallon F, Peric M, 2001, Numerical Simulation of Sloshing, Fluent.Inc, *Application Brief EX161*, Tokyo, Japan.

[32] Snyman J A, 2000, The LFOPC leap-frog method for constrained optimization. *Computers Math. App*., 40 8/9 (2000), 1085-1096.

[33] Toolkit for Design Optimisation (TDO) v1.2a user guide, 2000. *Multidisciplinary Design Optimisation Group*, University of Pretoria, South Africa.

[34] Stander N, Egggleston T A, Craig K J, Roux W J, 2003, LS-OPT user manual v2, Design Optimisation Software for the Engineering Analyst, Livermore Software Technology Corporation, Livermore, CA.

[35] Snyman J A, 2005, Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms, *Springer*, New York, New York.

[36] Groenwold AA, Snyman JA, 2001, Global optimisation using dynamic search trajectories, *Global Optimisation*.

[37] Craig K J, Stander N, Dooge D A, Varadappa S, 2002, MDO of automotive vehicle for crashworthiness and NVH using response surface methods, AIAA paper 2002-5607, *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimizatio*n, Atlanta, GA, USA.

[38] Snyman J A, Hay A M, 2000, The Dynamic-Q Optimisation method: An alternative to SQP? *Journal of Computers and Mathematical Applications.*

[39] Craig K J, De Kock D J, Snyman J A, 1999, Using CFD and Mathematical Optimization to Investigate Air Pollution due to Stacks, *International Journal for Numerical Methods in Engineering*, Vol. 44, pp 551-565.

[40] Myers R H, Montgomery D C, 1995, Response Surface Methodology, Process and Product Optimisation using Design Experiments, *Wiley*.

[41] INGENET website, http://ingenet.ulpgc.es , NETworked INdustrial Design and Control Applications using GEnetic Algorithms and Evolution Strategies.

[42] Qu X, Haftka T, Venter G, 2003, New Formulation of a Minimum-Bias Experimental Design Based on Gauss Quadrature. Proceedings of 5th World Congress on Multidisciplinary Design Optimization, Lido di Jesolo-Venice.

[43] Stander N, Craig K J, 2002, On the robustness of a simple domain reduction scheme for simulation-based optimization. *Engineering Computations*, Vol. 19 No. 4, 431 – 450.

[44] Bishop C M, 1995, Neural Networks for Pattern Recognition, *Oxford University Press*. ISBN: 0198538642.

[45] Simpson T W, 1998, A concept exploration method for product family design. Ph. D. Thesis, *Georgia Institute of Technology*.

[46] Stander N, Roux W J, Giger M, Redhe M, Fedorova N, Haarhoff J, 2003, Crashworthiness optimization in LS-OPT: Case studies in metamodeling and random search techniques. *Proceedings of the 4th European LS-DYNA Conference*, Ulm, Germany.

[47] Jin R, Chen W, Simpson T W, 2000, Comparative studies of metamodeling techniques under multiple modelling criteria, *AIAA* Paper AIAA-2000-4801.

[48] Zang T A, Green L L, 1999, Multidisciplinary Design Optimisation Techniques: Implications and Opportunities for Fluid Dynamics Research, *AIAA*, 99-3798.

[49] Giesing J P, Barthelemy J F, 1998, A Summary of Industry MDO Applications and Needs, 7-th *AIAA/USAF/NASA/ISSMO*, St. Louis.

[50] White F M, 2003, Fluid Mechanics, 5th ed. *McGraw Hill*, USA.

[51] Craig K J, Kingsley T C, Dieterich R, Haarhoff, L J & Stander N, 2003, Design optimization of the fluid-structure interaction in a fuel tank, *16th AIAA Computational Fluid Dynamics Conference*, Orlando, FA, 23-26 June 2003.

[52] Mathworks.Inc, 2003, Matlab users guide v6, www.mathworks.com

[53] Dieterich R, 2002. Geometric Design Optimisation of a Fuel Tank in Order to Minimize Sloshing. Undergraduate thesis report, *University of Pretoria*, South Africa.

[54] Craig K J, Stander N, Dooge D A, Varadappa S, 2005, Automotive crashworthiness design using response surface-based variable screening and optimization, *Engineering Computations*, Vol.22 No.1, 38-61.

APPENDICES

APPENDIX A: Sample 2D GAMBIT journal file

```
/ Combined vertical and horizontal baffles
/ Created by Thomas Kingsley

solver select "FLUENT 5/6"

 $MB_cent = <<mid_baf_centroid>>
 $SB_cent = <<side_baf_centroid>>
 $MB_height = <<mid_baf_height>>
 $SB_wide = <<side_baf_width>>

/ Geometry

vertex create coordinates 0 0 0
vertex create coordinates 0 100 0
vertex create coordinates 0 200 0
edge create straight "vertex.2" "vertex.3"
edge create straight "vertex.1" "vertex.2"
vertex create coordinates ($SB_cent-$SB_wide/2) 0 0
edge create straight "vertex.1" "vertex.4"
vertex create coordinates ($SB_cent+$SB_wide/2) 0 0
edge create straight "vertex.4" "vertex.5"
face create translate "edge.2" onedge "edge.3"
face create translate "edge.6" onedge "edge.4"
face create translate "edge.1" onedge "edge.7"
face create translate "edge.12" onedge "edge.10"
vertex create coordinates 200 0 0
vertex create coordinates 200 ($MB_cent-$MB_height/2) 0
vertex create coordinates 200 ($MB_cent+$MB_height/2) 0
vertex create coordinates 200 200 0
edge create straight "vertex.16" "vertex.17"
edge create straight "vertex.15" "vertex.16"
edge create straight "vertex.14" "vertex.15"
edge create straight "vertex.5" "vertex.14"
edge create straight "vertex.13" "vertex.17"
vertex create coordinates 400 0 0
vertex create coordinates (400-($SB_cent-$SB_wide/2)) 0 0
vertex create coordinates (400-($SB_cent+$SB_wide/2)) 0 0
vertex create coordinates 400 100 0
vertex create coordinates 400 200 0
edge create straight "vertex.21" "vertex.22"
edge create straight "vertex.18" "vertex.21"
edge create straight "vertex.18" "vertex.19"
edge create straight "vertex.19" "vertex.20"
face create translate "edge.23" onedge "edge.24"
face create translate "edge.22" onedge "edge.28"
face create translate "edge.27" onedge "edge.25"
face create translate "edge.30" onedge "edge.34"
edge create straight "vertex.17" "vertex.30"
edge create straight "vertex.14" "vertex.20"
face create wireframe "edge.38" "edge.17" "edge.18" "edge.19" "edge.39" \
  "edge.33" "edge.36" real
face create wireframe "edge.20" "edge.21" "edge.15" "edge.18" "edge.9" \
  "edge.19" "edge.17" real

/ Mesh

face mesh "face.1" "face.2" "face.3" "face.4" "face.5" "face.6" "face.7" \
  "face.8" "face.9" "face.10" map size 2

/ BC

physics create "baffles" btype "WALL" edge "edge.10" "edge.18" "edge.34"

export fluent5 "2D_VF_tank.msh" nozval

abort
```

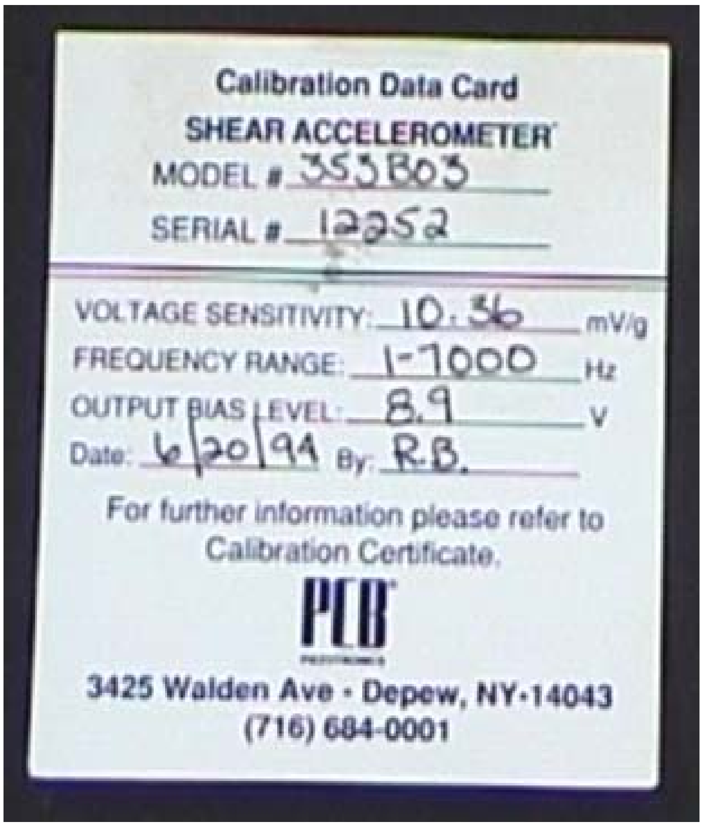
APPENDIX B: Acceleration to momentum source conversion code

```
#include "udf.h"
#include <stdio.h>
#include <stdlib.h>
#define timeend 4


real AlnrX,delt=0.015625;

DEFINE_ADJUST(Accl, domain)
{
        FILE *aptr = fopen("accelprof.txt","r");

        real t,line,lower,upper,tup,tlow;
        int i;
        char temp[20];

        t = RP_Get_Real("flow-time");

        if (t<=timeend) {
                line = (t/delt);
                for (i=0;i<=line;i++) {
                        fgets(temp, 20, aptr);
                }
                lower = atof(temp);
                fgets(temp, 20, aptr);
                upper = atof(temp);
                tup = i*delt;
                tlow = (i-1)*delt;
                AlnrX = ((tup-t)/(tup-tlow)*(upper-lower)-upper);

                /*    printf("interp = %f\n",((tup-t)/(tup-tlow)*(upper-lower)-
upper));*/
        }
        else {
                AlnrX = 0.0;
        }
        printf("time = %f\n",t);
        printf("Acceleration = %f\n",AlnrX);

        /*  printf("upper = %f\n",upper);
         *  printf("lower = %f\n",lower);
         *  printf("tup = %f\n",tup);
         *  printf("tlow = %f\n",tlow);*/
        fclose(aptr);
}


DEFINE_SOURCE(xmom, cell, thread, dS, eqn)
{
        double pos[ND_ND];
        double rho;
        double source;
        C_CENTROID(pos,cell,thread);

        rho  = C_R(cell,thread);

        dS[eqn] = 0.0;
        source = AlnrX *rho;

        return source;
}
```

APPENDIX C: Sample 2D Fluent journal file

```
f/rc "2D_VF_tank.msh"
g/s 0.001 0.001
de/m u1o y
de/m/multi vof 2 geo-reconstruct 0.25 no yes
de/m/v lam y
de/mat/cc air water y constant 998 n n y constant 0.001 n n n n n n
de/ph/pd phase-2 phase-2 y water
de/m/multi vof 2 geo-reconstruct 0.25 no yes
;de/ud/cf compile libudf y /home/thomas/2dslosh/LSOPT/LRF_udf_dp.c
de/ud/cf load ../../libudf
de/bc/fl fluid mixture y n y
"udf"
"xmom_source"
n n n y 0 0 n n
de/ud/fh
"none"
"Accl"
"none"
"none"
"none"
"none"
de/oc gr y 0 -9.81
so/se/dis-s p 13
so/se/dis-s f 22
so/se/dis-s m 0
so/se/ur p 0.8
so/se/ur d 1
so/se/ur bf 1
so/se/ur m 0.8
so/se/ur mp 1
so/in/sd/p2/mp 1
so/in if
so/se ts 0.00025
(rpsetvar 'piso/skew-iter 0)
a mir y n -1 1 <<fill_level>> 1
so/mo/re/pl y
(load "../../tui_patch.txt")
(patch (arg-patch-dom 'phase-2 'mp 0 () (list 'hexahedron-r0)))
(rpsetvar 'monitor/commands '((command-1 40 #t "su/is/phase-2 vof
free-surf () 0.5 ()") (command-2 40 #t "f/e/as data_%t free-surf () n
y yc q y") (command-3 40 #t "su/ds/free-surf")))
s/dti 8000 50
exit y
```

APPENDIX D: Accelerometer calibration certificate



Calibration Data Card

SHEAR ACCELEROMETER

MODEL # 353B03

SERIAL # 12252

VOLTAGE SENSITIVITY: 10.36 mV/g

FREQUENCY RANGE: 1-7000 Hz

OUTPUT BIAS LEVEL: 8.9 V

Date: 6/20/94 By: R.B.

For further information please refer to Calibration Certificate.

PCB

3425 Walden Ave • Depew, NY•14043

(716) 684-0001

APPENDIX E: Calibration of WIKA pressure sensor

Calibration of the WIKA pressure sensor was done using the equipment used in Figure E1 below. The equipment is the same as that which was used during the experimental testing and includes a laptop, data logger, WIKA pressure sensor, and power converter. Coloured fluid was poured into a Perspex cylinder to create the desired head of water. The level of water was then measured using a 1 meter ruler and referenced to a voltage reading that came from the equipment. Figure E2 below shows a comparison of the measured voltages compared with those expected from the pressure sensor based on the manufacturer's provided ratings.

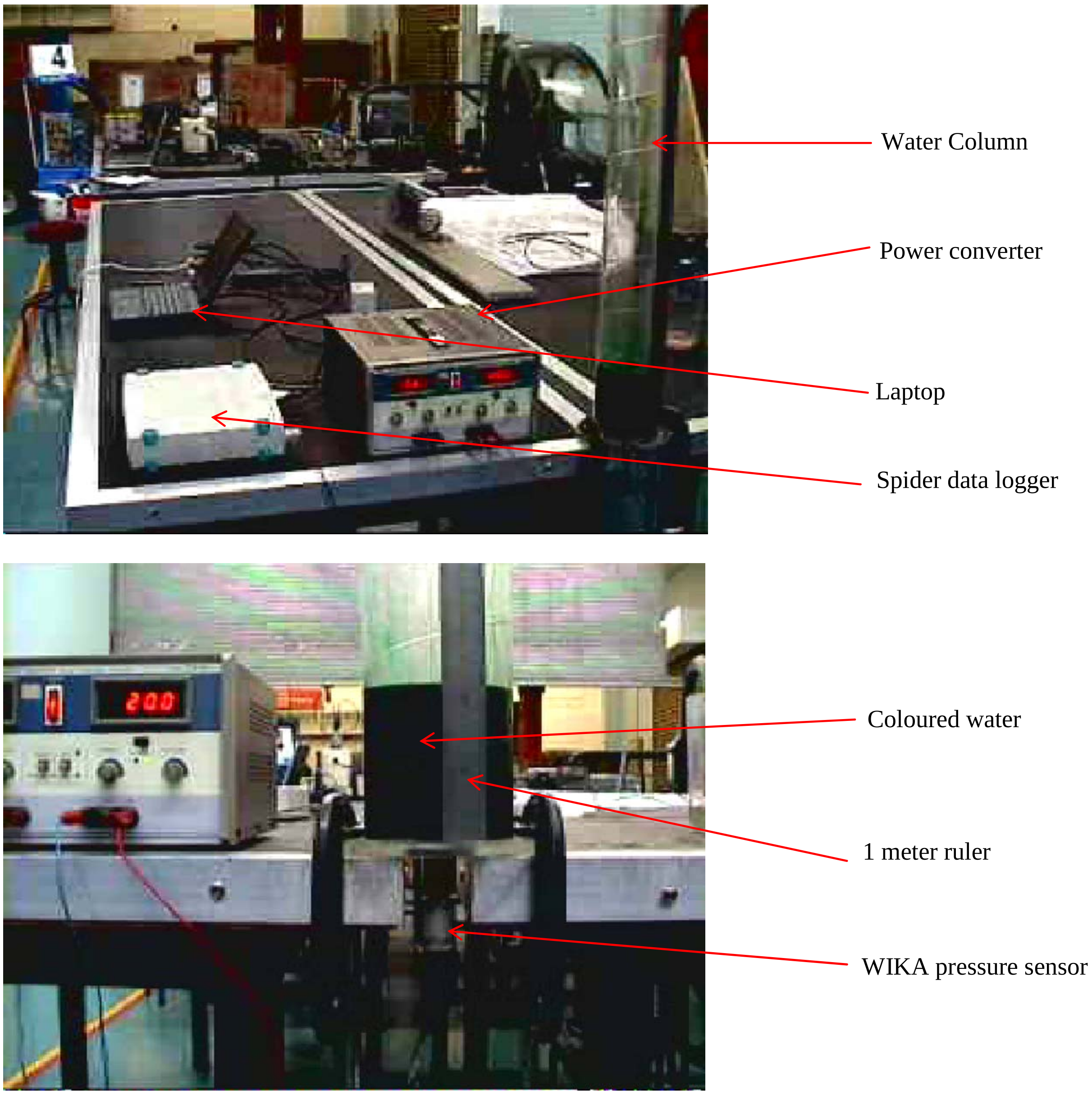


Figure E 1: WIKA pressure sensor calibration setup

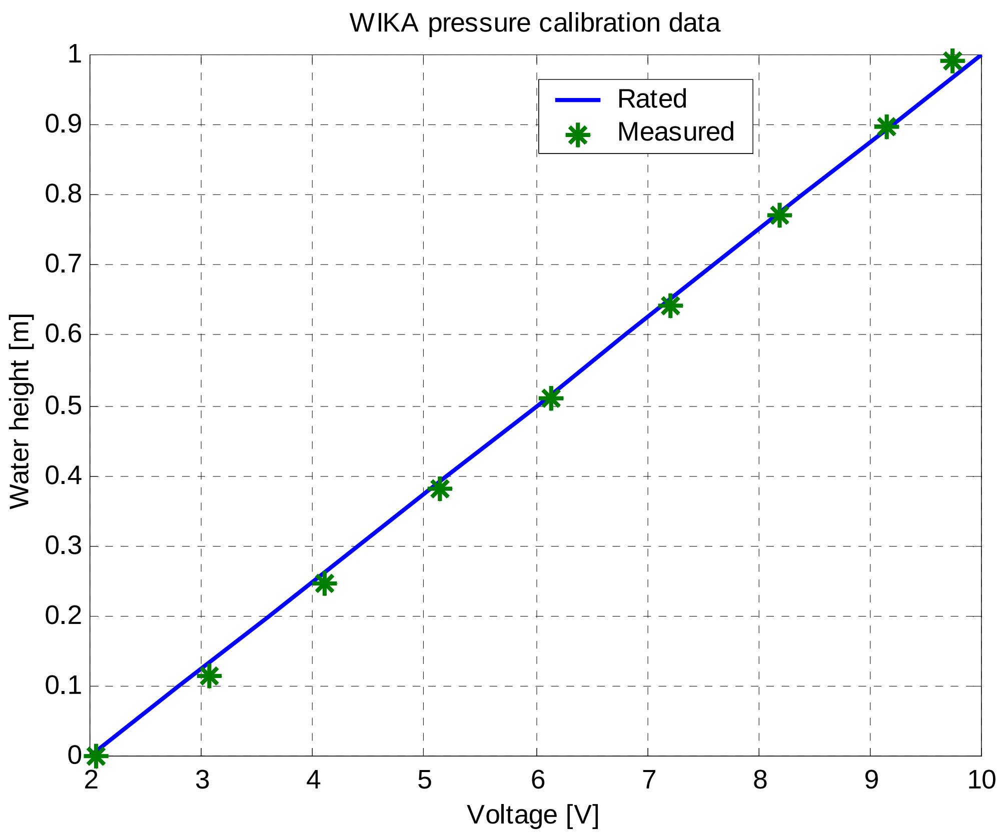


Figure E 2: Measured data versus rated voltages

Is was determined that the voltages were within acceptable limits for the purposes of the experiments. The R2 deviation from the rated line was 0.9995.

APPENDIX F: 3D optimisation LSOPT command file

```
"3D slosh opt"
Author "Rolf Deiterich"
$ Created on Fri Sep  6 00:44:41 2002
$
$ DESIGN VARIABLES
$
variables 2
 Variable 'LSholedia' 40
  Lower bound variable 'LSholedia' 15
  Upper bound variable 'LSholedia' 80
 Variable 'LSbaffleheight' 300
  Lower bound variable 'LSbaffleheight' 80
  Upper bound variable 'LSbaffleheight' 380
solvers 1
responses 6
$
$ NO HISTORIES ARE DEFINED
$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "/home/rolf/build/Rolf_fluent_script"
  solver input file "/home/rolf/build/fluentjourolf.jou"
$  prepro own
$  prepro command "gambit -inp"
$  prepro input file "/home/rolf/build/lsgridgen.jou"
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'Ay' {(LSbaffleheight-(2*LSholedia))/(1+2)}
 response 'Ay' linear
 response 'Az' {((400-(4*LSholedia))/(1+4))}
 response 'Az' linear
 response 'smallgap' {((LSbaffleheight-(2*LSholedia))/3)-(LSholedia/2)}
 response 'smallgap' linear
 response 'areamin' {(400*LSbaffleheight)-(8*3.14*LSholedia*LSholedia)}
 response 'areamin' linear
 response 'Max_Dev_from_file' 1.0 0.0 "../../fluentc.out"
 response 'Max_Dev_from_file' linear
 response 'Funnyval_from_file' 1.0 0.0 "../../funnyval.out"
 response 'Funnyval_from_file' linear
$
$ NO HISTORIES DEFINED FOR SOLVER "fluent"
$
$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'Funnyval_from_file'
$
$ CONSTRAINT DEFINITIONS
$
 constraints 4

 constraint 'areamin'
  strict
  upper bound constraint 'areamin' 95000
  slack
 constraint 'Ay'
  strict
  lower bound constraint 'Ay' 5
  slack
 constraint 'Az'
  strict
  lower bound constraint 'Az' 10
  slack
 constraint 'smallgap'
  strict
  lower bound constraint 'smallgap' 8
  slack
```

```
$
$ EXPERIMENTAL DESIGN
$
 Order linear
 Experimental design dopt
 Number experiment 5
$
$ JOB INFO
$
 concurrent jobs 1
 iterate param design 0.01
 iterate param objective 0.01
 iterate 1
STOP
```

APPENDIX G: 3D baffled tank GAMBIT journal file

```
/date 2002 09 22  Time 22:28 (By Rolf Deiterich)
/substituted file

RESET
SOLVER SELECT "FLUENT 5/6"
default set "MESH.FACE.AUTO_SMOOTH" numeric 0
default set "MESH.VOLUME.AUTO_SMOOTH" numeric 0


/define tank
$tank_height=400
$tank_width=400
$tank_length=500
$tank_half_width=($tank_width/2)

/define gaps
$tank_baffle_height=187.7
$bottom_gap_height=(($tank_height-$tank_baffle_height)/2)
$top_gap_height=$bottom_gap_height

/define line coords
$line1y=0
$line2y=$bottom_gap_height
$line4y=$tank_height
$line3y=($tank_height-$top_gap_height)

/define holes
$Nholes_accross=4
$half_holes_accross=(($Nholes_accross/2))
$Nholes_up=2

$D=15
$R=($D/2)
$holez=($R*COS(45))
$holey=($R*SIN(45))
$squaremove=($R/2)

/MESH
$meshinter=3

$meshsize=3
$NHmeshsize=5


$ratio1=1.02
$ratio2=1.02
$number_of_baffles=4
$copyntimes=($number_of_baffles)
$vollength=($tank_length/($number_of_baffles+1))

/holes accross z
$Az=(($tank_width-($Nholes_accross*$D))/(1+$Nholes_accross))

/holes up y
$Ay=(($tank_baffle_height-($Nholes_up*$D))/(1+$Nholes_up))

/????????????????????????SET MESH SIZE????????????????
$Aymeshsize=5
IF COND ($Ay .LE. 48)
            $Aymeshsize=4
ENDIF

IF COND ($Ay .LE. 32)
            $Aymeshsize=3
ENDIF

IF COND ($Ay .LE. 16)
            $Aymeshsize=2
ENDIF
//
```

```
$Azmeshsize=5
IF COND ($Az .LE. 48)
        $Azmeshsize=4
ENDIF

IF COND ($Az .LE. 32)
        $Azmeshsize=3
ENDIF

IF COND ($Az .LE. 16)
        $Azmeshsize=2
ENDIF

/???????????????????????SET MESH SIZE????????????????


/create base points
vertex create "p1" coordinates 0 0 0
vertex create "p2" coordinates 0 0 $tank_half_width
vertex create "p3" coordinates 0 $bottom_gap_height 0
vertex create "p4" coordinates 0 $bottom_gap_height $tank_half_width
vertex create "p5" coordinates 0 ($tank_height-$top_gap_height) 0
vertex create "p6" coordinates 0 ($tank_height-$top_gap_height) $tank_half_width
vertex create "p7" coordinates 0 $tank_height 0
vertex create "p8" coordinates 0 $tank_height $tank_half_width


/Create the holes accross and then up
/y is up z is accross
$y=1
$z=1
DO PARA "$y" INIT 1 COND ($y .le. $Nholes_up) INCR 1
        DO PARA "$z" INIT 1 COND ($z .le. $half_holes_accross) INCR 1
                IF COND($z .EQ. 1)
                        $("zhole"+NTOS($y)+NTOS($z))=$Az+$R
                ELSE
                        $("zhole"+NTOS($y)+NTOS($z))=$("zhole"+NTOS($y)+NTOS($z-1))+$Az+$D
                ENDIF

                IF COND($y .EQ. 1)
                        $("yhole"+NTOS($y)+NTOS($z))=$bottom_gap_height+$Ay+$R
                                                                /=$bottom_gap_height+$Ay+$R
                ELSE
                        $("yhole"+NTOS($y)+NTOS($z))=$("yhole"+NTOS($y-1)+NTOS($z))+$Ay+$D
                ENDIF

                vertex create ("hv"+NTOS($z)+NTOS($y)+"cv")   coordinates 0 $("yhole"+NTOS($y)+NTOS($z))
$("zhole"+NTOS($y)+NTOS($z))
                vertex create ("hv"+NTOS($z)+NTOS($y)+"1")    coordinates 0 ($("yhole"+NTOS($y)+NTOS($z))-$R)
$("zhole"+NTOS($y)+NTOS($z))
                vertex create ("hv"+NTOS($z)+NTOS($y)+"2")    coordinates 0 ($("yhole"+NTOS($y)+NTOS($z))-
$holey)      ($("zhole"+NTOS($y)+NTOS($z))-$holez)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"3")    coordinates 0 $("yhole"+NTOS($y)+NTOS($z))
($("zhole"+NTOS($y)+NTOS($z))-$R)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"4")    coordinates 0
($("yhole"+NTOS($y)+NTOS($z))+$holey)      ($("zhole"+NTOS($y)+NTOS($z))-$holez)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"5")    coordinates 0 ($("yhole"+NTOS($y)+NTOS($z))+$R)
$("zhole"+NTOS($y)+NTOS($z))
                vertex create ("hv"+NTOS($z)+NTOS($y)+"6")    coordinates 0
($("yhole"+NTOS($y)+NTOS($z))+$holey)      ($("zhole"+NTOS($y)+NTOS($z))+$holez)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"7")    coordinates 0 $("yhole"+NTOS($y)+NTOS($z))
($("zhole"+NTOS($y)+NTOS($z))+$R)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"8")    coordinates 0 ($("yhole"+NTOS($y)+NTOS($z))-
$holey)      ($("zhole"+NTOS($y)+NTOS($z))+$holez)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"9")    coordinates 0 ($("yhole"+NTOS($y)+NTOS($z))-
$squaremove)   $("zhole"+NTOS($y)+NTOS($z))
                vertex create ("hv"+NTOS($z)+NTOS($y)+"10")   coordinates 0 $("yhole"+NTOS($y)+NTOS($z))
($("zhole"+NTOS($y)+NTOS($z))-$squaremove)
                vertex create ("hv"+NTOS($z)+NTOS($y)+"11")   coordinates 0
($("yhole"+NTOS($y)+NTOS($z))+$squaremove)   $("zhole"+NTOS($y)+NTOS($z))
                vertex create ("hv"+NTOS($z)+NTOS($y)+"12")   coordinates 0 $("yhole"+NTOS($y)+NTOS($z))
($("zhole"+NTOS($y)+NTOS($z))+$squaremove)
```

```
                    edge create ("he"+NTOS($z)+NTOS($y)+"1")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"1")   ("hv"+NTOS($z)+NTOS($y)+"2")
                    edge create ("he"+NTOS($z)+NTOS($y)+"2")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"2")   ("hv"+NTOS($z)+NTOS($y)+"3")
                    edge create ("he"+NTOS($z)+NTOS($y)+"3")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"3")   ("hv"+NTOS($z)+NTOS($y)+"4")
                    edge create ("he"+NTOS($z)+NTOS($y)+"4")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"4")   ("hv"+NTOS($z)+NTOS($y)+"5")
                    edge create ("he"+NTOS($z)+NTOS($y)+"5")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"5")   ("hv"+NTOS($z)+NTOS($y)+"6")
                    edge create ("he"+NTOS($z)+NTOS($y)+"6")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"6")   ("hv"+NTOS($z)+NTOS($y)+"7")
                    edge create ("he"+NTOS($z)+NTOS($y)+"7")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"7")   ("hv"+NTOS($z)+NTOS($y)+"8")
                    edge create ("he"+NTOS($z)+NTOS($y)+"8")  arc center2points  ("hv"+NTOS($z)+NTOS($y)+"cv")
("hv"+NTOS($z)+NTOS($y)+"8")   ("hv"+NTOS($z)+NTOS($y)+"1")
                    edge create ("he"+NTOS($z)+NTOS($y)+"9")  straight
("hv"+NTOS($z)+NTOS($y)+"9")   ("hv"+NTOS($z)+NTOS($y)+"10")
                    edge create ("he"+NTOS($z)+NTOS($y)+"10")  straight
("hv"+NTOS($z)+NTOS($y)+"10")   ("hv"+NTOS($z)+NTOS($y)+"11")
                    edge create ("he"+NTOS($z)+NTOS($y)+"11")  straight
("hv"+NTOS($z)+NTOS($y)+"11")   ("hv"+NTOS($z)+NTOS($y)+"12")
                    edge create ("he"+NTOS($z)+NTOS($y)+"12")  straight
("hv"+NTOS($z)+NTOS($y)+"12")   ("hv"+NTOS($z)+NTOS($y)+"9")
                    edge create ("he"+NTOS($z)+NTOS($y)+"13")  straight
("hv"+NTOS($z)+NTOS($y)+"9")   ("hv"+NTOS($z)+NTOS($y)+"1")
                    edge create ("he"+NTOS($z)+NTOS($y)+"14")  straight
("hv"+NTOS($z)+NTOS($y)+"10")   ("hv"+NTOS($z)+NTOS($y)+"3")
                    edge create ("he"+NTOS($z)+NTOS($y)+"15")  straight
("hv"+NTOS($z)+NTOS($y)+"11")   ("hv"+NTOS($z)+NTOS($y)+"5")
                    edge create ("he"+NTOS($z)+NTOS($y)+"16")  straight
("hv"+NTOS($z)+NTOS($y)+"12")   ("hv"+NTOS($z)+NTOS($y)+"7")


          ENDDO
ENDDO


/create bottom and top gap verticies

DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross) INCR 1
          vertex create ("l1p"+NTOS($z)) coordinates 0 $line1y   $("zhole1"+NTOS($z))
          vertex create ("l2p"+NTOS($z)) coordinates 0 $line2y   $("zhole1"+NTOS($z))
          vertex create ("l3p"+NTOS($z)) coordinates 0 $line3y   $("zhole1"+NTOS($z))
          vertex create ("l4p"+NTOS($z)) coordinates 0 $line4y   $("zhole1"+NTOS($z))
ENDDO


/create vertical verticies on either side of baffle

DO PARA "$y" INIT 1 COND ($y .LE. $Nholes_up) INCR 1
          vertex create ("hlv"+NTOS($y)+"left")    coordinates 0 $("yhole"+NTOS($y)+"1")    0
          vertex create ("hlv"+NTOS($y)+"right")   coordinates 0 $("yhole"+NTOS($y)+"1")    $tank_half_width
ENDDO


/GAP

/create horisontal edges on bottom and top gap (always exist)

edge create "l1s1"                          straight "p1"                        "l1p1"
edge create ("l1s"+NTOS($half_holes_accross+1))  straight ("l1p"+NTOS($half_holes_accross)) "p2"

edge create "l2s1"                          straight "p3"                        "l2p1"
edge create ("l2s"+NTOS($half_holes_accross+1))  straight ("l2p"+NTOS($half_holes_accross)) "p4"
edge create "l3s1"                          straight "p5"                        "l3p1"
edge create ("l3s"+NTOS($half_holes_accross+1))  straight ("l3p"+NTOS($half_holes_accross)) "p6"

edge create "l4s1"                          straight "p7"                        "l4p1"
edge create ("l4s"+NTOS($half_holes_accross+1))  straight ("l4p"+NTOS($half_holes_accross)) "p8"

/need this single gap
/egde create "l1s1" straight "p1 "p2"
/egde create "l1s1" straight "p7 "p8"
```

```
/create horizontal edges on bottom and top gap (if needed)

IF COND($half_holes_accross .GT. 1)
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                edge create ("l1s"+NTOS($z+1)) straight ("l1p"+NTOS($z)) ("l1p"+NTOS($z+1))
                edge create ("l2s"+NTOS($z+1)) straight ("l2p"+NTOS($z)) ("l2p"+NTOS($z+1))
                edge create ("l3s"+NTOS($z+1)) straight ("l3p"+NTOS($z)) ("l3p"+NTOS($z+1))
                edge create ("l4s"+NTOS($z+1)) straight ("l4p"+NTOS($z)) ("l4p"+NTOS($z+1))
        ENDDO
ENDIF

/create vertical gap edges that always exist

edge create ("l5s1")              straight "p1" "p3"
edge create ("l6s1")              straight "p2" "p4"
edge create ("l5s"+NTOS($Nholes_up+3))  straight "p5" "p7"
edge create ("l6s"+NTOS($Nholes_up+3))  straight "p6" "p8"


/create vertical gap edges that might exist
IF COND($half_holes_accross .GE. 1)
        DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross) INCR 1
                edge create ("bgl"+NTOS($z)) straight ("l1p"+NTOS($z)) ("l2p"+NTOS($z))
                edge create ("tgl"+NTOS($z)) straight ("l3p"+NTOS($z)) ("l4p"+NTOS($z))
        ENDDO
ENDIF

/create vertical baffle lines that always exist

edge create ("l5s2")                straight ("p3")                 ("hlv1left")
edge create ("l5s"+NTOS($Nholes_up+2))    straight ("hlv"+NTOS($Nholes_up)+"left")    ("p5")
edge create ("l6s2")                straight ("p4")                 ("hlv1right")
edge create ("l6s"+NTOS($Nholes_up+2))    straight ("hlv"+NTOS($Nholes_up)+"right")   ("p6")

/create vertical baffle line
IF COND($half_holes_accross .GT. 1)
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                edge create ("l5s"+NTOS($y+2))   straight ("hlv"+NTOS($y)+"left")  ("hlv"+NTOS($y+1)+"left")
                edge create ("l6s"+NTOS($y+2))   straight ("hlv"+NTOS($y)+"right") ("hlv"+NTOS($y+1)+"right")
        ENDDO
ENDIF
/create internal baffle lines v bottom

DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross) INCR 1
        edge create ("lbl"+NTOS($z)) straight ("l2p"+NTOS($z)) ("hv"+NTOS($z)+"11")
ENDDO

/create internal baffle lines v midle
IF COND($half_holes_accross .GT. 1)
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up)
                DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross) INCR 1
                        edge create ("bivl"+NTOS($y)+NTOS($z)) straight ("hv"+NTOS($z)+NTOS($y)+"5")
("hv"+NTOS($z)+NTOS($y+1)+"1")
                ENDDO
        ENDDO
ENDIF

/create internal baffle lines v top
DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross) INCR 1
        edge create ("ubl"+NTOS($z)) straight ("hv"+NTOS($z)+NTOS($Nholes_up)+"5") ("l3p"+NTOS($z))
ENDDO

/create internal baffle lines h left
DO PARA "$y" INIT 1 COND ($y .LE. $Nholes_up) INCR 1
        edge create ("lhbl"+NTOS($y)) straight ("hlv"+NTOS($y)+"left") ("hv1"+NTOS($y)+"3")
ENDDO


/create internal baffle lines h middle
IF COND($half_holes_accross .GT. 1)
        DO PARA "$y" INIT 1 COND ($y .LE. $Nholes_up)
                DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                        edge create ("bihl"+NTOS($y)+NTOS($z)) straight ("hv"+NTOS($z)+NTOS($y)+"7")
("hv"+NTOS($z+1)+NTOS($y)+"3")
                ENDDO
```

```
		ENDDO
ENDIF

/create internal baffle lines h right
DO PARA "$y" INIT 1 COND ($y .LE. $Nholes_up) INCR 1
		edge create ("rhbl"+NTOS($y)) straight ("hv"+NTOS($half_holes_accross)+NTOS($y)+"7")
("hlv"+NTOS($y)+"right")
ENDDO

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/FACES
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/create faces on bottom gap
/lower gap face

face create "lgfleft"   wireframe "l1s1" "l5s1" "l2s1" "bgl1"
face create "lgfright"  wireframe ("l1s"+NTOS($half_holes_accross+1)) ("bgl"+NTOS($half_holes_accross))
("l2s"+NTOS($half_holes_accross+1)) "l6s1"

IF COND($half_holes_accross .GT. 1)
		DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
				face create ("lgf"+NTOS($z)) wireframe ("l1s"+NTOS($z+1)) ("bgl"+NTOS($z)) ("l2s"+NTOS($z+1))
("bgl"+NTOS($z+1))
		ENDDO
ENDIF

/create faces on top gap
/upper gap face

face create "ugfleft"   wireframe ("l3s1") ("l5s"+NTOS($Nholes_up+3)) "l4s1" "tgl1"
face create "ugfright"  wireframe ("l3s"+NTOS($half_holes_accross+1)) ("tgl"+NTOS($half_holes_accross))
("l4s"+NTOS($half_holes_accross+1)) ("l6s"+NTOS($Nholes_up+3))

IF COND($half_holes_accross .GT. 1)
		DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
				face create ("ugf"+NTOS($z)) wireframe ("l3s"+NTOS($z+1)) ("tgl"+NTOS($z)) ("l4s"+NTOS($z+1))
("tgl"+NTOS($z+1))
		ENDDO
ENDIF

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/create lower baffle faces that always exist

face create "bf11" wireframe "l2s1" "l5s2" "lhbl1" "he112" "he111" "lbl1"
face create ("bf1"+NTOS($half_holes_accross+1)) wireframe ("l2s"+NTOS($half_holes_accross+1))
("lbl"+NTOS($half_holes_accross)) ("he"+NTOS($half_holes_accross)+"18") ("he"+NTOS($half_holes_accross)+"17") "rhbl1"
"l6s2"

/create lower baffle faces that might exist

IF COND($half_holes_accross .GT. 1)
		DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
				face create ("bf1"+NTOS($z+1)) wireframe ("l2s"+NTOS($z+1)) ("lbl"+NTOS($z))
("he"+NTOS($z)+"18") ("he"+NTOS($z)+"17") ("bihl1"+NTOS($z)) ("he"+NTOS($z+1)+"12") ("he"+NTOS($z+1)+"11")
("lbl"+NTOS($z+1))
		ENDDO
ENDIF

/create upper baffle faces that always exist

face create ("bf"+NTOS($Nholes_up+1)+"1") wireframe ("l3s1") ("l5s"+NTOS($Nholes_up+2)) ("lhbl"+NTOS($Nholes_up))
("he1"+NTOS($Nholes_up)+"3") ("he1"+NTOS($Nholes_up)+"4") "ubl1"
face create ("bf"+NTOS($Nholes_up+1)+NTOS($half_holes_accross+1)) wireframe ("l3s"+NTOS($half_holes_accross+1))
("ubl"+NTOS($half_holes_accross)) ("he"+NTOS($half_holes_accross)+NTOS($Nholes_up)+"5")
("he"+NTOS($half_holes_accross)+NTOS($Nholes_up)+"6") ("rhbl"+NTOS($Nholes_up)) ("l6s"+NTOS($Nholes_up+2))

/create upper baffle faces that might exist

IF COND($half_holes_accross .GT. 1)
		DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
				face create ("bf"+NTOS($Nholes_up+1)+NTOS($z+1)) wireframe ("l3s"+NTOS($z+1))
("ubl"+NTOS($z)) ("he"+NTOS($z)+NTOS($Nholes_up)+"5") ("he"+NTOS($z)+NTOS($Nholes_up)+"6")
```

```
("bihl"+NTOS($Nholes_up)+NTOS($z)) ("he"+NTOS($z+1)+NTOS($Nholes_up)+"3")
("he"+NTOS($z+1)+NTOS($Nholes_up)+"4") ("ubl"+NTOS($z+1))
        ENDDO
ENDIF


/LONG IF

IF COND($half_holes_accross .GT. 1)

        /create middle faces left
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                face create ("bf"+NTOS($y+1)+"1") wireframe ("lhbl"+NTOS($y)) ("l5s"+NTOS($y+2))
("lhbl"+NTOS($y+1)) ("he1"+NTOS($y+1)+"2") ("he1"+NTOS($y+1)+"1") ("bivl"+NTOS($y)+"1") ("he1"+NTOS($y)+"4")
("he1"+NTOS($y)+"3")
        ENDDO

        /create middle faces right
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                face create ("bf"+NTOS($y+1)+NTOS($half_holes_accross+1)) wireframe ("rhbl"+NTOS($y))
("l6s"+NTOS($y+2)) ("rhbl"+NTOS($y+1)) ("he"+NTOS($half_holes_accross)+NTOS($y+1)+"8")
("he"+NTOS($half_holes_accross)+NTOS($y+1)+"7") ("bivl"+NTOS($y)+NTOS($half_holes_accross))
("he"+NTOS($half_holes_accross)+NTOS($y)+"5") ("he"+NTOS($half_holes_accross)+NTOS($y)+"6")
        ENDDO


        /create middle faces rest
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross)
                        face create ("bf"+NTOS($y+1)+NTOS($z+1)) wireframe ("bihl"+NTOS($y)+NTOS($z))
("he"+NTOS($z)+NTOS($y)+"6") ("he"+NTOS($z)+NTOS($y)+"5") ("bivl"+NTOS($y)+NTOS($z))
("he"+NTOS($z)+NTOS($y+1)+"8") ("he"+NTOS($z)+NTOS($y+1)+"7") ("bihl"+NTOS($y+1)+NTOS($z))
("he"+NTOS($z+1)+NTOS($y+1)+"2") ("he"+NTOS($z+1)+NTOS($y+1)+"1") ("bivl"+NTOS($y)+NTOS($z+1))
("he"+NTOS($z+1)+NTOS($y)+"4") ("he"+NTOS($z+1)+NTOS($y)+"3")

                ENDDO
        ENDDO
ENDIF


/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/create hole faces
DO PARA "$y" INIT 1 COND ($y .LE. $Nholes_up) INCR 1
        DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross ) INCR 1
                face create ("hf"+NTOS($y)+NTOS($z)+"1") wireframe ("he"+NTOS($z)+NTOS($y)+"1")
("he"+NTOS($z)+NTOS($y)+"2")  ("he"+NTOS($z)+NTOS($y)+"14") ("he"+NTOS($z)+NTOS($y)+"9")
("he"+NTOS($z)+NTOS($y)+"13")
                face create ("hf"+NTOS($y)+NTOS($z)+"2") wireframe ("he"+NTOS($z)+NTOS($y)+"14")
("he"+NTOS($z)+NTOS($y)+"3")  ("he"+NTOS($z)+NTOS($y)+"4")  ("he"+NTOS($z)+NTOS($y)+"15")
("he"+NTOS($z)+NTOS($y)+"10")
                face create ("hf"+NTOS($y)+NTOS($z)+"3") wireframe ("he"+NTOS($z)+NTOS($y)+"16")
("he"+NTOS($z)+NTOS($y)+"11") ("he"+NTOS($z)+NTOS($y)+"15") ("he"+NTOS($z)+NTOS($y)+"5")
("he"+NTOS($z)+NTOS($y)+"6")
                face create ("hf"+NTOS($y)+NTOS($z)+"4") wireframe ("he"+NTOS($z)+NTOS($y)+"7")
("he"+NTOS($z)+NTOS($y)+"8")  ("he"+NTOS($z)+NTOS($y)+"13") ("he"+NTOS($z)+NTOS($y)+"12")
("he"+NTOS($z)+NTOS($y)+"16")
                face create ("hf"+NTOS($y)+NTOS($z)+"5") wireframe ("he"+NTOS($z)+NTOS($y)+"9")
("he"+NTOS($z)+NTOS($y)+"10") ("he"+NTOS($z)+NTOS($y)+"11") ("he"+NTOS($z)+NTOS($y)+"12")

        ENDDO
ENDDO

/??????????????????????????????????????????????????????????????????????????????????
/START OF MESHING
/??????????????????????????????????????????????????????????????????????????????????

vertex create "volv1"   coordinates 0           0 -20
vertex create "volv2"   coordinates $vollength  0 -20
edge create   "voledge" straight    "volv1" "volv2"


/??????????????????????????????????????????????????????????????????????????????????
/MODIFY FACE CORNERS
/??????????????????????????????????????????????????????????????????????????????????
```

```
/lower baffle faces corner modify

face modify "bf11" corner "hv112"
face modify ("bf1"+NTOS($half_holes_accross+1)) corner ("hv"+NTOS($half_holes_accross)+"18")

IF COND($half_holes_accross .GT. 1)
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                face modify ("bf1"+NTOS($z+1)) corner ("hv"+NTOS($z)+"18") ("hv"+NTOS($z+1)+"12")
        ENDDO
ENDIF

/upper baffle faces corner modify

face modify ("bf"+NTOS($Nholes_up+1)+"1")                corner ("hv1"+NTOS($Nholes_up)+"4")
face modify ("bf"+NTOS($Nholes_up+1)+NTOS($half_holes_accross+1)) corner
("hv"+NTOS($half_holes_accross)+NTOS($Nholes_up)+"6")

IF COND($half_holes_accross .GT. 1)
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                face modify ("bf"+NTOS($Nholes_up+1)+NTOS($z+1)) corner
("hv"+NTOS($z)+NTOS($Nholes_up)+"6")  ("hv"+NTOS($z+1)+NTOS($Nholes_up)+"4")
        ENDDO
ENDIF


/Middle baffle faces corner modify    all if

IF COND($half_holes_accross .GT. 1)


        /middle faces left
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                face modify ("bf"+NTOS($y+1)+"1") corner ("hv1"+NTOS($y)+"4") ("hv1"+NTOS($y+1)+"2")
        ENDDO

        /middle faces right
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                face modify ("bf"+NTOS($y+1)+NTOS($half_holes_accross+1)) corner
("hv"+NTOS($half_holes_accross)+NTOS($y)+"6") ("hv"+NTOS($half_holes_accross)+NTOS($y+1)+"8")
        ENDDO

        /middle faces middle
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                        face modify ("bf"+NTOS($y+1)+NTOS($z+1)) corner ("hv"+NTOS($z)+NTOS($y)+"6")
("hv"+NTOS($z+1)+NTOS($y)+"4") ("hv"+NTOS($z)+NTOS($y+1)+"8")("hv"+NTOS($z+1)+NTOS($y+1)+"2")
                ENDDO
        ENDDO
ENDIF
```

```
/???????????????????????????????????????????????????????????????????????????????
/MESH HOLES    :    Seed edges ; Mesh faces ; Create volumes : Copy volumes
/???????????????????????????????????????????????????????????????????????????????

DO PARA "$y" INIT 1 COND ($y .LE. $Nholes_up) INCR 1
        DO PARA "$z" INIT 1 COND ($z .LE. $half_holes_accross) INCR 1

                edge mesh  ("he"+NTOS($z)+NTOS($y)+"1") ("he"+NTOS($z)+NTOS($y)+"2")
("he"+NTOS($z)+NTOS($y)+"3") ("he"+NTOS($z)+NTOS($y)+"4") ("he"+NTOS($z)+NTOS($y)+"5")
("he"+NTOS($z)+NTOS($y)+"6") ("he"+NTOS($z)+NTOS($y)+"7") ("he"+NTOS($z)+NTOS($y)+"8")
("he"+NTOS($z)+NTOS($y)+"13") size $meshsize
                face mesh  ("hf"+NTOS($y)+NTOS($z)+"1") ("hf"+NTOS($y)+NTOS($z)+"2")
("hf"+NTOS($y)+NTOS($z)+"3") ("hf"+NTOS($y)+NTOS($z)+"4") submap size $meshsize
                face mesh  ("hf"+NTOS($y)+NTOS($z)+"5") map


/face mesh ("hf"+NTOS($y)+NTOS($z)+"5") map size $meshsize

                /create volumes
                volume create ("hvol"+NTOS($y)+NTOS($z)+"1") translate ("hf"+NTOS($y)+NTOS($z)+"1") onedge
"voledge"
                volume create ("hvol"+NTOS($y)+NTOS($z)+"2") translate ("hf"+NTOS($y)+NTOS($z)+"2") onedge
"voledge"
                volume create ("hvol"+NTOS($y)+NTOS($z)+"3") translate ("hf"+NTOS($y)+NTOS($z)+"3") onedge
"voledge"
                volume create ("hvol"+NTOS($y)+NTOS($z)+"4") translate ("hf"+NTOS($y)+NTOS($z)+"4") onedge
"voledge"
                volume create ("hvol"+NTOS($y)+NTOS($z)+"5") translate ("hf"+NTOS($y)+NTOS($z)+"5") onedge
"voledge"

                face cmove ("hf"+NTOS($y)+NTOS($z)+"1") ("hf"+NTOS($y)+NTOS($z)+"2")
("hf"+NTOS($y)+NTOS($z)+"3") ("hf"+NTOS($y)+NTOS($z)+"4") multiple 1 unlinkmesh offset $vollength 0 0
                face cmove ("hf"+NTOS($y)+NTOS($z)+"5") multiple 1 unlinkmesh offset $vollength 0 0

                face connect real

                volume mesh  ("hvol"+NTOS($y)+NTOS($z)+"1") cooper size $NHmeshsize
                volume mesh  ("hvol"+NTOS($y)+NTOS($z)+"2") cooper size $NHmeshsize
                volume mesh  ("hvol"+NTOS($y)+NTOS($z)+"3") cooper size $NHmeshsize
                volume mesh  ("hvol"+NTOS($y)+NTOS($z)+"4") cooper size $NHmeshsize
                volume mesh  ("hvol"+NTOS($y)+NTOS($z)+"5") cooper size $NHmeshsize

                volume cmove ("hvol"+NTOS($y)+NTOS($z)+"1") multiple $copyntimes unlinkmesh offset $vollength 0
0
                volume cmove ("hvol"+NTOS($y)+NTOS($z)+"2") multiple $copyntimes unlinkmesh offset $vollength 0
0
                volume cmove ("hvol"+NTOS($y)+NTOS($z)+"3") multiple $copyntimes unlinkmesh offset $vollength 0
0
                volume cmove ("hvol"+NTOS($y)+NTOS($z)+"4") multiple $copyntimes unlinkmesh offset $vollength 0
0
                volume cmove ("hvol"+NTOS($y)+NTOS($z)+"5") multiple $copyntimes unlinkmesh offset $vollength 0
0



        ENDDO
ENDDO

/SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
/Mesh seed the straight baffle lines
/SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

/lower baffle left
edge modify ("lhbl1") backward
edge mesh ("lhbl1") successive ratio1 $ratio1 size $Azmeshsize

edge modify ("lbl1") backward
edge mesh ("lbl1")  successive ratio1 $ratio1 size $Aymeshsize

/lower baffle right

edge mesh ("rhbl1") successive ratio1 $ratio1 size $Azmeshsize
/edge mesh ("rhbl1") size $meshsize

IF COND($half_holes_accross .GT. 1)
```

```
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                /edge mesh ("bihl1"+NTOS($z)) size $meshsize
                /edge mesh ("lbl"+NTOS($z+1)) size $meshsize
                edge mesh ("bihl1"+NTOS($z)) successive ratio1 $ratio1 ratio2 $ratio2 size $Azmeshsize
                edge modify ("lbl"+NTOS($z+1)) backward
                edge mesh ("lbl"+NTOS($z+1)) successive ratio1 $ratio1 size $Aymeshsize
        ENDDO
ENDIF


/upper baffle left
/edge mesh ("lhbl"+NTOS($Nholes_up)) size $meshsize   /do not need as the lower mesh will take care of it

//edge mesh ("ubl1") size $meshsize
edge mesh ("ubl1")  successive ratio1 $ratio1 size $Aymeshsize


/upper baffle right
/edge mesh ("rhbl"+NTOS($Nholes_up)) size $meshsize

/upper baffle middle
IF COND($half_holes_accross .GT. 1)
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                /edge mesh ("bihl"+NTOS($Nholes_up)+NTOS($z))  size $meshsize
                /edge mesh ("ubl"+NTOS($z+1))                  size $meshsize
                edge mesh ("bihl"+NTOS($Nholes_up)+NTOS($z)) successive ratio1 $ratio1 ratio2 $ratio2 size
$Azmeshsize
                /edge modify ("ubl"+NTOS($z+1)) forward
                edge mesh ("ubl"+NTOS($z+1)) successive ratio1 $ratio1 size $Aymeshsize
        ENDDO
ENDIF


/Middle faces all if

IF COND($half_holes_accross .GT. 1)

        /middle faces left
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                edge modify ("lhbl"+NTOS($y+1)   ) backward
                edge mesh ("lhbl"+NTOS($y+1)   )    successive ratio1 $ratio1 size $Azmeshsize
                edge mesh ("bivl"+NTOS($y)+"1" )    successive ratio1 $ratio1 ratio2 $ratio2 size $Aymeshsize
        ENDDO

        /middle faces right
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                edge mesh ("rhbl"+NTOS($y+1))       successive ratio1 $ratio1 size $Azmeshsize
        ENDDO

        /middle faces middle
        DO PARA "$y" INIT 1 COND ($y .LT. $Nholes_up) INCR 1
                DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                        edge mesh ("bihl"+NTOS($y+1)+NTOS($z)) successive ratio1 $ratio1 ratio2 $ratio2 size
$Azmeshsize
                        edge mesh ("bivl"+NTOS($y)+NTOS($z+1)) successive ratio1 $ratio1 ratio2 $ratio2 size
$Aymeshsize
                ENDDO
        ENDDO
ENDIF


/MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
M
/MESH BAFFLE SQUARES
/MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
M

/face mesh     ("bf"+NTOS($y)+NTOS($z)) submap size $NHmeshsize
DO PARA "$y" INIT 1 COND ($y .LE. ($Nholes_up+1)) INCR 1
        DO PARA "$z" INIT 1 COND ($z .LE. ($half_holes_accross+1)) INCR 1
                face mesh     ("bf"+NTOS($y)+NTOS($z)) submap
                volume create ("bv"+NTOS($y)+NTOS($z)) translate ("bf"+NTOS($y)+NTOS($z)) onedge "voledge"
                face cmove    ("bf"+NTOS($y)+NTOS($z)) multiple 1 unlinkmesh offset $vollength 0 0
                face connect real
                volume mesh   ("bv"+NTOS($y)+NTOS($z)) cooper size $NHmeshsize
        volume cmove  ("bv"+NTOS($y)+NTOS($z)) multiple $copyntimes unlinkmesh offset $vollength 0 0


        ENDDO
ENDDO


/MMMMMMMMMMMMMMMMMMMMMMMMMMMMMGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
/MESH THE GAPS
/MMMMMMMMMMMMMMMMMMMMMMMMMMMMMGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG

/MESH THE LOWER GAP LEFT

        edge mesh      "l5s1" size $NHmeshsize
        face mesh      "lgfleft" map size $NHmeshsize
        volume create  "lgvl" translate "lgfleft" onedge "voledge"
        face cmove     "lgfleft" multiple 1 unlinkmesh offset $vollength 0 0
        face connect real
        volume mesh    "lgvl" map size $NHmeshsize
        volume cmove   "lgvl" multiple $copyntimes unlinkmesh offset $vollength 0 0


/MESH MIDDLE VOLS

IF COND ($half_holes_accross .GT. 1)
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                face mesh     ("lgf"+NTOS($z)) map size $NHmeshsize
                volume create ("lgv"+NTOS($z)) translate ("lgf"+NTOS($z)) onedge "voledge"
                face cmove    ("lgf"+NTOS($z)) multiple 1 unlinkmesh offset $vollength 0 0
                face connect real
                volume mesh   ("lgv"+NTOS($z)) map size $NHmeshsize
                volume cmove  ("lgv"+NTOS($z)) multiple $copyntimes unlinkmesh offset $vollength 0 0


        ENDDO
ENDIF

/MESH THE LOWER GAP LEFT
        face mesh      "lgfright" map size $NHmeshsize
        volume create  "lgvr" translate "lgfright" onedge "voledge"
        face cmove     "lgfright" multiple 1 unlinkmesh offset $vollength 0 0
        face connect real
        volume mesh    "lgvr" map size $NHmeshsize
        volume cmove   "lgvr" multiple $copyntimes unlinkmesh offset $vollength 0 0



/MMMMMMMMMMMMMMMMMMMM GGGGGGGGGGGGGGGGGGG
/MESH UPPER GAP
/MMMMMMMMMMMMMMMMMMMM GGGGGGGGGGGGGGGGGGG

/MESH THE UPPER GAP LEFT
        edge mesh      ("l5s"+NTOS($Nholes_up+3)) size $NHmeshsize
        face mesh      "ugfleft" map size $NHmeshsize
        volume create  "ugvl" translate "ugfleft" onedge "voledge"
        face cmove     "ugfleft" multiple 1 unlinkmesh offset $vollength 0 0
        face connect real
        volume mesh    "ugvl" map size $NHmeshsize
        volume cmove   "ugvl" multiple $copyntimes unlinkmesh offset $vollength 0 0


/MESH MIDDLE VOLS

IF COND ($half_holes_accross .GT. 1)
        DO PARA "$z" INIT 1 COND ($z .LT. $half_holes_accross) INCR 1
                face mesh     ("ugf"+NTOS($z)) map size $NHmeshsize
                volume create ("ugv"+NTOS($z)) translate ("ugf"+NTOS($z)) onedge "voledge"
                face cmove    ("ugf"+NTOS($z)) multiple 1 unlinkmesh offset $vollength 0 0
                face connect real
                volume mesh   ("ugv"+NTOS($z)) map size $NHmeshsize
                volume cmove  ("ugv"+NTOS($z)) multiple $copyntimes unlinkmesh offset $vollength 0 0

        ENDDO
ENDIF
```

```
/MESH THE UPPER GAP RIGHT
        face mesh     "ugfright" map size $NHmeshsize
        volume create "ugvr" translate "ugfright" onedge "voledge"
        face cmove    "ugfright" multiple 1 unlinkmesh offset $vollength 0 0
        face connect real
        volume mesh   "ugvr" map size $NHmeshsize
        volume cmove  "ugvr" multiple $copyntimes unlinkmesh offset $vollength 0 0

/vertex connect real
/edge connect real
face connect real


$lastvolid=lastid(t_vo)
$counterv=1

DO PARA "$counterv" INIT 1 COND ($counterv .LE. $lastvolid) INCR 1
        volume mesh ("volume."+NTOS($counterv)) submap size $NHmeshsize
ENDDO




$lastvolid=lastid(t_vo)
$counterv=1

DO PARA "$counterv" INIT 1 COND ($counterv .LE. $lastvolid) INCR 1
        volume mesh ("volume."+NTOS($counterv)) submap size $meshsize
ENDDO

/BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
/Set Boundary Conditions
/BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

/loop for setting the boundary conditions

declare $facecoords [1:3]

$lastfaceid=lastid(t_fa)
$counter=1
$xmin=10
$xmax=($tank_length-10)
/$ymin=($bottom_gap_height)
/$ymax=($tank_height-$top_gap_height)
$ymin=0
$ymax=0

/check to see if you need to create groups before you can add info to them
vertex create "phantom1" coordinates 0 0 -20
vertex create "phantom2" coordinates 0 5 -20
vertex create "phantom3" coordinates 0 5 -25
vertex create "phantom4" coordinates 0 0 -25
edge create "phe1" straight "phantom1" "phantom2"
edge create "phe2" straight "phantom2" "phantom3"
edge create "phe3" straight "phantom3" "phantom4"
edge create "phe4" straight "phantom4" "phantom1"
face create "phantomface1" wireframe "phe1" "phe2" "phe3" "phe4"

vertex create "phantom5" coordinates 0 0 -40
vertex create "phantom6" coordinates 0 5 -40
vertex create "phantom7" coordinates 0 5 -45
vertex create "phantom8" coordinates 0 0 -45
edge create "phe5" straight "phantom5" "phantom6"
edge create "phe6" straight "phantom6" "phantom7"
edge create "phe7" straight "phantom7" "phantom8"
edge create "phe8" straight "phantom8" "phantom5"
face create "phantomface2" wireframe "phe5" "phe6" "phe7" "phe8"

group create "wallsym"  face "phantomface1"
group create "intwalls" face "phantomface2"

$Shift=2
$z=0
$y=0
```

```
DO PARA "$counter" INIT 1 COND ($counter .LE. $lastfaceid) INCR 1
        $facecoords=ENT2LOC("face."+NTOS($counter))

        IF COND ($facecoords[3] .EQ. $tank_half_width)
                group add "wallsym" face ("face."+NTOS($counter))
        ENDIF

        DO PARA "$z" INIT 1 COND ($z .LE. ($half_holes_accross+1)) INCR 1

                DO PARA "$y" INIT 1 COND ($y .LE. ($Nholes_up+1)) INCR 1

                        IF COND ($z .EQ. 1)
                                $zmin=0
                                $zmax=($("zhole1"+NTOS($z))-$Shift)
                        ENDIF

                        IF COND (($z .GT. 1) .AND. ($z .LE. ($half_holes_accross)))
                                $zmin=($("zhole1"+NTOS($z-1))+$Shift)
                                $zmax=($("zhole1"+NTOS($z))-$Shift)
                        ENDIF

                        IF COND ( $z .EQ. ($half_holes_accross+1) )
                                $zmin=($("zhole1"+NTOS($z-1))+$Shift)
                                $zmax=$tank_half_width
                        ENDIF



                        IF COND ($y .EQ. 1)
                                $ymin=($bottom_gap_height+5)
                                $ymax=($("yhole"+NTOS($y)+"1")-$R)
                        ENDIF

                        IF COND (($y .GT. 1) .AND. ($y .LT. ($Nholes_up+1)))
                                $ymin=($("yhole"+NTOS($y-1)+"1")+$R)
                                $ymax=($("yhole"+NTOS($y)+"1")-$R)
                        ENDIF

                        IF COND ($y .EQ. ($Nholes_up+1))
                                $ymin=($("yhole"+NTOS($y-1)+"1")+$R)
                                $ymax=($tank_height-$top_gap_height-5)

                        ENDIF



                        IF COND ( ($facecoords[1] .GT. $xmin) .AND. ($facecoords[1] .LT. $xmax) .AND.
($facecoords[2] .GT. $ymin) .AND. ($facecoords[2] .LT. $ymax) .AND. ($facecoords[3] .GT. $zmin) .AND.
($facecoords[3] .LT. $zmax))
                                group add "intwalls" face ("face."+NTOS($counter))

                        ENDIF
                ENDDO
        ENDDO

ENDDO

edge delete "voledge" lowertopology
face delete "phantomface1" lowertopology
face delete "phantomface2" lowertopology

/check to see if this is valid
physics create "symwallplane"  btype "SYMMETRY" group "wallsym"
physics create "internalwalls" btype "WALL"     group "intwalls"

default set "FILE_IO.FLUENT5.EXPORT_USING_UTILITY" numeric 1
/export fluent5 "/home/rolf/gm1.msh"
export fluent5 "lsmesh.msh"


//EOF
```

APPENDIX H: 3D baffled tank Fluent journal file

```
!3D sloshing Fluent setup journal
f/rc lsmesh.msh
g/s 0.001 0.001 0.001
!echo grid/check
!echo display/grid-outline
!echo display/view dv
d/v read-v
"viewr.vw"
de/m u1o y
de/m/solver seg y
de/m multi vof 2 geo-reconstruct 0.25 no yes
de/m/v lam y
de/mat/cc air water y constant 998 n n y constant 0.001 n n n n n n
de/ph pd phase-2 y water
de/m multi vof 2 geo-reconstruct 0.25 no yes
de/ud cf libudf
de/bc/fl
fluid
mixture
y
y
"udf"
"xmom_source"
n
0
n
0
n
y
0
0
0
0
0
1
n
de/ud/fh
"none"
"Accl"
"none"
"none"
"none"
"none"
!echo de/mat/cc air water y constant 998 n n y constant 0.001 n n n n n n
!echo de/ph pd phase-2 y water
de/oc grav y 0 -9.81 0
so/se/dis-s p 13
so/se/dis-s f 22
so/se/dis-s m 0
so/se/ur p 0.6
so/se/ur d 1
so/se/ur bf 1
so/se/ur m 0.8
so/se/ur mp 1
so/in/cd az
so/in if
so/se ts 0.001
(rpsetvar 'piso/skew? #f)
a mih y n 0 0.5 0 0.2 0 0.2
(cx-gui-do cx-activate-item "MenuBar*InitializeSubMenu*Patch...")
(cx-gui-do cx-set-list-selections "Patch*Frame1*Frame2*List2(Variable)" '( 4))
(cx-gui-do cx-activate-item "Patch*Frame1*Frame2*List2(Variable)")
(cx-gui-do cx-set-real-entry-list "Patch*Frame2*RealEntry1(Value)" '( 1))
(cx-gui-do cx-set-list-selections "Patch*Frame3*Frame2*List2(Registers To Patch)" '( 0))
(cx-gui-do cx-activate-item "Patch*Frame3*Frame2*List2(Registers To Patch)")
(cx-gui-do cx-activate-item "Patch*PanelButtons*PushButton1(Patch)")
(cx-gui-do cx-activate-item "Patch*PanelButtons*PushButton1(Close)")
so/mo/re/pl y
!echo de/mat/ copy fluid water-liquid
!echo de/ph pd phase-2 y water-liquid
```

```
!echo (rpsetvar 'monitor/commands '((command-1 10 #t "su/is/vof-phase-2 free-surf () 0.5 ()") (command-2 10 #t "f/e/as data_%t
free-surf () n y yc q y") (command-3 10 #t "su/ds/free-surf")))
(rpsetvar 'monitor/commands '((command-1 20 #t "su/is/vof-phase-2 free-surf () 0.5 ()") (command-2 20 #t "f/e/as data_%t free-
surf () n y yc q y") (command-3 20 #t "su/ds/free-surf")))
!echo so/ dti 4 20
f/wc lsmesh.cas.gz

f/as/ rn lsmesh.gz
f/as/ df 500
f/as/ cf 500
so dti 7000 20
```

APPENDIX I: 3D data extraction source code

```
// Create by Rolf Deitrich
// 3D Data extraction source code

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define NO_OF_FILES 349         //350 inc of 2 starting at 2; 349 inc of 2 starting at 4; 700 inc of 1 starting at 1
#define END_OF_LOOP 700
#define LINE_BUFFER_SIZE 100

int main()

{
   int i;
   char line[LINE_BUFFER_SIZE] = "";
   char filename[80] = "";
   char *basename = "data_\0";
   char *postfix  = "\0";
   char tempstr[5];
            char ootempstr[80];

   FILE *fh;

            char line1[255];
            FILE* fh1;      //dev file

            char line2[255];
            FILE* fh2;      //max file

            char line3[255];
            FILE* fh3;      //funny val file file

            char linenoiseup[255];
            FILE* fhnup;                    //max height of noise value

            char linenoisedown[255];
            FILE* fhndown;                  //max height of noise value

            char linenormalplat[255];
            FILE* fhnormal;

            double value[5];
             int num = 0;

            double dMultifilesquaredev=0;
            double dLevel=0.2;
            int       iMultifilecounter=0;
            double dMultifilesingledev=0;
            double dMaxmultidev=0;
            double dSinglefiledev=0;
            double dFinaldev;
            char *filename2="multilev.txt";
            char *filename3="maxdev.txt";
            char *filename4="funnyval.txt";

            char *filenamenoiseup="noisemaxup.txt";
            char *filenamenoisedown="noisedown.txt";
            double dGetnoiseup=0;
            double dGetnoisedown=400;

            char *filenamenormalizedpeak="normalizedpeak.txt";
            double dNormalplat=0;
            double dOvershootpercent;

            fh1 = fopen( filename2, "w" );
            fclose( fh1 );

            fhnup = fopen( filenamenoiseup, "w" );
            fclose( fhnup );

            fhndown = fopen( filenamenoisedown, "w" );
            fclose( fhndown );


            fhnormal = fopen ( filenamenormalizedpeak, "w" );
            fclose( fhnormal );

for( i = 4; i <= END_OF_LOOP; i+=2 )
{
            if (i > END_OF_LOOP)
            printf("problems!!!");

   strcpy( filename, basename );

            strcpy(ootempstr, "");

            if ( i < 100 )
            {
            strcpy(ootempstr, "0");
            }
            if ( i < 10 )
            {
            strcpy(ootempstr, "00");
            }
            if ( i < 1 )
            {
            strcpy(ootempstr, "000");
            }

            strcat( filename, ootempstr );
    sprintf( tempstr, "%i" , i*10 );
            strcat( filename, tempstr );
    strcat( filename, postfix );
    //printf("filename = %s\n", filename);

    fh = fopen( filename, "r" );

     if( fh == NULL )
    {
      printf("Error opening file to be read in \n");
            exit(0);
   }
   else
   {
   fgets( line, LINE_BUFFER_SIZE, fh );    // Get rid of first line

     fgets( line, LINE_BUFFER_SIZE, fh );

            while ( ( line != NULL ) && (strlen(line) > 5) )
      {
          iMultifilecounter++;
          num = atoi( strtok( line, " \t\r\n," ) );
          value[0] = atof( strtok( NULL, " \t\r\n,") );
          value[1] = atof( strtok( NULL, " \t\r\n,") );
          value[2] = atof( strtok( NULL, " \t\r\n,") );
          value[3] = atof( strtok( NULL, " \t\r\n,") );
               //value[4] = atof( strtok( NULL, " \t\r\n,") );

          // Calculations
                                                    dMultifilesquaredev=dMultifilesquaredev+pow((value[3]-dLevel),2);

            if( dGetnoiseup <= value[3] )
            {
            dGetnoiseup = value[3];
            }
            if( dGetnoisedown >= value[3] )
            {
            dGetnoisedown = value[3];
            }
            if( i == 400 )
            {
            dNormalplat = value[3];
            }
// End of calculations

                fgets( line, LINE_BUFFER_SIZE, fh );
            }//end while loop reading file


            //printf( "EOF, num = %i\n", num );
        }//end if else for multifilw


                dMultifilesingledev=(dMultifilesquaredev)/iMultifilecounter;
                if( dMaxmultidev <= dMultifilesingledev )
                dMaxmultidev = dMultifilesingledev;
                }
                //begin write level dev to file
                fh1 = fopen( filename2, "a" );
        if( fh1 == NULL )
        {
        printf("Error opening file\n");
        exit(0);
        }
        sprintf( line1, "%f\n",dMultifilesingledev );
                fputs( line1, fh1 );
                fclose( fh1 );                                          //end write level dev to file
                //begin writing to noise file up
                fhnup = fopen( filenamenoiseup, "a" );
                sprintf( linenoiseup, "%f\n", dGetnoiseup );
                fputs( linenoiseup,fhnup   );
                fclose( fhnup );
                //end writing to noise file up

//begin writing to noise file down
fhndown = fopen( filenamenoisedown, "a" );
sprintf( linenoisedown, "%f\n", dGetnoisedown );
fputs( linenoisedown,fhndown );
fclose( fhndown );
//end writing to noise file down

//reset loop var
dGetnoiseup=0;
dGetnoisedown=400;
//end reset loop var

                        dSinglefiledev=dSinglefiledev+dMultifilesingledev;
                fclose(fh);  //close multifile
    }// end of for loop going throught the files

                //Funny val start

                dFinaldev=dSinglefiledev/NO_OF_FILES;
fh3 = fopen( filename4, "w" );
                if( fh3 == NULL )
                {
                printf("Error opening file\n");
                exit(0);
                }
                sprintf( line3, "%f", dFinaldev );
                fputs( line3, fh3 );
                fclose( fh3 );
                //printf( "dFinaldev = %d\n", dFinaldev );

                //Funny val end

                //begin write max dev to file

                fh2 = fopen( filename3, "w" );
                if( fh2 == NULL )
                {
                printf("Error opening file\n");
                exit(0);
                }
                sprintf( line2, "%f" , dMaxmultidev );
                printf( line2,dMaxmultidev );
                fputs( line2, fh2 );
                fclose( fh2 );
                //end write max dev to file

                //begin write normalized overshoot to file
                dOvershootpercent=((dMaxmultidev-dNormalplat)/(dNormalplat))*100;

                fhnormal = fopen ( filenamenormalizedpeak, "w" );
                if( fhnormal == NULL )
                {
                printf("Error opening file\n");
                exit(0);
                }
                sprintf( linenormalplat, "%f" , dOvershootpercent );
                fputs( linenormalplat , fhnormal);d
                fclose( fhnormal );

                //end writing normalized overshoot to file


}// end of main
```

APPENDIX J: Fluent UDF momentum source input file

```
#include "udf.h"
#define time 2
#define velocity 30

double AlnrX;

DEFINE_ADJUST(Accl, domain)
{
  double t;

  t = RP_Get_Real("flow-time");
  /*t=0.2;*/

  if (t<=time) {
    AlnrX = velocity/(3.6*time);
  }
  else {
    AlnrX = 0.0;
  }
  printf("time = %f\n",t);
  printf("accel = %f\n",AlnrX);

}


DEFINE_SOURCE(xmom_source, cell, thread, dS, eqn)
{
  double pos[ND_ND];
  double rho;
  double source;

  C_CENTROID(pos,cell,thread);

  rho  = C_R(cell,thread);

  dS[eqn] = 0.0;
  source = AlnrX *rho;

  return source;
}
```

APPENDIX K: 2D TDV extraction source code

```
// fluentc.c edited by Thomas Kingsley on 5/11/2002 for
// adaptation to 2-D sloshing and TDV

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define NO_OF_FILES 125         // Number of data files to proccess
#define END_OF_LOOP 500                      // Final data file name
#define LINE_BUFFER_SIZE 100

int main()

{
    int i;
    char line[LINE_BUFFER_SIZE] = "";
    char filename[80] = "";
    char *basename = "data_\0";
    char *postfix  = "\0";
    char tempstr[5];
        char ootempstr[80];

    FILE *fh;

        char line1[255];
        FILE* fh1;      //dev file

        char line2[255];
        FILE* fh2;      //max file

        char line3[255];
        FILE* fh3;      //funny val file file

        char linenoiseup[255];
        FILE* fhnup;           //max height of noise value

        char linenoisedown[255];
        FILE* fhndown;         //max height of noise value

        char linenormalplat[255];
        FILE* fhnormal;

        double value[5];
        int num = 0;

        double dMultifilesquaredev=0;
        double dLevel=0.28;
        int        iMultifilecounter=0;
        double dMultifilesingledev=0;
        double dMaxmultidev=0;
        double dSinglefiledev=0;
        double dFinaldev=0;
        char *filename2="multilev.txt";
        char *filename3="maxdev.txt";
        char *filename4="TotalDev.txt";

        char *filenamenoiseup   = "noisemaxup.txt";
        char *filenamenoisedown = "noisedown.txt" ;
        double dGetnoiseup=0;
        double dGetnoisedown=400;

        char *filenamenormalizedpeak="normwave.txt";
        double dNormalplat=0;
        double dOvershootpercent;

char temp1[255];
char temp2[255];

        fh1 = fopen( filename2, "w" );
        fclose( fh1 );
/*
        fhnup = fopen( filenamenoiseup, "w" );
        fclose( fhnup );

        fhndown = fopen( filenamenoisedown, "w" );
        fclose( fhndown );
*/

        fhnormal = fopen ( filenamenormalizedpeak, "w" );
        fclose( fhnormal );

    for( i = 4; i <= END_OF_LOOP; i+=4 )
    {
                if (i > END_OF_LOOP)
                        printf("problems!!!");

        strcpy( filename, basename );

                strcpy(ootempstr, "");

                if ( i < 100 )
                {
                        strcpy(ootempstr, "0");
                }
                if ( i < 10 )
                {
                        strcpy(ootempstr, "00");
                }
                if ( i < 1 )
                {
                        strcpy(ootempstr, "000");
                }

                strcat(  filename, ootempstr );
        sprintf( tempstr , "%i" , i*10 );
                strcat(  filename, tempstr );
        strcat(  filename, postfix );

        fh = fopen( filename, "r" );

        if( fh == NULL )
        {
            printf("Error opening file to be read in \n");
                        exit(0);
        }
        else
        {
            fgets( line, LINE_BUFFER_SIZE, fh );    // Get rid of first line

            fgets( line, LINE_BUFFER_SIZE, fh );

                        while ( ( line != NULL ) && (strlen(line) > 5) )
            {
                iMultifilecounter++;
                              num = atoi( strtok( line, " \t\r\n," ) );
                value[0] = 0;//atof( strtok( NULL, " \t\r\n,") );
                value[1] = atof( strtok( NULL, " \t\r\n,") );
                value[2] = atof( strtok( NULL, " \t\r\n,") );
                value[3] = atof( strtok( NULL, " \t\r\n,") );
                              //value[4] = atof( strtok( NULL, " \t\r\n,") );

                // Calculations

        //dMultifilesquaredev=dMultifilesquaredev+pow((value[3]-dLevel),2);

        dMultifilesquaredev=dMultifilesquaredev+pow(((value[3]-dLevel)*(value[3]-
dLevel)),0.5);

                                          if( dGetnoiseup <= value[3] )
                                          {
                                                  dGetnoiseup = value[3];
                                          }
                                          if( dGetnoisedown >= value[3] )
                                          {
                                                  dGetnoisedown = value[3];
                                          }
```

```
                         // End of calculations

              fgets( line, LINE_BUFFER_SIZE, fh );
          }//end while loop reading file


          //printf( "EOF, num = %i\n", num );
      }//end if else for multifilw



             dMultifilesingledev=(dMultifilesquaredev)/iMultifilecounter;

             if( i == 400 )
             {
                    dNormalplat = dMultifilesingledev ;
             }

             if( dMaxmultidev <= dMultifilesingledev )
             {
                    dMaxmultidev = dMultifilesingledev;
             }

             //begin write level dev to file

             fh1 = fopen( filename2, "a" );
      if( fh1 == NULL )
      {
          printf("Error opening file\n");
                    exit(0);
      }
             sprintf( line1, "%f\n",dMultifilesingledev );
             fputs( line1, fh1 );
             fclose( fh1 );
             //end write level dev to file

             /*//begin writing to noise file up
             fhnup = fopen( filenamenoiseup, "a" );
             sprintf( linenoiseup, "%f\n", dGetnoiseup );
             fputs( linenoiseup,fhnup   );
             fclose( fhnup );
             //end writing to noise file up

             //begin writing to noise file down
             fhndown = fopen( filenamenoisedown, "a" );
             sprintf( linenoisedown, "%f\n", dGetnoisedown );
             fputs( linenoisedown,fhndown );
             fclose( fhndown );
             //end writing to noise file down*/

             //reset loop var
             dGetnoiseup=0;
             dGetnoisedown=400;
             //end reset loop var

             dSinglefiledev=dSinglefiledev+dMultifilesingledev;
      fclose(fh);  //close multifile
  }// end of for loop going through the files

      // Final deveation start

      dFinaldev=dSinglefiledev/NO_OF_FILES;

      fh3 = fopen( filename4, "w" );
  if( fh3 == NULL )
  {
             printf("Error opening file\n");
             exit(0);
      }
      sprintf( line3, "%f", dFinaldev );
      fputs( line3, fh3 );
      fclose( fh3 );
      //printf( "dFinaldev = %d\n", dFinaldev );

      // Final deveation end
```

```
        //begin write max dev to file

        fh2 = fopen( filename3, "w" );
        if( fh2 == NULL )
        {
                printf("Error opening file\n");
                exit(0);
        }
        sprintf( line2, "%f" , dMaxmultidev );
//      printf( line2,dMaxmultidev );
        fputs( line2, fh2 );
        fclose( fh2 );

        //end write max dev to file

        //begin writing percent overshoot of the squared data to file
        dOvershootpercent=((dMaxmultidev-dNormalplat)/(dNormalplat))*100;

        fhnormal = fopen ( filenamenormalizedpeak, "a" );
        if( fhnormal == NULL )
        {
                printf("Error opening file\n");
                exit(0);
        }

//      tempory checks
//      sprintf( temp1, "%f\n" , dNormalplat );
//      printf( temp1, dNormalplat );
//      sprintf( temp2, "%f\n" , dOvershootpercent );
//      printf( temp2, dOvershootpercent);
//
        sprintf( linenormalplat, "%f" , dOvershootpercent );
        fputs( linenormalplat , fhnormal);
        fclose( fhnormal );

        //end writing percent overshoot of the squared data to file

}// end of main
```

APPENDIX L: LS-OPT command file – Linear RSM optimisation (Design 1)

```
"2D sloshing optimisation for total deviation value (TDV) with variables of baffle
height, hole size, and baffle centroid location"
Author "Thomas Kingsley, Ken Craig"
$ Created on Mon Sep  1 16:30:29 2003
solvers 1
responses 4
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 3
 Variable 'centroid' 100
  Lower bound variable 'centroid' 10
  Upper bound variable 'centroid' 190
 Variable 'b_height' 100
  Lower bound variable 'b_height' 20
  Upper bound variable 'b_height' 180
 Variable 'hole' 50
  Lower bound variable 'hole' 10
  Upper bound variable 'hole' 140

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "/home/thomas/2dslosh/LSOPT/fluent_script"
  solver input file "/home/thomas/2dslosh/LSOPT/fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "/home/thomas/2dslosh/LSOPT/2D_70p_gambit.jou"
  solver order linear
  solver experiment design dopt
  solver number experiments 7
  solver basis experiment 3toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'too_high_baf' expression {(190-0.5*b_height)-centroid}
 response 'too_low_baf' expression {(10+0.5*b_height)-centroid}
 response 'too_big_hole' expression {(0.8*b_height)-hole}

$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'TDV' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 3
  move
 constraint 'too_high_baf'
  lower bound constraint 'too_high_baf' 0
 constraint 'too_low_baf'
  upper bound constraint 'too_low_baf' 0
 constraint 'too_big_hole'
  lower bound constraint 'too_big_hole' 0
$
$ JOB INFO
$
 iterate param design 0.001
 iterate param objective 0.001
 iterate param stoppingtype and
 iterate 10
STOP
```

APPENDIX M: LSOPT command file – Quadratic RSM optimisation (Design 1)

```
"2D sloshing optimisation for total deviation value (TDV) with variables of baffle
height, hole size, and baffle centroid location"
Author "Thomas Kingsley, Ken Craig"
$ Created on Mon Sep 15 12:00:18 2003
solvers 1
responses 5
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 3
 Variable 'centroid' 100
  Lower bound variable 'centroid' 10
  Upper bound variable 'centroid' 190
 Variable 'b_height' 100
  Lower bound variable 'b_height' 20
  Upper bound variable 'b_height' 180
 Variable 'hole' 50
  Lower bound variable 'hole' 10
  Upper bound variable 'hole' 140

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "/home/thomas/2dslosh/LSOPT/fluent_script"
  solver input file "/home/thomas/2dslosh/LSOPT/fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "/home/thomas/2dslosh/LSOPT/2D_70p_gambit.jou"
  solver order quadratic
  solver experiment design dopt
  solver number experiments 16
  solver basis experiment 5toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'too_high_baf' expression {(190-0.5*b_height)-centroid}
 response 'too_low_baf' expression {(10+0.5*b_height)-centroid}
 response 'too_big_hole' expression {(0.8*b_height)-hole}
 response 'baffle_length' expression {4*(b_height-hole)}

$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'TDV' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 4
  move
 constraint 'too_high_baf'
```

```
  lower bound constraint 'too_high_baf' 0
 constraint 'too_low_baf'
  upper bound constraint 'too_low_baf' 0
 constraint 'too_big_hole'
  lower bound constraint 'too_big_hole' 0
  stay
 constraint 'baffle_length'
  lower bound constraint 'baffle_length' 0
  upper bound constraint 'baffle_length' 800
$
$ JOB INFO
$
 iterate param design 0.001
 iterate param objective 0.001
 iterate param stoppingtype and
 iterate 8
STOP
```

APPENDIX N: LSOPT command file – Neural Network optimisation (Design 1)

```
"2D sloshing optimisation for total deviation value (TDV) with variables of baffle
height, hole size, and baffle centroid location"
Author "Thomas Kingsley, Ken Craig"
$ Created on Mon Sep  8 10:05:28 2003
solvers 1
responses 4
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 3
 Variable 'centroid' 100
  Lower bound variable 'centroid' 10
  Upper bound variable 'centroid' 190
 Variable 'b_height' 100
  Lower bound variable 'b_height' 20
  Upper bound variable 'b_height' 180
 Variable 'hole' 50
  Lower bound variable 'hole' 10
  Upper bound variable 'hole' 140

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "/home/thomas/2dslosh/LSOPT/fluent_script"
  solver input file "/home/thomas/2dslosh/LSOPT/fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "/home/thomas/2dslosh/LSOPT/2D_70p_gambit.jou"
  solver order FF
  solver update doe
  solver experiment design latin_hypercube
  solver number experiments 129
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'too_high_baf' expression {(190-0.5*b_height)-centroid}
 response 'too_low_baf' expression {(10+0.5*b_height)-centroid}
 response 'too_big_hole' expression {(0.8*b_height)-hole}

$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'TDV' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 3
  move
 constraint 'too_high_baf'
```

```
  strict
  lower bound constraint 'too_high_baf' 0
  slack
 constraint 'too_low_baf'
  strict
  upper bound constraint 'too_low_baf' 0
 constraint 'too_big_hole'
  lower bound constraint 'too_big_hole' 0
  slack
$
$ JOB INFO
$
 iterate param design 0.001
 iterate param objective 0.001
 iterate param stoppingtype and
 iterate 1
STOP
```

APPENDIX O: LSOPT command file – Quadratic RSM optimisation (Design 2)

```
"2D sloshing optimisation for total deviation value (TDV) with 4 variables and
Variable Fill"
Author "Thomas Kingsley, Ken Craig"
$ Created on Wed Sep 17 11:02:55 2003
solvers 1
responses 6
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 4
 Variable 'mid_baf_centroid' 100
  Lower bound variable 'mid_baf_centroid' 15
  Upper bound variable 'mid_baf_centroid' 185
 Variable 'side_baf_centroid' 100
  Lower bound variable 'side_baf_centroid' 15
  Upper bound variable 'side_baf_centroid' 185
 Variable 'mid_baf_height' 100
  Lower bound variable 'mid_baf_height' 10
  Upper bound variable 'mid_baf_height' 180
 Variable 'side_baf_width' 100
  Lower bound variable 'side_baf_width' 10
  Upper bound variable 'side_baf_width' 180
$
$ CONSTANTS
$
constants 1
 Constant 'fill_level' 0.14

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "../../fluent_script"
  solver input file "fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "2D_VF_gambit.jou"
  solver order quadratic
  solver experiment design dopt
  solver number experiments 23
  solver basis experiment 5toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'SB_side_constraint' expression {side_baf_centroid-side_baf_width/2}
 response 'SB_mid_constaint' expression {side_baf_centroid+side_baf_width/2}
 response 'MB_up_constaint' expression {mid_baf_centroid+mid_baf_height/2}
 response 'MB_low_constraint' expression {mid_baf_centroid-mid_baf_height/2}
 response 'baffle_length' expression {mid_baf_height+2*side_baf_width}

$
```

```
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'TDV' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 5
  move
 constraint 'SB_side_constraint'
  strict
  lower bound constraint 'SB_side_constraint' 10
  slack
 constraint 'SB_mid_constaint'
  strict
  upper bound constraint 'SB_mid_constaint' 190
 constraint 'MB_up_constaint'
  slack
  strict
  upper bound constraint 'MB_up_constaint' 190
 constraint 'MB_low_constraint'
  lower bound constraint 'MB_low_constraint' 10
  slack
  stay
 constraint 'baffle_length'
  lower bound constraint 'baffle_length' 0
  upper bound constraint 'baffle_length' 540
$
$ JOB INFO
$
 iterate param design 0.001
 iterate param objective 0.001
 iterate param stoppingtype and
 iterate 8
STOP
```

APPENDIX P: LSOPT command file – Quadratic RSM optimisation (Design 2B)

```
"2dslosh opt (full size) (design 2B"
Author "Thomas Kingsley, Ken Craig"
$ Created on Tue Dec  9 12:27:54 2003
solvers 1
responses 6
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 4
 Variable 'mid_baf_centroid' 100
  Lower bound variable 'mid_baf_centroid' 60
  Upper bound variable 'mid_baf_centroid' 320
 Variable 'side_baf_centroid' 100
  Lower bound variable 'side_baf_centroid' 15
  Upper bound variable 'side_baf_centroid' 185
 Variable 'mid_baf_height' 100
  Lower bound variable 'mid_baf_height' 40
  Upper bound variable 'mid_baf_height' 320
 Variable 'side_baf_width' 100
  Lower bound variable 'side_baf_width' 10
  Upper bound variable 'side_baf_width' 180
$
$ CONSTANTS
$
constants 1
 Constant 'fill_level' 0.28

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "../../fluent_script"
  solver input file "fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "2D_tom_gambit.jou"
  solver order linear
  solver experiment design dopt
  solver number experiments 8
  solver basis experiment 3toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'SB_side_constraint' expression {side_baf_centroid-side_baf_width/2}
 response 'SB_mid_constaint' expression {side_baf_centroid+side_baf_width/2}
 response 'MB_up_constaint' expression {mid_baf_centroid+mid_baf_height/2}
 response 'MB_low_constaint' expression {mid_baf_centroid-mid_baf_height/2}
 response 'baffle_length' expression {mid_baf_height+2*side_baf_width}

$
$ OBJECTIVE FUNCTIONS
$
```

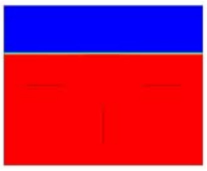
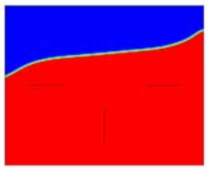
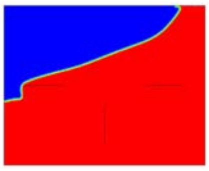
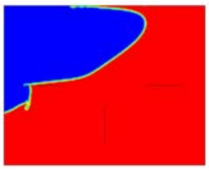
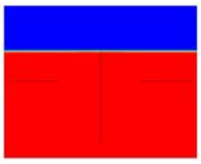
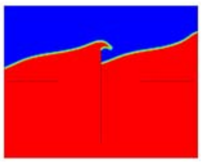
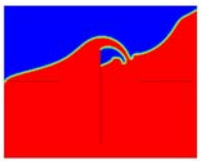
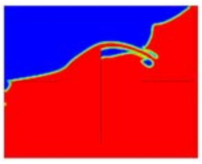
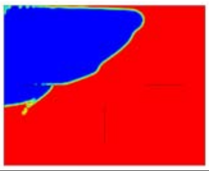
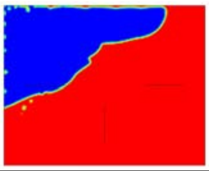
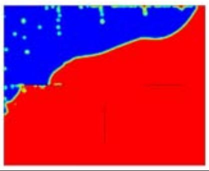
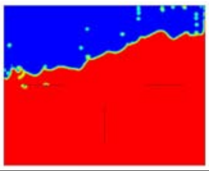
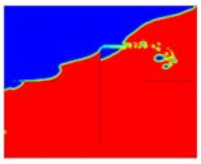
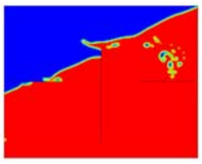
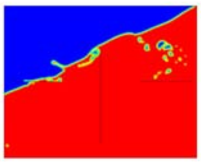
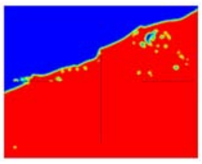
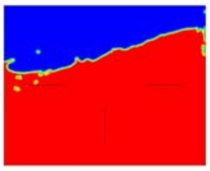
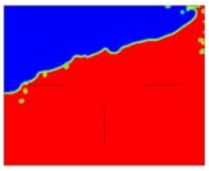
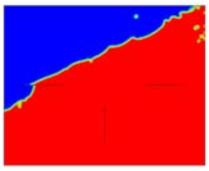
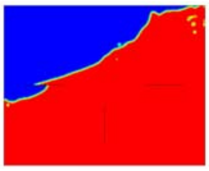
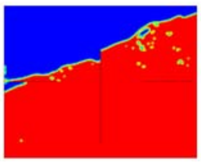
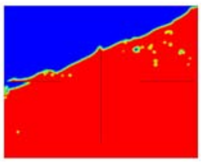
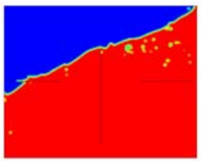
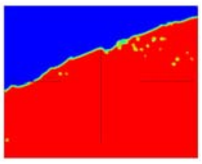
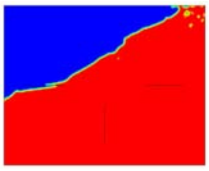
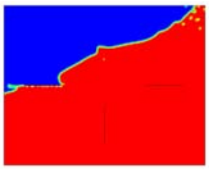
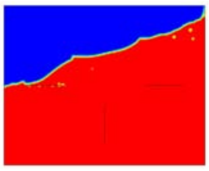
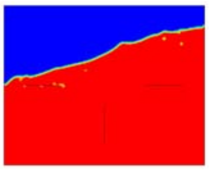
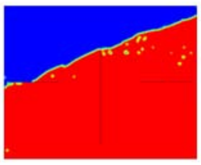
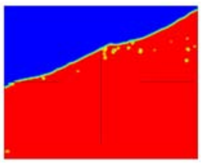
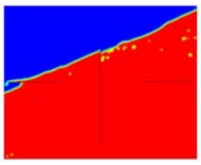
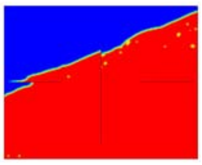
```
 objectives 1
 objective 'TDV' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 5
  move
 constraint 'SB_side_constraint'
  strict
  lower bound constraint 'SB_side_constraint' 10
  slack
 constraint 'SB_mid_constaint'
  strict
  upper bound constraint 'SB_mid_constaint' 190
 constraint 'MB_up_constaint'
  slack
  strict
  upper bound constraint 'MB_up_constaint' 340
 constraint 'MB_low_constraint'
  lower bound constraint 'MB_low_constraint' 40
  slack
  stay
 constraint 'baffle_length'
  lower bound constraint 'baffle_length' 0
  upper bound constraint 'baffle_length' 540
$
$ JOB INFO
$
 iterate param design 0.001
 iterate param objective 0.001
 iterate param stoppingtype and
 iterate 6
STOP
```

APPENDIX Q: LSOPT command file – Quadratic RSM saddle-point optimisation (Design 2)

```
"2D sloshing optimisation for total deviation value (TDV) with 4 variables and
Variable Fill"
Author "Thomas Kingsley, Ken Craig"
$ Created on Wed Oct  8 12:55:19 2003
solvers 1
responses 7
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 5
 Variable 'fill_level' 0.1
  Lower bound variable 'fill_level' 0.01
  Upper bound variable 'fill_level' 0.19
  Variable 'fill_level' max
 Variable 'mid_baf_centroid' 100
  Lower bound variable 'mid_baf_centroid' 15
  Upper bound variable 'mid_baf_centroid' 185
 Variable 'side_baf_centroid' 100
  Lower bound variable 'side_baf_centroid' 15
  Upper bound variable 'side_baf_centroid' 185
 Variable 'mid_baf_height' 100
  Lower bound variable 'mid_baf_height' 10
  Upper bound variable 'mid_baf_height' 180
 Variable 'side_baf_width' 100
  Lower bound variable 'side_baf_width' 10
  Upper bound variable 'side_baf_width' 180

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "../../fluent_script"
  solver input file "fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "2D_VF_gambit.jou"
  solver order quadratic
  solver experiment design dopt
  solver number experiments 32
  solver basis experiment 5toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'SB_side_constraint' expression {side_baf_centroid-side_baf_width/2}
 response 'SB_mid_constaint' expression {side_baf_centroid+side_baf_width/2}
 response 'MB_up_constaint' expression {mid_baf_centroid+mid_baf_height/2}
 response 'MB_low_constraint' expression {mid_baf_centroid-mid_baf_height/2}
 response 'baffle_length' expression {mid_baf_height+2*side_baf_width}
 response 'Percent_Fill' expression {100*fill_level/0.2}

$
$ OBJECTIVE FUNCTIONS
```

```
$
 objectives 1
 objective 'TDV' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 6
  move
 constraint 'SB_side_constraint'
  strict
  lower bound constraint 'SB_side_constraint' 10
  slack
 constraint 'SB_mid_constaint'
  strict
  upper bound constraint 'SB_mid_constaint' 190
 constraint 'MB_up_constaint'
  slack
  strict
  upper bound constraint 'MB_up_constaint' 190
 constraint 'MB_low_constraint'
  lower bound constraint 'MB_low_constraint' 10
  slack
  stay
 constraint 'baffle_length'
  lower bound constraint 'baffle_length' 0
  upper bound constraint 'baffle_length' 540
 constraint 'Percent_Fill'
  lower bound constraint 'Percent_Fill' 10
  upper bound constraint 'Percent_Fill' 90
$
$ JOB INFO
$
 iterate param design 0.001
 iterate param objective 0.001
 iterate param stoppingtype and
 iterate 5
STOP
```

APPENDIX R: Comparative sloshing frames for design 2b (Case 7 of Chapter 4)

Base case				
Final case (iteration 10)				
	Time = 0s	T = 0.125s	T = 0.25s	T = 0.375s
Base case				
Final case (iteration 10)				
	T = 0.5s	T = 0.625s	T = 0.75s	T = 0.875s
Base case				
Final case (iteration 10)				
	T = 1s	T = 1.125s	T = 1.25s	T = 1.375s
Base case				
Final case (iteration 10)				
	T = 1.5s	T = 1.625s	T = 1.75s	T = 1.825s

APPENDIX S: Sample Patran baffled tank session file

```
$
$               Patran session file for generating 90% full tank
$                   2D extended to 3D geometry ; baffles 1
$                   Created by: Thomas Kingsley 17/11/2003
$
uil_file_new.go( "/usr/local/msc/patran2003r2/lsdyna_prefrence.db",
"patran_tester1.db" )
db_set_pref( 303, 3, 0, FALSE, 0.0005, "" )
$
$
$                             Define Variables
$
$
REAL baffle_h
REAL baf_cent
REAL fe_size1
REAL fe_size2
$
$
$                             Variable values
$
$
baffle_h = <<baffle_H>>
baf_cent = <<baffle_centroid>>
baffle_t = 0.0001
$
fe_size1 = 0.01
fe_size2 = 0.0025
$
$                             Front
$
$
gu_fit_view(  )
ga_view_aa_set( -90., 0., 90. )
sys_poll_option( 2 )
ga_group_create( "rigid" )
ga_viewport_group_post( "", "rigid" )
sys_poll_option( 0 )
ga_group_current_set( "rigid" )
STRING asm_create_grid_xyz_created_ids[VIRTUAL]
asm_const_grid_xyz( "1", "[0 0 0]", "Coord 0", asm_create_grid_xyz_created_ids )
STRING asm_create_grid_xyz_created_ids[VIRTUAL]
asm_const_grid_xyz( "2", "[0 0 `baf_cent`]", "Coord 0",
asm_create_grid_xyz_created_ids )
STRING sgm_transform_point_created_ids[VIRTUAL]
asm_transform_grid_translate( "3", "<0 0 `-baffle_h/2`>", "Coord 0", 1, FALSE, FALSE,
"Point 2", sgm_transform_point_created_ids )
asm_transform_grid_translate( "4", "<0 0 `baffle_h/2`>", "Coord 0", 1, FALSE, FALSE,
"Point 2", sgm_transform_point_created_ids )
STRING asm_create_grid_xyz_created_ids[VIRTUAL]
asm_const_grid_xyz( "5", "[0 0 0.4]", "Coord 0", asm_create_grid_xyz_created_ids )
STRING asm_create_grid_xyz_created_ids[VIRTUAL]
asm_const_grid_xyz( "6", "[0 0 0.36]", "Coord 0", asm_create_grid_xyz_created_ids )
STRING asm_line_2point_created_ids[VIRTUAL]
asm_const_line_2point( "1", "Point 1", "Point 3", 0, "", 50., 1,
asm_line_2point_created_ids )
asm_const_line_2point( "2", "Point 3", "Point 4", 0, "", 50., 1,
asm_line_2point_created_ids )
asm_const_line_2point( "3", "Point 4", "Point 6", 0, "", 50., 1,
asm_line_2point_created_ids )
asm_const_line_2point( "4", "Point 6", "Point 5", 0, "", 50., 1,
asm_line_2point_created_ids )
STRING sgm_transform_curve_created_ids[VIRTUAL]
sgm_transform_translate( "5", "curve", "<0 0.4 0>", "Coord 0", 1, FALSE, "Curve 1:4",
sgm_transform_curve_created_ids )
STRING sgm_surface_2curve_created_ids[VIRTUAL]
sgm_const_surface_2curve( "1", "Curve 1", "Curve 5", sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "2", "Curve 2", "Curve 6", sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "3", "Curve 3", "Curve 7", sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "4", "Curve 4", "Curve 8", sgm_surface_2curve_created_ids )
$
```

```
$                              shells
$
INTEGER fem_create_mesh_surfa_num_nodes
INTEGER fem_create_mesh_surfa_num_elems
STRING fem_create_mesh_s_nodes_created[VIRTUAL]
STRING fem_create_mesh_s_elems_created[VIRTUAL]
fem_create_mesh_surf_4( "IsoMesh", 49152, "Surface 2", 1, ["`fe_size1`"], "Quad4",
"1", "1", "Coord 0", "Coord 0", fem_create_mesh_surfa_num_nodes,
fem_create_mesh_surfa_num_elems, fem_create_mesh_s_nodes_created,
fem_create_mesh_s_elems_created )
fem_create_mesh_surf_4( "IsoMesh", 49152, "Surface 1 3", 1, ["`fe_size1`"], "Quad4",
"5000", "5000", "Coord 0", "Coord 0", fem_create_mesh_surfa_num_nodes,
fem_create_mesh_surfa_num_elems, fem_create_mesh_s_nodes_created,
fem_create_mesh_s_elems_created )
fem_create_mesh_surf_4( "IsoMesh", 49152, "Surface 4", 1, ["`fe_size1`"], "Quad4",
"10000", "10000", "Coord 0", "Coord 0", fem_create_mesh_surfa_num_nodes,
fem_create_mesh_surfa_num_elems, fem_create_mesh_s_nodes_created,
fem_create_mesh_s_elems_created )
$
$
$                                Baffles
$
$
sys_poll_option( 2 )
ga_group_create( "baffles" )
ga_group_current_set( "baffles" )
STRING fem_transform_elem_created_nids[VIRTUAL]
STRING fem_transform_elem_created_eids[VIRTUAL]
STRING fem_transform_elem_deleted_nids[VIRTUAL]
STRING fem_transform_elem_deleted_eids[VIRTUAL]
fem_translate_elems_1( "20000", "<0.25 0 0>", "Coord 0", 1, FALSE, 2, "Elm 1:4500",
fem_transform_elem_created_nids, fem_transform_elem_created_eids,  @
fem_transform_elem_deleted_nids, fem_transform_elem_deleted_eids )
$? YESFORALL 2009007
$
$                              Water
$
sys_poll_option( 2 )
ga_group_create( "water" )
ga_group_current_set( "water" )
$
INTEGER fem_sweep_elems_n_nodes_created
INTEGER fem_sweep_elems_n_elems_created
STRING fem_sweep_elems_ex_created_nids[VIRTUAL]
STRING fem_sweep_elems_ex_created_eids[VIRTUAL]
fem_sweep_extrude_1( "30000", "30000", "Coord 0", "<0.25 0 0>", "0.25", "0.0 ",  @
2, "Elm 1:9500", 10, ["Bar2", "Quad4", "Quad8", "Quad12", "Wedge6", "Wedge15" @
, "Wedge24", "Hex8", "Hex20", "Hex32", "", "", "", "", "", "", "", "", "", "", @
 "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",  @
"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "" @
, "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",  @
"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "" @
, "", ""], "Coord 0", "Coord 0", "Uniform: Element Length", ["3", "1.5", "0.1" @
, "0.2", "`fe_size1`", "2", "", "", "", ""], fem_sweep_elems_n_nodes_created,  @
fem_sweep_elems_n_elems_created, fem_sweep_elems_ex_created_nids,  @
fem_sweep_elems_ex_created_eids )
$? YESFORALL 2001200
$
STRING fem_transform_elem_created_nids[VIRTUAL]
STRING fem_transform_elem_created_eids[VIRTUAL]
STRING fem_transform_elem_deleted_nids[VIRTUAL]
STRING fem_transform_elem_deleted_eids[VIRTUAL]
fem_translate_elems_1( "100000", "<0.25 0 0>", "Coord 0", 1, FALSE, 2,  @
"Elm 30000:95000", fem_transform_elem_created_nids,  @
fem_transform_elem_created_eids, fem_transform_elem_deleted_nids,  @
fem_transform_elem_deleted_eids )
$? YESFORALL 2009007
$
$                              Air
$
sys_poll_option( 2 )
ga_group_create( "air" )
ga_group_current_set( "air" )
$
INTEGER fem_sweep_elems_n_nodes_created
INTEGER fem_sweep_elems_n_elems_created
STRING fem_sweep_elems_ex_created_nids[VIRTUAL]
STRING fem_sweep_elems_ex_created_eids[VIRTUAL]
fem_sweep_extrude_1( "300000", "300000", "Coord 0", "<0.25 0 0>", "0.25", "0.0 " @
, 2, "Elm 10000:19000", 10, ["Bar2", "Quad4", "Quad8", "Quad12", "Wedge6",  @
"Wedge15", "Wedge24", "Hex8", "Hex20", "Hex32", "", "", "", "", "", "", "", "" @
, "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",  @
"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "" @
, "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",  @
"", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "" @
, "", "", "", ""], "Coord 0", "Coord 0", "Uniform: Element Length", ["3",  @
"1.5", "0.1", "0.2", "`fe_size1`", "2", "", "", "", ""],  @
fem_sweep_elems_n_nodes_created, fem_sweep_elems_n_elems_created,  @
fem_sweep_elems_ex_created_nids, fem_sweep_elems_ex_created_eids )
$? YESFORALL 2001200
$
fem_translate_elems_1( "320000", "<0.25 0 0>", "Coord 0", 1, FALSE, 2,  @
"Elm 300000:318000", fem_transform_elem_created_nids,  @
fem_transform_elem_created_eids, fem_transform_elem_deleted_nids,  @
fem_transform_elem_deleted_eids )
$? YESFORALL 2009007
$
$                              Close box
$
ga_group_current_set( "rigid" )
STRING sgm_transform_surf__created_ids[VIRTUAL]
sgm_transform_translate( "5", "surface", "<0.25 0 0>", "Coord 0", 2, FALSE,  @
"Surface 1:4", sgm_transform_surf__created_ids )
STRING sgm_surface_2curve_created_ids[VIRTUAL]
sgm_const_surface_2curve( "13", "Curve 1", "Surface 5.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "14", "Curve 2", "Surface 6.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "15", "Curve 3", "Surface 7.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "16", "Curve 4", "Surface 8.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "17", "Surface 5.4", "Surface 9.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "18", "Surface 6.4", "Surface 10.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "19", "Surface 7.4", "Surface 11.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "20", "Surface 8.4", "Surface 12.4",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "21", "Surface 12.2", "Surface 8.2",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "22", "Surface 8.2", "Curve 8",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "23", "Curve 5", "Surface 5.2",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "24", "Surface 5.2", "Surface 9.2",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "25", "Surface 10.2", "Surface 6.2",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "26", "Curve 6", "Surface 6.2",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "27", "Curve 7", "Surface 7.2",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "28", "Surface 1.1", "Surface 5.1",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "29", "Surface 5.1", "Surface 9.1",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "30", "Surface 4.3", "Surface 8.3",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "31", "Surface 8.3", "Surface 12.3",  @
sgm_surface_2curve_created_ids )
sgm_const_surface_2curve( "32", "Surface 7.2", "Surface 11.2",  @
sgm_surface_2curve_created_ids )
INTEGER fem_create_mesh_surfa_num_nodes
INTEGER fem_create_mesh_surfa_num_elems
STRING fem_create_mesh_s_nodes_created[VIRTUAL]
STRING fem_create_mesh_s_elems_created[VIRTUAL]
fem_create_mesh_surf_4( "IsoMesh", 49152, "Surface 9:32", 1, ["`fe_size1`"],  @
"Quad4", "500000", "500000", "Coord 0", "Coord 0",  @
fem_create_mesh_surfa_num_nodes, fem_create_mesh_surfa_num_elems,  @
fem_create_mesh_s_nodes_created, fem_create_mesh_s_elems_created )
$
$                              Create Materials (dummy)
```

```
$
material.create( "Analysis code ID", 10001, "Analysis type ID", 1, "rigid_mat" @
, 0, "Date: 27-Aug-03           Time: 12:17:54", "Isotropic", 1,  @
"Directionality", 1, "Linearity", 11001, "Homogeneous", 0, "Rigid", 11001,  @
"Model Options & IDs", ["Material Type 20", "", "", "", ""], [11006, 0, 0, 0,  @
0], "Active Flag", 1, "Create", 10, "External Flag", FALSE, "Property IDs", [ @
"Density", "Elastic Modulus", "Poisson Ratio"], [16, 2, 5, 0],  @
"Property Values", ["7280", "2e11", "0.3", ""] )
material.create( "Analysis code ID", 10001, "Analysis type ID", 1,  @
"baffle_mat", 0, "Date: 27-Aug-03           Time: 12:17:54", "Isotropic", 1,  @
"Directionality", 1, "Linearity", 1, "Homogeneous", 0, "Linear Elastic", 1,  @
"Model Options & IDs", ["Linear Elastic(MAT1)", "Solid", "", "", ""], [11002,  @
11003, 0, 0, 0], "Active Flag", 1, "Create", 10, "External Flag", FALSE,  @
"Property IDs", ["Density", "Elastic Modulus", "Poisson Ratio"], [16, 2, 5, 0] @
, "Property Values", ["7280.", "2E+11", "0.30000001", ""] )
material.create( "Analysis code ID", 10001, "Analysis type ID", 1, "water_mat" @
, 0, "Date: 27-Aug-03           Time: 12:17:54", "Isotropic", 1,  @
"Directionality", 1, "Linearity", 1, "Homogeneous", 0, "Linear Elastic", 1,  @
"Model Options & IDs", ["Linear Elastic(MAT1)", "Fluid", "", "", ""], [11002,  @
11004, 0, 0, 0], "Active Flag", 1, "Create", 10, "External Flag", FALSE,  @
"Property IDs", ["Density", "Bulk Modulus"], [16, 14, 0], "Property Values", [ @
"1000.", "2e9", ""] )
material.create( "Analysis code ID", 10001, "Analysis type ID", 1, "air_mat",  @
0, "Date: 27-Aug-03           Time: 12:17:54", "Isotropic", 1,  @
"Directionality", 1, "Linearity", 1, "Homogeneous", 0, "Linear Elastic", 1,  @
"Model Options & IDs", ["Linear Elastic(MAT1)", "Fluid", "", "", ""], [11002,  @
11004, 0, 0, 0], "Active Flag", 1, "Create", 10, "External Flag", FALSE,  @
"Property IDs", ["Density", "Bulk Modulus"], [16, 14, 0], "Property Values", [ @
"1.", "2e5", ""] )
$
$                             Assign materials to elements
$
elementprops_create( "rigid.1", 51, 25, 35, 11003, 1, 20, [13, 20, 36, 1004,  @
11044, 11136, 11026, 11027, 1011, 11182], [5, 2, 1, 1, 4, 3, 4, 4, 1, 1], [ @
"m:rigid_mat", "", "0.01", "", "", "", "", "", "", ""],  @
"Element 1:19000 500000:600000" )
elementprops_create( "baffle.2", 51, 25, 35, 11003, 1, 20, [13, 20, 36, 1004,  @
11044, 11136, 11026, 11027, 1011, 11182], [5, 2, 1, 1, 4, 3, 4, 4, 1, 1], [ @
"m:baffle_mat", "", "0.001", "", "", "", "", "", "", ""],  @
"Element 20000:29000" )
elementprops_create( "air.3", 71, 25, 20, 11027, 1, 20, [13, 20, 1011, 11182], @
 [5, 2, 1, 1], ["m:air_mat", "", " ", " "], "Element 300000:400000" )
elementprops_create( "water.4", 71, 25, 20, 11027, 1, 20, [13, 20, 1011,  @
11182], [5, 2, 1, 1], ["m:water_mat", "", " ", " "], "Element 30000:290000" )
$
$                             Equivilence
$
REAL fem_equiv_all_x_equivtol
INTEGER fem_equiv_all_x_segment
fem_equiv_all_group3( [" "], 0, "", 1, 1.0E-04, FALSE,  @
fem_equiv_all_x_equivtol, fem_equiv_all_x_segment )
$
$                             Delete duplicates
$
INTEGER fem_verify_duplicates_num_dupl
STRING fem_verify_duplicates_keep_ids[VIRTUAL]
STRING fem_verify_duplicates_toss_ids[VIRTUAL]
fem_verify_element_duplicates( TRUE, TRUE, fem_verify_duplicates_num_dupl,
fem_verify_duplicates_keep_ids, fem_verify_duplicates_toss_ids )
$
$                             Translate to LS-Dyna keyword file
$
$
loadsbcs_eval_all(  )
jobfile.open( "patran_keyword", "ANALYZE NO JOBFILE" )
jobfile.create_param( "version_opt", 0, 0., "LS-930", 4 )
jobfile.create_param( "sep_mesh", 0, 0., "OFF", 4 )
jobfile.create_param( "sep_mat", 0, 0., "OFF", 4 )
jobfile.create_param( "label_file", 0, 0., "*.sif", 4 )
jobfile.create_param( "list_file", 0, 0., "", 1 )
jobfile.create_param( "cpu_time_limit", 0, 0., "", 3 )
jobfile.create_param( "termination_step", 0, 0., "", 1 )
jobfile.create_param( "termination_time", 0, 0., "", 3 )
jobfile.create_param( "time_step_ratio_limit", 0, 0., "", 3 )
jobfile.create_param( "energy_ratio_limit", 0, 0., "", 3 )
jobfile.create_param( "mass_ratio_limit", 0, 0., "", 3 )
jobfile.create_param( "time_size_calc_basis", 0, 0., "Area/Edge", 4 )
jobfile.create_param( "initial_time_step", 0, 0., "", 3 )
jobfile.create_param( "step_scale_factor", 0, 0.89999998, "", 3 )
jobfile.create_param( "step_size_for_mass_scale", 0, 0., "", 3 )
jobfile.create_param( "shell_minimum_time_step", 0, 0., "", 3 )
jobfile.create_param( "load_curve_max_time_step", 0, 0., "", 1 )
jobfile.create_param( "erosion_flag", 0, 0., "OFF", 4 )
jobfile.create_param( "mass_scaling_first_step", 0, 0., "OFF", 4 )
jobfile.create_param( "Number_of_cpus", 1, 0., "", 1 )
jobfile.create_param( "One_rhs_only", 0, 0., "OFF", 4 )
jobfile.create_param( "Consistency_flg", 0, 0., "OFF", 4 )
jobfile.create_param( "parallel_force", 0, 0., "OFF", 4 )
jobfile.create_param( "optmenu_relaxation", 0, 0., "None.Active", 4 )
jobfile.create_param( "geometry_file", 0, 0., "", 4 )
jobfile.create_param( "relax_termination_time", 0, 1E+30, "", 3 )
jobfile.create_param( "con_tolerance", 0, 0.001, "", 3 )
jobfile.create_param( "Iterations_checks", 250, 0., "", 1 )
jobfile.create_param( "auto_control", 0, 0., "OFF", 4 )
jobfile.create_param( "papadrakakis", 0, 0.039999999, "", 3 )
jobfile.create_param( "Relaxation_Factor", 0, 0.995, "", 3 )
jobfile.create_param( "Time_scale_Factor", 0, 0.89999998, "", 3 )
jobfile.create_param( "global_damping_curve", 0, 0., "f:ouatiafltlak", 4 )
jobfile.create_param( "system_damping_constant", 0, 0., "", 3 )
jobfile.create_param( "linear_viscosity_coefficient", 0, 0.059999999, "", 3 )
jobfile.create_param( "quadratic_viscosity_coefficient", 0, 1.5, "", 3 )
jobfile.create_param( "hourglass_viscosity_type", 0, 0., "LS_DYNA", 4 )
jobfile.create_param( "hourglass_viscosity_coefficient", 0, 0.1, "", 3 )
jobfile.create_param( "hourglass_energy_calc", 0, 0., "OFF", 4 )
jobfile.create_param( "stonewall_energy_diss", 0, 0., "ON", 4 )
jobfile.create_param( "sliding_int_energy_diss", 0, 0., "OFF", 4 )
jobfile.create_param( "rayleigh_energy_diss", 0, 0., "OFF", 4 )
jobfile.create_param( "Warning_angle_warpage", 0, 20., "", 3 )
jobfile.create_param( "Treat_degen_quads_as_tris", 0, 0., "OFF", 4 )
jobfile.create_param( "shell_theory", 0, 0., "Belytschko", 4 )
jobfile.create_param( "warping_stiffness", 0, 0., "Belytschko-Tsay", 4 )
jobfile.create_param( "normal_update", 0, 0., "Each Cycle", 4 )
jobfile.create_param( "update_option", -1, 0., "", 1 )
jobfile.create_param( "update_shell_thick", 0, 0., "OFF", 4 )
jobfile.create_param( "plastics_method", 0, 0., "Secant", 4 )
jobfile.create_param( "consider_shell_thickness", 0, 0.,  @
"Thickness not considered", 4 )
jobfile.create_param( "check_penetration", 0, 0., "ON", 4 )
jobfile.create_param( "max_check_multiplier", 0, 4., "", 3 )
jobfile.create_param( "step_bet_search", 10, 0., "", 1 )
jobfile.create_param( "search_old_surface", 0, 0., "OFF", 4 )
jobfile.create_param( "stiffness_value", 0, 0., "Min. of Master & Slave", 4 )
jobfile.create_param( "scale_interface", 0, 0.1, "", 3 )
jobfile.create_param( "scale_rigid_wall", 0, 0., "", 3 )
jobfile.create_param( "shell_thick_include", 0, 0., "OFF", 4 )
jobfile.create_param( "auto_reorientation", 0, 0., "Active for Automated", 4 )
jobfile.create_param( "control_subroutine", 0, 0., "", 1 )
jobfile.create_param( "friction_subroutine", 0, 0., "", 1 )
jobfile.create_param( "bin_state_time_int", 0, 0., "", 3 )
jobfile.create_param( "excl_damp_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "local_coord_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "one_plot_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "elim_rigid_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "output_hglass_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "output_time_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "num_beam_int_dbox", 0, 0., "", 1 )
jobfile.create_param( "inc_surf_strain_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "excl_sh_tensor_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "excl_sh_strain_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "excl_sh_res_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "excl_int_energy_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "num_var_sol_dbox", 0, 0., "", 1 )
jobfile.create_param( "num_var_sh_dbox", 0, 0., "", 1 )
jobfile.create_param( "num_sh_int_dbox", 0, 0., "", 1 )
jobfile.create_param( "bin_history_time_int", 0, 0., "", 3 )
jobfile.create_param( "inc_nodes_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "inc_beams_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "inc_shell_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "inc_solid_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "inc_th_sh_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "inc_extra_tg", 0, 0., "OFF", 4 )
jobfile.create_param( "Pri_During_Input", 0, 0., "ON", 4 )
jobfile.create_param( "Echo_File_Options", 0, 0., "Suppress Both", 4 )
jobfile.create_param( "Velocities", 0, 0., "OFF", 4 )
jobfile.create_param( "Update_Beam", 0, 0., "OFF", 4 )
jobfile.create_param( "Interface_File", 0, 0., "", 3 )
jobfile.create_param( "Interface_File_Name", 0, 0., "INTFOR", 4 )
jobfile.create_param( "Time_Step", 0, 0., "ON", 4 )
jobfile.create_param( "Time_Interval", 0, 0., "", 1 )
jobfile.create_param( "Tolerance_Status", 0, 0., "OFF", 4 )
jobfile.create_param( "Tolerance_Value", 0, 0., "", 3 )
jobfile.create_param( "viewport", 0, 0., "default_viewport", 4 )
jobfile.create_param( "datbox_groupname", 0, 0., "default_group", 4 )
jobfile.create_param( "tranlational", 0, 0., "<0,0,0>", 4 )
jobfile.create_param( "rotational", 0, 0., "<0,0,0>", 4 )
jobfile.create_param( "loadcase_for_analysis", 0, 0., "Default", 4 )
jobfile.create_param( "set_node_opt", 0, 0., "YES", 4 )
jobfile.create_param( "set_beam_opt", 0, 0., "YES", 4 )
jobfile.create_param( "set_discrete_opt", 0, 0., "YES", 4 )
jobfile.create_param( "set_shell_opt", 0, 0., "YES", 4 )
jobfile.create_param( "set_solid_opt", 0, 0., "YES", 4 )
jobfile.create_param( "set_tshell_opt", 0, 0., "YES", 4 )
jobfile.create_param( "w_db_node", 5, 0., "", 1 )
jobfile.create_param( "w_db_beam", 5, 0., "", 1 )
jobfile.create_param( "w_db_discrete", 5, 0., "", 1 )
jobfile.create_param( "w_db_shell", 5, 0., "", 1 )
jobfile.create_param( "w_db_solid", 5, 0., "", 1 )
jobfile.create_param( "w_db_tshell", 5, 0., "", 1 )
jobfile.create_param( "all_groups_in_db_id_1", 0, 0., "baffles", 101 )
jobfile.create_param( "all_groups_in_db_id", 1, 0., "", 1 )
jobfile.create_param( "setcard_fullmodel_group_id_1", 0, 0., "baffles", 101 )
jobfile.create_param( "setcard_fullmodel_group_id", 1, 0., "", 1 )
jobfile.create_param( "factor_length_from_MADYMO", 0, 1., "", 3 )
jobfile.create_param( "factor_on_time_from_MADYMO", 0, 1., "", 3 )
jobfile.create_param( "factor_on_force_from_MADYMO", 0, 1., "", 3 )
jobfile.create_param( "Wait_time_as_MADYMO_computes", 0, 0., "", 3 )
jobfile.create_param( "Flip_X_coord_of_MADYMO", 0, 0., "OFF", 4 )
jobfile.create_param( "Flip_Y_coord_of_MADYMO", 0, 0., "OFF", 4 )
jobfile.create_param( "Flip_Z_coord_of_MADYMO", 0, 0., "OFF", 4 )
jobfile.create_param( "Num_Dyna_steps_per_MADYMO_step", 1, 0., "", 1 )
jobfile.create_param( "filename_interface", 0, 0.,  @
"Default_interfacefile.isf1", 4 )
jobfile.create_param( "restart_label_file", 0, 0., "", 4 )
jobfile.create_param( "DTIwrite_switch_id", 0, 0., "End Deck", 4 )
jobfile.create_param( "DTIdirect_text_toggle_id", 0, 0., "OFF", 4 )
jobfile.close(  )
uil_file_close.go(  )
sys_library( "add", "lsdyna3d.plb" )
lsdyna3d_spawn_generic( "pat3lsdyna", " -d patran_tester1.db -j patran_keyword", TRUE
)
$uil_file_open.go( "patran_tester1.db" )
$
$
$                                    END
$
```

APPENDIX T: Mesh file cleaning SED file

```
1,28 d
/SECTION_SHELL$/{
n
n
d
}
/SECTION_SHELL$/{
n
d
}
/SECTION_SHELL$/{
d
}
/*PART$/{
n
n
d
}
/*PART$/{
n
d
}
/*PART$/{
d
}
/SECTION_SOLID$/{
n
d
}
/SECTION_SOLID$/{
d
}
/Material :/,$ d
w mesh.k
```

APPENDIX U: LS-DYNA keyword file (Model settings section only)

```
*KEYWORD 100000000
$ continuum control
*CONTROL_ALE
2,1,2,-1.00,.000E+00,.000E+00,.000E+00,.000E+00
.000E+00,.000E+00,.000E+00,.000E+00,.000E+00,0
*CONTROL_CPU
18e3
*CONTROL_TERMINATION
<<term_time>>,0,.000E+00,.000E+00,.000E+00
*DATABASE_BINARY_D3PLOT
<<d3_interval>>,0
*DATABASE_BINARY_D3DUMP
<<dump_interval>>
$ define degree of freedom
*BOUNDARY_PRESCRIBED_MOTION_NODE
1,1,1,3,1.00,0,0.000,0.000
*load_body_z
2,9.81
*set_part_list
2
3,4
*set_part_list
1
1,2
*DATABASE_HISTORY_SHELL
$id1,id2,id3,id4,id5,id6,id7,id8
$
20000,20001,20002,20003
$
*DATABASE_HISTORY_NODE
1
$3318,4578,5838,7098,3977,5237,6497,7757
*DATABASE_ELOUT
$       dt
    0.0001         3
*DATABASE_NODOUT
    0.0001         3
*CONSTRAINED_LAGRANGE_IN_SOLID
$ Coupling control (penalty/energy)
         2         2         1         0         2         4         2
1
$    START       END
                                                  0.3        0
                                       1
*ALE_MULTI-MATERIAL_GROUP
$ group mixable materials (air/water)
         3         1
         4         1
*ALE_REFERENCE_SYSTEM_GROUP
$ set group that will follow local coordinate system
2,0,5,1
0
*ALE_REFERENCE_SYSTEM_NODE
$ define local coordinate system RHR (x1,x2,x3)
1
5000,1,5100
0
$
$  Rigid
$
*MAT_RIGID
1,7.830E+03,2.070E+11,0.300,0.
```

```
$ constraint directions (translation y-z)(all rotation)
1.0,5.0,7.0
0
*HOURGLASS
$ Bulk viscosity properties
1,0,0.,0,0.,0.
*SECTION_SHELL
$ 1 integration point (no bending)
1,2,0.,1.,0.,0.,0
1.000E-02,1.000E-002,1.000E-02,1.000E-02,0.
*PART
material type # 20 (Rigid)
1,1,1,0,1,0
$
$  Baffles
$
*MAT_PLASTIC_KINEMATIC
2,7.830E+03,2.070E+11,0.300,5.000E+09,100.,0.
0.,0.,0.
*HOURGLASS
2,0,0.,0,0.,0.
*SECTION_SHELL
2,2,0.,0.,0.,0.,0
$ Baffle thickness
<<baffle_T>>,<<baffle_T>>,<<baffle_T>>,<<baffle_T>>,0.
*PART
material type # 3 (Kinematic/Isotropic Elastic-Plastic)
2,2,2,0,2,0
$
$ Air
$
*MAT_NULL
         4    1.1845       0.0 0.0000184       0.0       0.0       0.0
0.0
*HOURGLASS
4,0,0.,0,0.,0.
*EOS_LINEAR_POLYNOMIAL_WITH_ENERGY_LEAK
4,0.0,0.0,0.0,0.0,0.4,0.4,0.0
,1.0,0
*SECTION_SOLID
4,11,0
*PART
outer air (kg-m-s)
3,4,4,4,4,0
$
$ Water
$
*MAT_NULL
         3     998.0       0.0     0.001       0.0       0.0       0.0
0.0
*HOURGLASS
3,0,0.,0,0.,0.
*EOS_GRUNEISEN
3,1647.0,1.921,-0.096,0.0,0.35,0.0,0.0
0.0
*SECTION_SOLID
3,11,0
*PART
water (kg-m-s)
4,3,3,3,3,0
$
*INCLUDE
mesh.k
*END

$
$ LOAD CURVES
$
```

```
*DEFINE_CURVE
1,0,.000E+00,.000E+00,.000E+00,.000E+00,0
.0000000000000E+00,800.0000000000
1.0000000474975E-03,.0000000000000E+00
5.000000000000,.0000000000000E+00
*DEFINE_CURVE
2,0,.000E+00,.000E+00,.000E+00,.000E+00,0
.0000000000000E+00,1.000000000000
5.000000000000,1.000000000000
*DEFINE_CURVE
3,0,.000E+00,.000E+00,.000E+00,.000E+00,0
0,20.34915942
0.000005,22.56476372
…
… (acceleration data)
…
0.01,0
0.09,0
*END
```

APPENDIX V: LSOPT command file – 3D impact only optimisation

```
"Structural optimisation of 3D baffled tank to reduce mass and maintain structural
integrity."
Author "K Craig, T C Kingsley"
$ Created on Sat Nov  6 11:31:41 2004
solvers 1
responses 4
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 3
 Variable 'baffle_H' 0.1
  Lower bound variable 'baffle_H' 0.08
  Upper bound variable 'baffle_H' 0.3
 Variable 'hole_D' 0.025
  Lower bound variable 'hole_D' 0.015
  Upper bound variable 'hole_D' 0.05
 Variable 'baffle_T' 0.002
  Lower bound variable 'baffle_T' 0.001
  Upper bound variable 'baffle_T' 0.01
$
$ CONSTANTS
$
constants 3
 Constant 'term_time' 0.015
 Constant 'd3_interval' 0.002
 Constant 'dump_interval' 0.005

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Patran-Dyna"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Patran-Dyna"
$
 solver dyna960 'Patran-Dyna'
  solver command "ls970"
  solver input file "header.k"
  solver append file "footer.k"
  prepro own
  prepro command "../../process"
  prepro input file "prepro_pre_tank.ses"
  solver order linear
  solver experiment design dopt
  solver number experiments 7
  solver basis experiment 3toK
  solver concurrent jobs 1
$
$ RESPONSES FOR SOLVER "Patran-Dyna"
$
 response 'Max_PriStress_baff' 1 0 "DynaPStress S1 2 MAX"
 response 'baffle_mass' 1 0 "DynaMass 2 MASS"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "Patran-Dyna"
$
 response 'too_big_hole' expression {(baffle_H/2)-hole_D}
 response 'baffle_mass_calc' expression {7830*baffle_T*(4*(baffle_H*0.4-
(2*(3.1415926536*hole_D**2))))}

$
$ OBJECTIVE FUNCTIONS
```

```
$
 objectives 1
 objective 'baffle_mass_calc' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 2
 constraint 'Max_PriStress_baff'
  strict
  upper bound constraint 'Max_PriStress_baff' 2e+08
  move
 constraint 'too_big_hole'
  lower bound constraint 'too_big_hole' 0.03
  slack
$
$ JOB INFO
$
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype and
 iterate 6
STOP
```

APPENDIX W: LSOPT command file – 2D extruded impact only optimisation

```
"Impact analysis of baffled liquid container (Effective element stress)"
Author "Thomas Kingsley, Ken Craig"
$ Created on Mon Jul 26 08:17:05 2004
solvers 1
responses 10
$
$ NO HISTORIES ARE DEFINED
$
$
$ DESIGN VARIABLES
$
variables 3
 Variable 'mid_baf_centroid' 100
  Lower bound variable 'mid_baf_centroid' 60
  Upper bound variable 'mid_baf_centroid' 320
 Variable 'mid_baf_height' 100
  Lower bound variable 'mid_baf_height' 40
  Upper bound variable 'mid_baf_height' 320
 Variable 'baffle_T' 0.008
  Lower bound variable 'baffle_T' 0.001
  Upper bound variable 'baffle_T' 0.015
$
$ CONSTANTS
$
constants 3
 Constant 'term_time' 0.015
 Constant 'd3_interval' 0.001
 Constant 'dump_interval' 0.005
$
$ DEPENDENT VARIABLES
$
dependent 2
 Dependent 'baffle_H' {mid_baf_height/1000}
 Dependent 'baffle_centroid' {mid_baf_centroid/1000}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Patran-Dyna"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Patran-Dyna"
$
 solver dyna960 'Patran-Dyna'
  solver command "../../dyna_script"
  solver input file "header.k"
  solver append file "footer.k"
  prepro own
  prepro command "../../process"
  prepro input file "pat_prepro_2Dtank.ses"
  solver order linear
  solver experiment design dopt
  solver number experiments 7
  solver basis experiment 3toK
  solver concurrent jobs 2
$
$ RESPONSES FOR SOLVER "Patran-Dyna"
$
 response 'max_eff_stress_20000b' 1 0 "DynaASCII Elout E_STRESS 20000 2 MAX 0 0.1"
 response 'max_eff_stress_20000a' 1 0 "DynaASCII Elout E_STRESS 20000 1 MAX 0 0.1"
 response 'max_eff_stress_20001a' 1 0 "DynaASCII Elout E_STRESS 20001 1 MAX 0 0.1"
 response 'max_eff_stress_20001b' 1 0 "DynaASCII Elout E_STRESS 20001 2 MAX 0 0.1"
 response 'max_eff_stress_20002b' 1 0 "DynaASCII Elout E_STRESS 20002 2 MAX 0 0.1"
 response 'max_eff_stress_20002a' 1 0 "DynaASCII Elout E_STRESS 20002 1 MAX 0 0.1"
 response 'max_eff_stress_20003b' 1 0 "DynaASCII Elout E_STRESS 20003 2 MAX 0 0.1"
 response 'max_eff_stress_20003a' 1 0 "DynaASCII Elout E_STRESS 20003 1 MAX 0 0.1"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "Patran-Dyna"
$
 response 'baffle_upper' expression {(mid_baf_height/2)+mid_baf_centroid}
 response 'baffle_lower' expression {mid_baf_centroid-(mid_baf_height/2)}

composites 9
$
$ COMPOSITE RESPONSES
$
 composite 'Max_eff_stress_scaled_0a' type weighted
  composite 'Max_eff_stress_scaled_0a' response 'max_eff_stress_20000a' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_0b' type weighted
  composite 'Max_eff_stress_scaled_0b' response 'max_eff_stress_20000b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_1b' type weighted
  composite 'Max_eff_stress_scaled_1b' response 'max_eff_stress_20001b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_1a' type weighted
  composite 'Max_eff_stress_scaled_1a' response 'max_eff_stress_20001a' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_2a' type weighted
  composite 'Max_eff_stress_scaled_2a' response 'max_eff_stress_20002a' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_2b' type weighted
  composite 'Max_eff_stress_scaled_2b' response 'max_eff_stress_20002b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_3b' type weighted
  composite 'Max_eff_stress_scaled_3b' response 'max_eff_stress_20003b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_3a' type weighted
  composite 'Max_eff_stress_scaled_3a' response 'max_eff_stress_20003a' 1 scale 1e+07
$
$ COMPOSITE EXPRESSIONS
$
 composite 'Baffle_Mass' {10*(7830*baffle_T*0.4*((mid_baf_height/1000)+2*0.01))}
$
$ OBJECTIVE FUNCTIONS
$
 objectives 1
 objective 'Baffle_Mass' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 10
  move
 constraint 'baffle_upper'
  strict
  upper bound constraint 'baffle_upper' 340
 constraint 'baffle_lower'
  lower bound constraint 'baffle_lower' 20
  slack
  stay
 constraint 'Max_eff_stress_scaled_0a'
  strict
  upper bound constraint 'Max_eff_stress_scaled_0a' 20
 constraint 'Max_eff_stress_scaled_0b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_0b' 20
 constraint 'Max_eff_stress_scaled_1b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_1b' 20
 constraint 'Max_eff_stress_scaled_1a'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_1a' 20
 constraint 'Max_eff_stress_scaled_2a'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_2a' 20
 constraint 'Max_eff_stress_scaled_2b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_2b' 20
 constraint 'Max_eff_stress_scaled_3b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_3b' 20
 constraint 'Max_eff_stress_scaled_3a'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_3a' 20
$
$ JOB INFO
$
 concurrent jobs 1
 iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype and
 iterate 8
STOP
```

APPENDIX X: LSOPT command file – Multidisciplinary optimisation (Sloshing and impact)

```
"MDO of baffled liquid container (scaled TDV and effective element stress)"
Author "Thomas Kingsley, Ken Craig"
$ Created on Mon Sep 13 10:16:44 2004
solvers 2
responses 16
$
$ NO HISTORIES ARE DEFINED
$
$ DESIGN VARIABLES
$
variables 5
 Variable 'mid_baf_centroid' 100
  Lower bound variable 'mid_baf_centroid' 60
  Upper bound variable 'mid_baf_centroid' 320
 Variable 'side_baf_centroid' 100
  Lower bound variable 'side_baf_centroid' 15
  Upper bound variable 'side_baf_centroid' 185
  Local 'side_baf_centroid'
 Variable 'mid_baf_height' 100
  Lower bound variable 'mid_baf_height' 40
  Upper bound variable 'mid_baf_height' 320
 Variable 'side_baf_width' 100
  Lower bound variable 'side_baf_width' 10
  Upper bound variable 'side_baf_width' 180
  Local 'side_baf_width'
 Variable 'baffle_T' 0.008
  Lower bound variable 'baffle_T' 0.001
  Upper bound variable 'baffle_T' 0.015
  Local 'baffle_T'
$
$ CONSTANTS
$
constants 4
 Constant 'fill_level' 0.28
 Constant 'term_time' 0.015
 Constant 'd3_interval' 0.001
 Constant 'dump_interval' 0.005
$
$ DEPENDENT VARIABLES
$
dependent 4
 Dependent 'baffle_H' {mid_baf_height/1000}
 Dependent 'baffle_centroid' {mid_baf_centroid/1000}
 Dependent 'side_baffle_W' {side_baf_width/1000}
 Dependent 'side_baffle_cent' {side_baf_centroid/1000}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      OPTIMIZATION METHOD
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
Optimization Method SRSM

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "fluent"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "fluent"
$
 solver own 'fluent'
  solver command "../../fluent_script"
  solver input file "fluentjou2D.jou"
  prepro own
  prepro command "gambit -inp"
  prepro input file "2D_tom_gambit.jou"
  solver order linear
  solver experiment design dopt
  solver basis experiment 3toK
  solver concurrent jobs 1
$
```

```
$ LOCAL DESIGN VARIABLES FOR SOLVER "fluent"
$
  solver variable 'side_baf_centroid'
  solver variable 'side_baf_width'
$
$ RESPONSES FOR SOLVER "fluent"
$
 response 'TDV' 1000 0 "cat TotalDev.txt"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "fluent"
$
 response 'SB_side_constraint' expression {side_baf_centroid-side_baf_width/2}
 response 'SB_mid_constaint' expression {side_baf_centroid+side_baf_width/2}
 response 'MB_up_constaint' expression {mid_baf_centroid+mid_baf_height/2}
 response 'MB_low_constraint' expression {mid_baf_centroid-mid_baf_height/2}
 response 'baffle_length' expression {mid_baf_height+2*side_baf_width}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      SOLVER "Patran-Dyna"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ DEFINITION OF SOLVER "Patran-Dyna"
$
 solver dyna960 'Patran-Dyna'
  solver command "../../dyna_script"
  solver input file "header.k"
  solver append file "footer.k"
  prepro own
  prepro command "../../process"
  prepro input file "pat_prepro_2Dtank.ses"
  solver order linear
  solver experiment design dopt
  solver basis experiment 3toK
  solver concurrent jobs 2
$
$ LOCAL DESIGN VARIABLES FOR SOLVER "Patran-Dyna"
$
  solver variable 'baffle_T'
$
$ RESPONSES FOR SOLVER "Patran-Dyna"
$
 response 'max_eff_stress_20000b' 1 0 "DynaASCII Elout E_STRESS 20000 2 MAX 0 0.1"
 response 'max_eff_stress_20000a' 1 0 "DynaASCII Elout E_STRESS 20000 1 MAX 0 0.1"
 response 'max_eff_stress_20001a' 1 0 "DynaASCII Elout E_STRESS 20001 1 MAX 0 0.1"
 response 'max_eff_stress_20001b' 1 0 "DynaASCII Elout E_STRESS 20001 2 MAX 0 0.1"
 response 'max_eff_stress_20002b' 1 0 "DynaASCII Elout E_STRESS 20002 2 MAX 0 0.1"
 response 'max_eff_stress_20002a' 1 0 "DynaASCII Elout E_STRESS 20002 1 MAX 0 0.1"
 response 'max_eff_stress_20003b' 1 0 "DynaASCII Elout E_STRESS 20003 2 MAX 0 0.1"
 response 'max_eff_stress_20003a' 1 0 "DynaASCII Elout E_STRESS 20003 1 MAX 0 0.1"
$
$ RESPONSE EXPRESSIONS FOR SOLVER "Patran-Dyna"
$
 response 'baffle_upper' expression {(mid_baf_height/2)+mid_baf_centroid}
 response 'baffle_lower' expression {mid_baf_centroid-(mid_baf_height/2)}

composites 9
$
$ COMPOSITE RESPONSES
$
 composite 'Max_eff_stress_scaled_0a' type weighted
  composite 'Max_eff_stress_scaled_0a' response 'max_eff_stress_20000a' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_0b' type weighted
  composite 'Max_eff_stress_scaled_0b' response 'max_eff_stress_20000b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_1b' type weighted
  composite 'Max_eff_stress_scaled_1b' response 'max_eff_stress_20001b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_1a' type weighted
  composite 'Max_eff_stress_scaled_1a' response 'max_eff_stress_20001a' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_2a' type weighted
  composite 'Max_eff_stress_scaled_2a' response 'max_eff_stress_20002a' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_2b' type weighted
  composite 'Max_eff_stress_scaled_2b' response 'max_eff_stress_20002b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_3b' type weighted
  composite 'Max_eff_stress_scaled_3b' response 'max_eff_stress_20003b' 1 scale 1e+07
 composite 'Max_eff_stress_scaled_3a' type weighted
  composite 'Max_eff_stress_scaled_3a' response 'max_eff_stress_20003a' 1 scale 1e+07
$
$ COMPOSITE EXPRESSIONS
```

```
$
 composite 'Baffle_Mass' {10*(7830*baffle_T*0.4*(baffle_H+2*side_baffle_W))}
$
$ OBJECTIVE FUNCTIONS
$
 objectives 2
 objective 'TDV' 1
 objective 'Baffle_Mass' 1
$
$ CONSTRAINT DEFINITIONS
$
 constraints 15
  move
 constraint 'SB_side_constraint'
  strict
  lower bound constraint 'SB_side_constraint' 10
  slack
 constraint 'SB_mid_constaint'
  strict
  upper bound constraint 'SB_mid_constaint' 190
 constraint 'MB_up_constaint'
  slack
  strict
  upper bound constraint 'MB_up_constaint' 340
 constraint 'MB_low_constraint'
  lower bound constraint 'MB_low_constraint' 40
  slack
  stay
 constraint 'baffle_length'
  lower bound constraint 'baffle_length' 0
  upper bound constraint 'baffle_length' 540
  move
 constraint 'baffle_upper'
  strict
  upper bound constraint 'baffle_upper' 340
 constraint 'baffle_lower'
  lower bound constraint 'baffle_lower' 20
  slack
  stay
 constraint 'Max_eff_stress_scaled_0a'
  strict
  upper bound constraint 'Max_eff_stress_scaled_0a' 20
 constraint 'Max_eff_stress_scaled_0b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_0b' 20
 constraint 'Max_eff_stress_scaled_1b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_1b' 20
 constraint 'Max_eff_stress_scaled_1a'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_1a' 20
 constraint 'Max_eff_stress_scaled_2a'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_2a' 20
 constraint 'Max_eff_stress_scaled_2b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_2b' 20
 constraint 'Max_eff_stress_scaled_3b'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_3b' 20
 constraint 'Max_eff_stress_scaled_3a'
  slack
  strict
  upper bound constraint 'Max_eff_stress_scaled_3a' 20
$
$ JOB INFO
iterate param design 0.01
 iterate param objective 0.01
 iterate param stoppingtype and
 iterate 15
STOP
```