

University of Strathclyde

Department of Mechanical & Aerospace Engineering

Weir Advanced Research Centre

Modelling Cavitation in Centrifugal Pumps Using OpenFOAM

by

David Andrew Smith

A thesis presented in fulfilment of the requirements for the degree of

Master of Philosophy


2016

Declaration of Authenticity and Author's Rights

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

The work undertaken and presented within this thesis was funded by The Weir Group.

Signed: 

Date: 30/03/16

Acknowledgements

I would like to thank everyone who has made it possible for me to complete this thesis.

I am grateful to my supervisor, Dr Matt Stickland, firstly for agreeing to undertake this role but also for his support and continued patience over the duration of this work.

I am extremely grateful to The Weir Group who provided the development opportunity within the Weir Advanced Research Centre and financial support required, with particular thanks to David Manson and Graeme Kentley.

An additional thank you must also go to friends and staff at the University of Strathclyde as well as Dr William Nicholls and Dougal White for their support in numerous ways throughout this work.

Special thanks go to my family who have continued to support me over the years and believed in me through my studies. To Pam, I thank you for your continued support and patience, keeping me going through the difficult times and putting up with me the rest of the time.

Abstract

Whilst there continues to be significant development in high performance computing technology, the ever-increasing licensing costs associated with commercial computational fluid dynamics (CFD) software continues to prohibit the exploitation of such advancements within the engineering industry to the desired extent. In order to address this issue, open source software such as OpenFOAM is being developed by user communities and tailored to their own applications both in academia and commercial organisations alike.

This thesis investigates the applicability of the CFD code OpenFOAM for modelling rotor-stator turbomachinery simulations as an alternative to the commercial offerings currently being utilised by the Weir Group. More specifically, a Weir Warman AH 8/6 horizontal centrifugal pump, with a best efficiency point of 235 l/s at 53.8 m generated head and a rotational speed of 1100 rpm, is investigated as a test case.

Fully 3D transient single-phase simulations of the pump conducted in OpenFOAM examining the sensitivity of the solver to a variety of set up conditions show that the results are comparable to those obtained through the commercial solver ANSYS CFX.

In addition to the single-phase studies, the thesis also focuses on the problematic and costly phenomenon of cavitation within centrifugal pumps, with a review of cavitation and cavitation erosion modelling techniques being undertaken. Transient two-phase flow studies were subsequently conducted on the Weir Warman pump in both OpenFOAM and ANSYS CFX, however at this time realistic results were only obtained using the ANSYS CFX solver, with further work required in order to utilise the OpenFOAM code for such applications.

A further output from the work undertaken is a user manual for performing single-phase centrifugal pump simulations in OpenFOAM aimed at engineers involved in hydraulic design activities across the Weir Group.

Contents

CONTENTS	5
LIST OF SYMBOLS	8
LIST OF ACRONYMS	13
CHAPTER 1	14
1. INTRODUCTION	14
1.1 Modelling Cavitation	15
1.2 Computational Fluid Dynamics in Turbomachinery Design	15
1.3 OpenFOAM	16
1.4 Thesis Objectives	17
1.5 Thesis Layout	18
CHAPTER 2	20
2. CAVITATION AND MODELLING TECHNIQUES	20
2.1 What is Cavitation?	20
2.2 Cavitation in Centrifugal Pumps	22
2.2.1 Main Forms of Vapour Cavities	24
2.3 Predicting Cavitation: Cavitation Parameters	26
2.3.1 Cavitation Number	26
2.3.2 Net Positive Suction Head (NPSH)	27
2.3.3 Thoma Cavitation Number	29
2.4 Cavitation Nuclei and Modelling Bubble Dynamics using the Rayleigh-Plesset Equation	30
2.4.1 Bubble Dynamics Basics	32
2.5 Cavitation Modelling Techniques	35
2.5.1 Interface Tracking Models	35
2.5.2 Single-fluid Homogeneous Two-Phase Mixture Models	37
2.5.3 'Hybrid' Models	52
2.5.4 Fully Two-Phase Mixture Models	53
2.6 Conclusions	56
CHAPTER 3	58
3. CAVITATION EROSION MODELLING	58
3.1 Cavitation Erosion	58
3.2 Energy Considerations in Cavitation Erosion	59
3.3 Cavitation Erosion Mechanisms	60

3.3.1	Collapse and Rebound of a Spherical Bubble	60
3.3.2	Micro-jet	61
3.3.3	Collective Collapse	62
3.3.4	Cavitating Vortices	63
3.4	Predicting Cavitation Erosion	64
3.4.1	Empirical Methods	64
3.4.2	Cavitation Erosion Models	64
3.5	Complete Quantitative and Qualitative Modelling in CFD	74
3.5.1	Dular and Coutier-Delgosha	75
3.6	Conclusions	78
CHAPTER 4		79
4.	CFD METHODOLOGY	79
4.1	General Transport Equation	79
4.2	<i>k-ω</i> SST Turbulence Model	80
4.2.1	OpenFOAM Cases	82
4.2.2	Dimensional Units	83
4.2.3	Numerical Schemes	84
4.2.4	Time Discretisation	84
4.2.5	Convection Discretisation	87
4.2.6	Arbitrary Mesh Interface (AMI)	88
4.2.7	Solvers and Utilities	88
4.3	Centrifugal Pump Case Study	92
4.3.1	Computational Mesh	94
4.3.2	Boundary Conditions	96
4.3.3	Post-processing in ANSYS CFD-Post	99
4.4	Conclusions	102
CHAPTER 5		103
5.	SINGLE-PHASE MODELLING	103
5.1	Frozen Rotor Simulation	103
5.2	Transient Simulations	109
5.2.1	Setup	109
5.2.2	Solver Capability	110
5.2.3	Number of Pressure Correcting Loops (nOuterCorrectors)	118
5.2.4	Time Discretisation Scheme	121
5.2.5	Velocity Convection Scheme	126
5.2.6	Comparisons between OpenFOAM and ANSYS CFX	130

5.3	Conclusions	144
CHAPTER 6		146
6. TWO-PHASE MODELLING		146
6.1	NACA66 Hydrofoil	147
6.2	OpenFOAM Tutorial of Propeller	151
6.3	Warman 8/6 AH Pump	155
6.3.1	ANSYS CFX	155
6.3.2	OpenFOAM	167
6.3.3	OpenFOAM-2.1.x	167
6.3.4	OpenFOAM-2.3.x	171
6.4	Conclusions	171
CHAPTER 7		173
7. CONCLUSIONS AND FUTURE WORK		173
7.1	Summary	173
7.2	Conclusions	175
7.3	Future Work	175
8. REFERENCES		177
APPENDIX A - SUPPLEMENTARY DATA		182
APPENDIX B – TURBULENCE COEFFICIENTS		236
APPENDIX C – OPENFOAM USER GUIDE		237

List of Symbols

Symbol	Description
A	Turbulence Model Coefficient
A_i	Cavitation Model Coefficient, Interfacial Area Concentration
A_1	Inlet Area
A_2	Outlet Area
a	Camber Distribution
a_{min}	Minimum Speed of Sound
B	Turbulence Model Coefficient
b	Density Ratio
C	Fluid Parameter Function of C_0, L^*, U^*
c	Sonic Velocity of Liquid, Chord Length
C_{dest}	Coefficient of Destruction of Vapour
C_{prod}	Coefficient of Production of Vapour
C_0	Empirical Dimensional Parameter
C_1	Constant for Mass Transfer Blending Function
C_2	Constant for Mass Transfer Blending Function
D_1	Inlet Diameter
D_2	Outlet Diameter
d_{pit}	Maximum Depth of the Pit
E_c	Potential Energy Contained Within a Shed Collapsing Cavity
E_s	Damage Threshold Energy
f	Mass Vapour Fraction
f_g	Local Void Fraction
g	Gravitational Acceleration
H	Head

H_{opt}	Head at Best Efficiency Volume Flow Rate (Q_{opt})
H_{NC}	Non-Cavitating Generated Head
h	Reference Trajectory
h_k	Phase Averaged Enthalpy for Phase k
h_{ki}	Interfacial-Averaged Enthalpy
h_b	Bubble Layer Thickness
h_e	Effective Layer Thickness
I	Turbulence Intensity
K	Polytropic Coefficient for Gas Inside Bubble
k	Turbulent Kinetic Energy
L, L^*	Characteristic Length
\underline{M}_k	Interfacial Momentum Transfer per Unit Volume and Time
\dot{m}_i^k	Evaporation Term (Kubota Model)
\dot{m}_v^k	Condensation Term (Kubota Model)
\dot{m}_i^s	Evaporation Term (Senocak and Shyy Model)
\dot{m}_v^s	Condensation Term (Senocak and Shyy Model)
\dot{m}^+	Liquid-Vapour Evaporation Rate
\dot{m}^-	Liquid-Vapour Condensation Rate
n	Bubble Number, Rotational Speed (rpm)
n_0	Nuclei Concentration per Unit Volume of Pure Liquid
$NPSH_A$	Net Positive Suction Head Available
$NPSH_{BD}$	Breakdown Net Positive Suction Head
$NPSH_R$	Net Positive Suction Head Required
P, p	Pressure
p_c	Critical Pressure
p_{def}	Deformation Pressure
p_g	Gas Pressure
p_{g0}	Initial Gas Pressure

p_{min}	Minimum Pressure in Flow
p_{pot}^{mat}	Flow Aggressiveness Potential Power
p_{ref}	Reference Pressure
p_{stat}	Static Pressure
p_{tot}	Total Pressure
p_{waves}^{mat}	Pressure Wave Power
p_v	Vapour Pressure of Working Fluid
p_y	Material Yield Stress
p_{∞}	Applied Pressure
Q	Volume Flow Rate (l/s)
Q_{opt}	Best Efficiency Volume Flow Rate (l/s)
$q''_{ki}A_i$	Interfacial Transfer of Heat
R	Bubble Radius
R_C	Bubble Critical Radius, Condensation Rate
R_e	Evaporation Rate
r_{jet}	Radius of Jet
R_0	Initial Bubble Radius
S	Surface Tension Coefficient, Unit Surface Area
s	Grey Level, Chord Span
T	Temperature
t	Time
t_{def}	Time for Impact Signal to Travel Full Radius of the Jet
t_{∞}	Mean Time Flow Scale
U	Fluid Velocity
U_{ref}	Reference Flow Velocity
\vec{u}	Fluid Velocity Vector
U^*	Characteristic Velocity
U_{∞}	Reference Velocity

\vec{V}, \mathbf{v}	Velocity Vector
V_{cell}	Volume of Cell
V_{ch}	Characteristic Velocity
v_{crit}	Critical Micro-Jet Impact Velocity
V_d	Volume Damage Rate
v_{jet}	Micro-jet Impact Velocity
\underline{V}_k	Velocity of void fraction
V_l	Volume of Liquid
V_v, V_{vap}	Volume of Vapour
V_p	Control Volume
y^+	Dimensionless Wall Distance
α	Volume Vapour Fraction
α_a	Angle of Attack, degrees
α_k	Void Fraction
α_{ng}	Volume Fraction of Non-Condensable Gas
β	Turbulence Preconditioning Parameter
ε	Turbulence Dissipation Rate
Γ	Diffusion Coefficient, Effective Exchange Coefficient
Γ_k	Interfacial Mass Transfer per Unit Volume and Time
γ	Non-dimensional Distance from Bubble Centre to Surface
η^*	Efficiency Determined by Collapsing Spherical Vapour and Gas Bubbles
η^{**}	Energy Transfer Efficiency
μ	Dynamic Viscosity
μ_G	Vapour Dynamic Viscosity
μ_L	Liquid Dynamic Viscosity
$\mu_{m,t}$	Turbulent Viscosity
ρ	Fluid Density
ρ_k	Averaged Density

ρ_L	Liquid Density
ρ_l	Liquid Density
ρ_m	Mixture Density
ρ_v	Vapour Density
σ	Cavitation Number
σ_{BD}	Breakdown Cavitation Number
σ_C	Critical Cavitation Number
σ_I	Cavitation Inception Number
σ_{TH}	Thoma Cavitation Number
$\underline{\underline{\tau_k}}$	Molecular Stress Tensor
$\underline{\underline{\tau_k}}^T$	Turbulent Stress Tensor
ν	Fluid Kinematic Viscosity
ϕ	General Flux Variable
χ	Mass Transfer Blending Function
$\vec{\Omega}$	Angular Velocity Vector
ω	Specific Turbulence Dissipation
ω_i	Angular Velocity

All variables quoted above are in SI Units unless otherwise stated.

List of Acronyms

Acronym	Meaning
BTF	Bubble Two-phase Flow
CFD	Computational Fluid Dynamics
GUI	Graphical User Interface
IDM	Interfacial Dynamics Model
MDBM	Modified Density Based Model
NACA	National Advisory Committee for Aeronautics
NCG	Non Condensable Gas
NPSH	Net Positive Suction Head
SIG	Special Interest Group
VOF	Volume of Fluid

Chapter 1

1. Introduction

Pumping systems account for nearly 20% of the World's energy demand [1]. In the United Kingdom, this figure is 13% [2]. Consider this in conjunction with the fact that energy costs can typically represent 90% of the total lifetime cost of owning a pump and it is clear that efficiency is key [1].

These statistics are the reason why so much emphasis is placed on the design of pumps from both hydraulic and mechanical perspectives; the more efficient the components are during operation, the less energy is consumed and the longer these components will last.

In terms of reliability and efficiency, the German Engineering Federation (VDMA) found that 80% of pump failures in the chemical and process industries were from dry running, gas containing liquids, externally excited vibrations, imbalance, wear of bearings and blockages. However, the most common issue was found to be cavitation, where vapour bubbles form and collapse within hydraulic components, the effects of which vary from small reductions in pump performance to complete catastrophic failure [3]. In both instances, this can result in lost production and, more importantly for operating companies, reduced profits and significant disruption to the surrounding plant.

1.1 Modelling Cavitation

With cavitation being such an important design and operational consideration, the prediction of potential problems associated with its existence in a system is extremely valuable to both pump manufacturers and their customers.

Throughout the last fifty years there has been a large range of work surrounding the prediction of the occurrence of the vapour bubbles that cause cavitation from a quantitative perspective, using techniques such as volume fraction relationships, complex bubble dynamics theory and mass transfer concepts.

This has been taken a stage further in recent years by attempting to simulate the interaction between the collapsing vapour bubbles and the material surface of relevant components. The aim in this instance is to predict the rate of degradation of the components based on local conditions and therefore estimate the overall lifetime of the unit.

1.2 Computational Fluid Dynamics in Turbomachinery Design

A powerful tool in the field of turbomachinery, computational fluid dynamics (CFD), is used frequently for the hydraulic design of new equipment. It helps in the identification of issues associated with existing units as well as the upgrade/re-rating of components to satisfy new design or operational requirements.

In terms of centrifugal pumps, CFD software can allow companies to predict the performance of particular hydraulic components such as impellers and diffusers as well as overall pump performance. This is done not only by predicting standard performance parameters such as head, power and efficiency but by also allowing detailed investigation

of fluid properties such as velocity, pressure and turbulence at any point in order to highlight areas of concern.

In fact, to aid the design process further and increase accuracy, cavitation models are now standard on most commercial CFD software offerings, allowing the qualitative prediction of regions at risk and an opportunity to redesign particular aspects of the component geometry prior to manufacture and physical testing.

1.3 OpenFOAM

Whilst the methods commonly used for the design and upgrade of hydraulic components can be significantly enhanced using CFD software packages, the associated licenses required to run simulations and perform analyses are becoming increasingly expensive. This often prohibits companies from improving their design processes sufficiently if at all.

In order to overcome these licensing issues, some users are now investigating the potential of using alternatives that are free to use such as OpenFOAM to either compliment or replace aspects of their hydraulic design processes.

Although OpenFOAM has clear benefits in terms of cost, there are drawbacks associated with using the software. Perhaps most prohibitive is the lack of a graphical user interface (GUI), meaning that all simulations must be set up and run using command line driven coding, an approach that takes time to become accustomed to. However, once the user is experienced in this approach, the modification of cases is much more straightforward due to the modular approach of the code.

The other main drawback in terms of turbomachinery is associated with the extremely limited amount of relevant work in the public domain. Without a template on which to

build and the significant investment of time required to learn the new system code, many users and companies are put off.

1.4 Thesis Objectives

With the issue of increasingly prohibitive licensing costs associated with commercial codes, The Weir Group has commissioned this work in order to establish whether or not OpenFOAM is powerful and accurate enough to be used to compliment aspects of the centrifugal pump hydraulic design processes, with the possibility of cavitation prediction also being investigated.

To achieve the overall objective of assessing the potential of using OpenFOAM, this thesis considers a centrifugal pump case study and compares results of both steady state and transient moving mesh single-phase analyses conducted in OpenFOAM against results using the commercial software package ANSYS CFX.

In order to take this a stage further and conduct an initial investigation into the potential of using OpenFOAM for cavitation prediction within transient moving mesh simulations, particular focus will firstly be paid to the theory of cavitation and relevant modelling techniques. Using this approach, the knowledge gained will be used to assess the software packages (OpenFOAM and ANSYS CFX) by conducting two-phase analyses of the centrifugal pump.

To ensure that the experience of using OpenFOAM for simulating centrifugal pumps is adequately captured for The Weir Group, a user manual for performing single-phase centrifugal pump simulations in OpenFOAM will be developed, aimed at engineers involved in hydraulic design activities across the business.

1.5 Thesis Layout

An introduction to the requirement of highly efficient turbomachinery systems and identification of cavitation as the most common cause of failure was given in Chapter 1. The significant limitations, associated with the licensing costs of current CFD modelling software, were also discussed and the requirement for investigating potential alternatives such as OpenFOAM was identified.

Chapter 2 introduces the theoretical concepts of cavitation, standard methods of measurement and presents an overview of the main quantitative modelling techniques currently available, discussing the particular advantages and drawbacks associated with each. This investigation of modelling techniques is taken a stage further in Chapter 3 by examining work in the field of cavitation erosion modelling in which quantitative prediction of cavitation from a material degradation perspective is sought.

Progressing onto CFD simulations; Chapter 4 introduces the OpenFOAM code, discussing the various solvers that will be used in the subsequent single-phase, transient and cavitating flow simulations as well as the utilities that allow the desired information to be tracked during the solving period. The centrifugal pump case study used in this work is also introduced.

Chapter 5 compares the single-phase modelling capability of OpenFOAM against commercial CFD software, ANSYS CFX, using the centrifugal pump case study in order to establish whether the code is a viable alternative whilst establishing the most effective solver setup conditions for such simulations.

Chapter 6 takes the work from Chapter 5 a stage further, in conjunction with the knowledge gained in Chapter 2, by performing initial investigations into the potential of

using the OpenFOAM software to model the occurrence of cavitation in the transient centrifugal pump case study.

Finally, Chapter 7 reviews the thesis and outline recommendations for future work.

As an additional resource *APPENDIX C – OpenFOAM User Guide* contains a user guide that provides a detailed procedure to enable the download and installation of the OpenFOAM software as well as a methodology for the construction and running of a turbomachinery simulation based on the work related to Chapter 5.

Chapter 2

2. Cavitation and Modelling Techniques

This chapter focuses on the theory of cavitation as well as the prediction and modelling of this phenomenon. The theoretical concept of cavitation is introduced in Section 2.1, with its occurrence in centrifugal pumps discussed in Section 2.2. Following this brief overview, Section 2.3 discusses the standard methods of predicting cavitation in industry and the concept of bubble dynamics is outlined in Section 2.4. The main focus of this chapter is contained in Section 2.5 where a review of the various methods of modelling the phenomenon of cavitation are discussed in turn, with concluding remarks contained in Section 2.6.

2.1 What is Cavitation?

Cavitation, a term derived from the Latin word *cavus*, can be defined as the formation of an empty space within a solid object or body or, more specifically, the appearance of a vapour phase in an initially homogenous liquid medium. It occurs in a variety of areas of interest from industry to nature, including rotating machinery such as centrifugal pumps.

The formation of a vapour phase is visible through the development of bubbles in the working fluid, occurring when a local pressure is below the liquid's saturating vapour pressure. This is nearly always accompanied by the production of gases previously dissolved in the liquid and results in a significant increase in the compressibility of the liquid-vapour mixture and thus severe alterations to the properties of the fluid [4].

The threshold at which cavitation occurs can be illustrated by a typical phase diagram, as shown in Figure 2.1, with vaporisation and therefore cavitation occurring when crossing the liquid-vapour line. Utilising this diagram, vaporisation by cavitation is often compared to the process of boiling. Whilst there are similarities between the two processes, it is important to differentiate between the two as the driving mechanism is different for each.

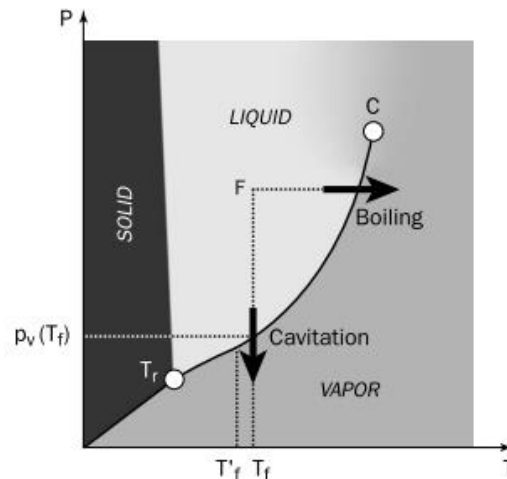


Figure 2.1 - Typical Phase Diagram [5]

Although it is relatively straightforward to impart uniform changes of pressure in a liquid, it is extremely difficult to change temperature in a similar manner. Brennan [6] discusses the differences between these processes in significant depth, highlighting the ‘complicating factors’ that occur in a cavitating flow and the temperature gradients and wall effects that occur in boiling liquid.

In more simple terms, the mechanism driving cavitation is the local pressure controlled by the flow dynamics, with only extremely small amounts of heat being required for significant vaporisation.

The term cavitation is used to describe the entire process of bubble formation, growth and subsequent collapse therefore it is beneficial to understand the fundamental processes involved.

Formation – Cavitation inception occurs and vapour bubbles are formed when the local static pressure becomes equal to or below the vapour pressure of the liquid at the operating temperature.

Growth – Assuming that the operating conditions remain constant, the existing bubbles grow in size and new bubbles form. These bubbles are carried in the liquid as it flows along or through the particular geometry.

Collapse – As these vapour bubbles move along the geometry, the pressure around them begins to increase until it is greater than the pressure inside the bubble, causing it to implode. The rupturing of this bubble by the surrounding fluid and the subsequent rush to fill the void can cause pitting if it occurs close enough to the material surface.

2.2 Cavitation in Centrifugal Pumps

Whilst cavitation occurs in a wide range of areas, this research is focussed on the particular area of centrifugal pumps and its relationship with the design and operation of such equipment in terms of performance and lifetime.

Cavitation in such turbomachinery occurs when the localised pressures on the impeller blades are sufficiently low enough to form a vapour. Whilst a small quantity of cavitation usually has no detrimental impact, severe amounts are problematic enough to warrant significant attention at both the design stage, focussing on the hydraulic design of the impeller, and in operation, in terms of the upstream conditions.

The most effective way of visualising the effects of cavitation in terms of performance is by using a typical head-flow curve on which generated head, H , is plotted against the volume flow rate, Q . When the pump is tested by the manufacturer, an excess of pressure at inlet will be available, eliminating the risk of cavitation. However, in industry, these ideal conditions are often not met due to the site process being different to the original design.

When a pump is being implemented into a system it has to overcome a level of resistance head that consists of two components; a static part, constant across the entire operational flow range of the pump and a frictional part associated with frictional losses which increases at higher flow rates. For a single pump, the point of operation will be positioned at the intersection between the pump performance curve and the system resistance curve, such as point A in Figure 2.2.

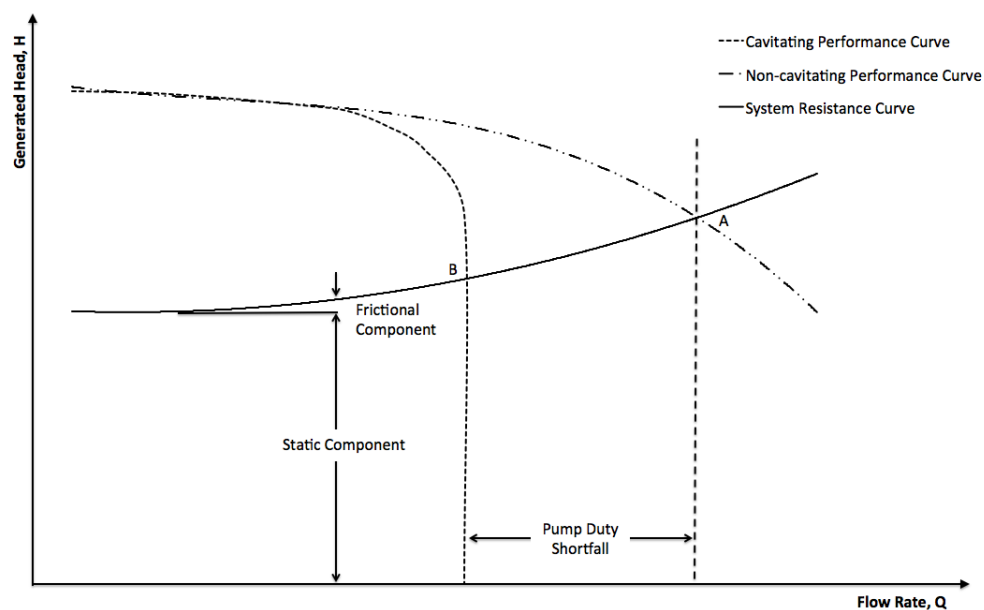


Figure 2.2 - Impact of Cavitation on Pump Performance Curve

However, if the pump is cavitating, the point of operation on the same system curve would be in the region of point B, significantly reducing the capacity of the pump. This can be attributed to the bubbles formed during the cavitation process reducing the volume

available for fluid in the pump. The generated head is also reduced, attributed to the collapsing of the bubbles during which the surrounding liquid rushes to fill the cavities formed, increasing its velocity and expending more energy in the process [7].

Whilst this is a considerable issue related to the development of cavitation in a system, there are many other issues which have been discussed at great length [8] but are not discussed in detail here, including but not limited to:

- Cavitation Surging (hydrodynamically and thermodynamically induced)
- Sound/vibration
- Flow instabilities
- Damage to components and mechanical deformations
- Cavitation erosion

2.2.1 Main Forms of Vapour Cavities

In terms of centrifugal pumps, there are three main types of cavitation that designers should be familiar with.

2.2.1.1 Travelling Bubble Cavitation

The configuration illustrated in Figure 2.3 is commonly called travelling bubble cavitation since the bubbles are conveyed by the flow over the wall. They grow in the low-pressure region before collapsing in the region of pressure recovery downstream.

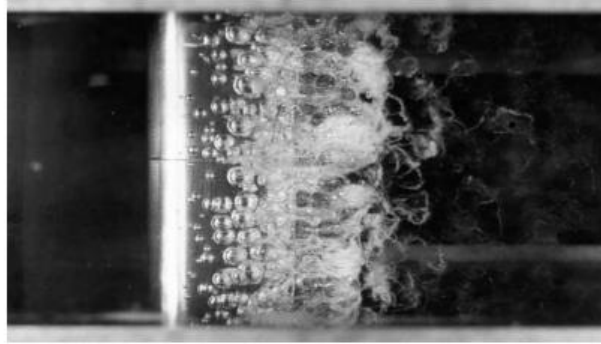


Figure 2.3 – Bubble Cavitation on a Hydrodynamic Test Body [6]

2.2.1.2 Attached Cavitation

If the pressure is reduced further, the travelling bubbles merge to form attached cavities, such as in Figure 2.4, often referred to as blade cavitation in terms of pumps. Unlike travelling bubble cavitation, attached cavitation is connected to the impeller itself until collapsing on the impeller surface downstream where the pressure is increased.

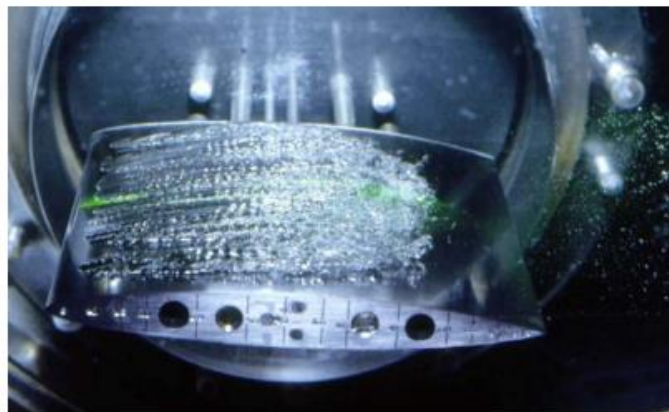


Figure 2.4 - Leading Edge Cavity on the Suction Side of a Foil in a Hydrodynamic Tunnel [5]

2.2.1.3 Vortex Cavitation

In the configuration of a 3D hydrofoil, the pressure difference between the pressure and suction sides generates a secondary flow that goes round the tip, giving birth to a vortex attached to the tip, seen in Figure 2.5.

Due to high rotational velocities, the static pressure within the vortex core is lower than the surrounding fluid, resulting in a minimum pressure, and therefore the highest level of cavitation, at the vortex centre. This type of cavitation is also often observed on marine propellers as shown in Figure 2.5.



Figure 2.5 - Vortex Cavitation Generated by a Three-Dimensional Hydrofoil [5]

2.3 Predicting Cavitation: Cavitation Parameters

Before discussing cavitation models in detail, it is useful to understand the three common parameters used to predict the occurrence of cavitation in relation to centrifugal pumps; cavitation number, σ , net positive suction head (NPSH) and Thoma cavitation number.

2.3.1 Cavitation Number

This parameter defines the occurrence of cavitation as being when the minimum pressure in the flow, p_{min} , is equal to or lower than the vapour pressure of the working fluid, p_v :

$$p_{min} \leq p_v \quad (2.1)$$

The non-dimensional cavitation coefficient, often referred to as the 'cavitation number', σ , is given in (2.2).

$$\sigma = \frac{p_{ref} - p_v}{\frac{1}{2}\rho U_{ref}^2} \quad (2.2)$$

where p_{ref} and U_{ref} are selected values for pressure and velocity in the flow and ρ is the fluid density. For centrifugal pumps, p_{ref} is typically the upstream static pressure and U_{ref} is commonly taken as the mean velocity of the inlet flow.

One of the main advantages of the cavitation number is that it can be scaled and applied to different operating conditions and is typically represented by a head drop curve as seen in Figure 2.6. This allows the important parameters of cavitation inception number, σ_I , critical cavitation number, σ_C , and breakdown cavitation number σ_{BD} , to be highlighted (for a constant rotational speed and flow rate).

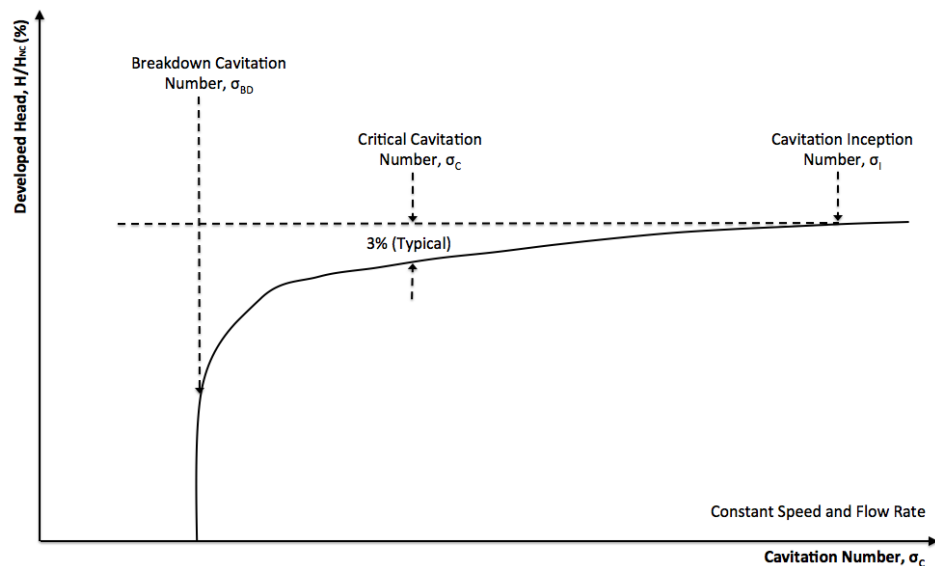


Figure 2.6 – Typical Head Drop Curve

2.3.2 Net Positive Suction Head (NPSH)

Whilst the cavitation number, σ , is perhaps the most fundamental parameter in cavitation prediction, it is often more convenient to analyse turbomachinery in terms of net positive suction head (NPSH), the excess of inlet total head over the head equivalent of the vapour pressure of the pumped liquid [8].

When using this parameter, there are two important values:

- $NPSH_A$ – NPSH available at the pump inlet
- $NPSH_R$ – NPSH required at the pump inlet in order to operate efficiently without the issues associated with the existence of cavitation

Based on these definitions, it can be stated that in order for the pump to run efficiently, the value of $NPSH_A$ must be greater than or equal to the $NPSH_R$ as in (2.3).

$$NPSH_A \geq NPSH_R \quad (2.3)$$

$NPSH_R$ is very much a design parameter which is dependent on pump speed, impeller diameter, flow velocity and flow approach angle, whereas the available $NPSH_A$ is a process-dependant property governed by the total pressure and velocity of the liquid in the intake pipe[9].

The significance of available suction head on pump performance is highlighted in Figure 2.7, which plots the generated head, developed, H , as a percentage of non-cavitating generated head, H_{NC} against the net positive suction head available. It can be seen that as the available net positive suction head decreases, the developed head drops gradually, before a complete breakdown occurs at $NPSH_{BD}$ when the impeller eye is completely vapour locked. In relation to real-world applications, the Hydraulic Standards Institute defines cavitation as a 3% drop in head developed across the pump.

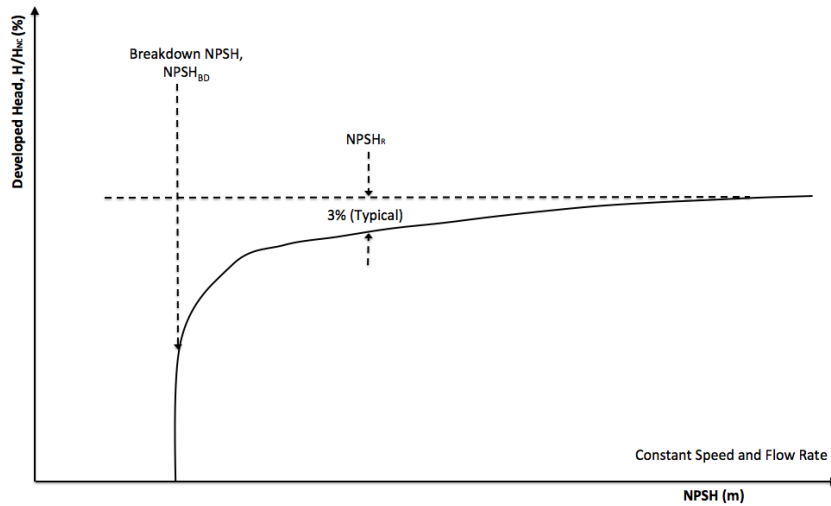


Figure 2.7 – Another Typical Head Drop Curve, This Time Against NPSH

In terms of predicting the onset of cavitation, the following relationship in (2.4) is used:

$$NPSH_R = \frac{p_{tot} - p_v}{\rho g} \quad (2.4)$$

where p_v is the vapour pressure of the fluid and p_{tot} is the total suction pressure upstream of the location of cavitation, defined in (2.5).

$$p_{i(tot)} = p_{stat} + \frac{1}{2}\rho U_{ref}^2 \quad (2.5)$$

where p_{stat} and U_{ref} refer to upstream static pressure and inlet velocity respectively.

Whilst useful, it must be noted that this equation assumes that the liquid cannot withstand tension and that cavitation bubbles appear when the static pressure p_{stat} reaches the vapour pressure, p_v .

2.3.3 Thoma Cavitation Number

Another non-dimensional parameter which attempts to predict the formation of cavitation is the Thoma Number, described in (2.6).

$$\sigma_{TH} = \frac{NPSH_A}{H} \quad (2.6)$$

where H is the total head generated by the pump.

Its relevance to pumps, however, has been severely questioned. Schiavello and Visser [4] refer to it as an ‘archaic’ parameter, Yedidiah [10] describes how it is not appropriate for use in pumps due to a lack of a direct connection between the developed head and its suction capabilities and Al-Chalaby [9] states that the Thoma number should be used very carefully for pumps and at the best efficiency point (BEP) only.

2.4 Cavitation Nuclei and Modelling Bubble Dynamics using the Rayleigh-Plesset Equation

Cavitation is usually initiated from microscopic nuclei carried by the flow [11]. These nuclei are points of weakness in the liquid that grow to form larger cavities when exposed to low-pressure regions.

The most widely used method of modelling such nuclei is the microbubble of a few microns in diameter [11]. These bubbles are considered to be spherical and contain a gaseous mixture made consisting of the vapour of the liquid in question and, in certain cases, non-condensable gases (NCGs) such as oxygen and nitrogen in water.

In order to understand cavitation, an understanding of the dynamics of such bubbles is clearly of great importance. The Rayleigh-Plesset equation shown in (2.7) provides this and is the concept upon which several modelling techniques are based.

$$\rho \left[R\ddot{R} + \frac{3}{2}R^2 \right] = [p_v - p_\infty(t)] + p_{g0} \left(\frac{R_0}{R} \right)^{3K} - \frac{2S}{R} - 4\mu \frac{\dot{R}}{R} \quad (2.7)$$

where \dot{R} and \ddot{R} are the first and second order derivatives of the bubble radius with respect to time and R_0 is the initial bubble radius.

The terms of the right hand side are outlined below:

- $[p_v - p_\infty(t)]$ - the driving term for the bubbles evolution, tracking how close the applied pressure within the fluid is in relation to the vapour pressure.
- $p_{g0} \left(\frac{R_0}{R}\right)^{3K}$ - the contribution of any non-condensable gases present and is based on the assumption that the mass of the NCG inside the bubbles remains constant during its evolution and that this constant mass of gas is assumed to follow a polytropic thermodynamic behaviour characterised by a polytropic coefficient, K .
- $\frac{2S}{R}$ - the contribution of surface tension, with S representing the surface tension coefficient. As R is the denominator it is only important for small radii.
- $4\mu \frac{\dot{R}}{R}$ - accounts for the effect of the dynamic viscosity of the liquid, μ . This is also considered to be relevant only to small radii as the dissipation due to viscosity is proportional to the bubble deformation rate, \dot{R} , and inversely proportional to the bubble radius.

Simplifications of this equation are often made for larger bubbles, such as in (2.8) where the effects of NCGs, surface tension and viscosity are removed.

$$\rho \left[R\ddot{R} + \frac{3}{2}R^2 \right] = [p_v - p_\infty(t)] \quad (2.8)$$

In addition, if the applied pressure, p_∞ , is constant, then the Rayleigh-Plesset equation can be integrated, supplying the bubble interface velocity from (2.9).

$$\dot{R}^2 = \frac{2}{3} \frac{p_v - p_\infty}{\rho} \left[1 - \left(\frac{R_0}{R} \right)^3 \right] \quad (2.9)$$

2.4.1 Bubble Dynamics Basics

2.4.1.1 Bubble Equilibrium

It is useful to consider the equilibrium of the microbubble using the Rayleigh-Plesset equation. By setting all time derivatives to zero and assuming that the gas transformation is isothermal ($k=1$), the equilibrium condition shown in Figure 2.8 and (2.10) is established:

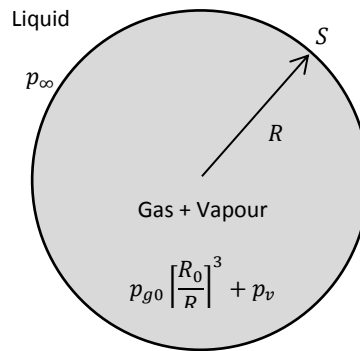


Figure 2.8 – Illustration of Forces on Bubble

$$p_{g0} \left[\frac{R_0}{R} \right]^3 + p_v = p_\infty + \frac{2S}{R} \quad (2.10)$$

From this, it can be seen that the pressure difference between the inside and outside of the bubble is due to the surface tension, with the relationship between p_∞ and the bubble radius, R , being shown in Figure 2.9.

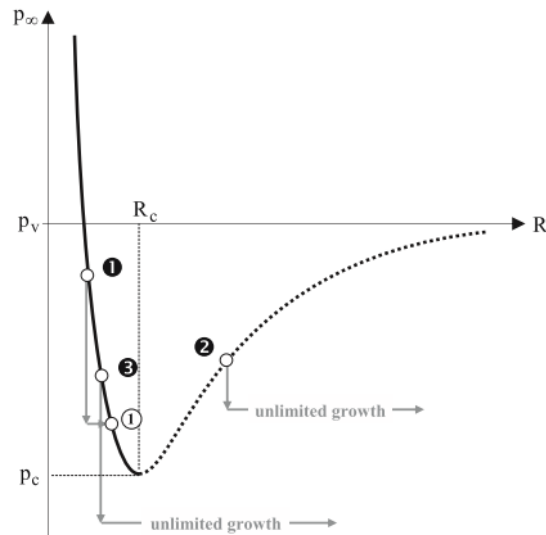


Figure 2.9 - Relationship Between p_∞ and R [11]

It is important to note here that this equilibrium is not always stable and there is a minimum point on the equilibrium curve seen in Figure 11, represented by a 'critical' pressure and radius, p_c and R_c , given by (2.11) and (2.12).

$$R_c = \sqrt{\frac{3p_{g0}R_0^3}{2S}} \quad (2.11)$$

$$p_c = p_v - \frac{4S}{3R_c} \quad (2.12)$$

where these critical values depend on the surface tension, S , and the group of parameters $p_{g0}R_0^3$.

In order to understand where the equilibrium is stable, Franc's example of three different nuclei exposed to a pressure lower than the vapour pressure can be used [11]:

- Nucleus 1 – The pressure is lowered to a point where the new equilibrium is reached at point 1', illustrating that this descending part of the equilibrium curve is stable.

- Nucleus 2 – The bubble grows indefinitely without crossing the curve, resulting in an unstable equilibrium.
- Nucleus 3 – The pressure is reduced to below the critical pressure, p_c , resulting in the bubble again growing indefinitely without reaching equilibrium.

It is clear from this that the critical pressure can be used as a threshold above which the microbubble will explode, turning into a macroscopic cavitation bubble.

2.4.1.2 Bubble Growth and Collapse

If the effects of NCGs, surface tension and viscosity are assumed negligible then the bubble growth and collapse rates can be determined using the simplified Rayleigh-Plesset equation.

If $p_\infty < p_v$, the bubble will grow according (2.13).

$$\dot{R} \cong \sqrt{\frac{2}{3} \frac{p_v - p_\infty}{\rho}} \quad (2.13)$$

It is interesting to note here that when the cavitation bubble becomes three times larger than its initial nucleus, the percentage error of this equation is below 2% [11].

Also, if $p_\infty > p_v$, the bubble will collapse according to (2.14).

$$\dot{R} \cong -\sqrt{\frac{2}{3} \frac{p_\infty - p_v}{\rho} \left[\left(\frac{R_0}{R} \right)^3 - 1 \right]} \quad (2.14)$$

As will be seen in the later, the Rayleigh-Plesset equation is utilised in numerous cavitation models.

2.5 Cavitation Modelling Techniques

The cavitating flow seen in engineering systems involve complex interactions between the liquid and vapour phases, not to mention its turbulent nature. Significant research in modelling these interactions has led to a range of models being developed to simulate multi-phase flows with phase transition which can be categorised depending on the technique used [12].

The most basic interface-tracking models consider each phase separately and are used for simpler problems. Mixture models use the theory of a homogenous mixture to generate transfer terms representing transition between phases. Completely two-phase models are more complex and involve solving the relevant balance equations for each phase present in the fluid. Hybrid methods are also seen, using existing models together in order to model particular aspects of the cavitation phenomenon effectively. Examples of work using these methods will be discussed in turn.

2.5.1 Interface Tracking Models

The most simplified numerical models of cavitation are referred to as interface tracking models and treat the computational domains with individual phases modelled separately, using time-wise grid regeneration according to the cavity shape [13].

A variety of work has been conducted within this category, with potential flow models being developed as far back as 1969, Brennen [14] and Furness and Hutton [15]. More recently, Euler and Navier-Stokes based models have been developed and include the well-referenced works of Chen and Heister [16] and Deshpande et al [17]. Some work has also been conducted specifically for cavitating flows in pumps and propeller blades such as

Kueny et al. [18], Peallat and Pellone [19] and Kinnas and Fine [20]. These interface-tracking models are based upon three basic assumptions:

1. The cavity surface (or the interface) is a free surface
2. The pressure inside the cavity is equal to the local vapour pressure of the surrounding fluid.
3. The rear part of the cavitation bubble is approximated by using a wake model, where the pressure is no longer equal to the vapour pressure

Figure 2.10 is the schematic of a cavitation bubble on a headform/cylindrical body used by Chen et al. to illustrate their interface-tracking model. Point A is the point of inception, Point B links the 'forebody' (A-B) and 'afterbody' (wake region) (B-C) of the cavity and Point C is the end point of the cavitation.

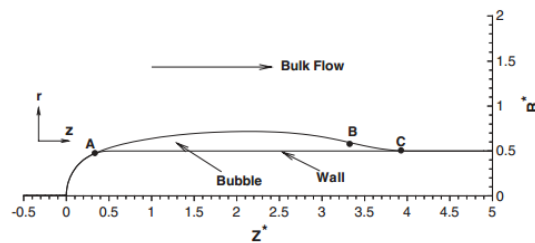


Figure 2.10 - Schematic of Chen et al.'s Cavitation Model

Point A is located by searching for the point on the initially non-cavitating wall where the pressure is minimal and drops below the local vapour pressure.

The forebody (A-B) of the cavity is determined iteratively using a cavity shape-updating scheme, with the final shape being attained when the pressure on the cavity surface is equal to the vapour pressure of the surrounding fluid.

The two-phase turbulent wake region (B-C) is, for simplicity, modelled using a cubic polynomial function as the structures and mechanisms existing in this region are unknown.

This 'afterbody' begins at Point B and is located at the point on the cavity surface where the local height is, for example [21], half of the maximum height of the cavity and serves as a smooth link between the cavity itself and the local wall surface. The length of this closure region as well as the location where the cavity is attached to the wall (Point C) is automatically determined by a wake model.

Whilst these simplified models are able to predict the point of detachment from the wall and wall pressure distribution reasonably accurately, they have a variety of issues associated with them. Firstly, they are limited to modelling steady sheet cavitation in a quasi-steady state and cannot complete the vapour cloud shedding process. No turbulence model is utilised, no 'true cavitation model' is involved in a strict sense [22] and they are limited to two-dimensional simulations. This results in models only capable of predicting the mean shape of cavities, unable to adapt to 'unsteady' cavitation phenomena.

2.5.2 Single-fluid Homogeneous Two-Phase Mixture Models

In order to simulate the unsteady nature of cavitation a different method is required.

A common approach is to treat the cavitating fluid as a homogeneous two-phase mixture consisting of liquid and vapour and solving a single set of mass and momentum equations. By neglecting the slip between the liquid and vapour phases, the density of the single-phase mixture can vary from pure liquid to pure vapour.

Many authors [22] [23], highlight the main issue as how to define the density of this mixture, with the simultaneous treatment of very different flow conditions - pure liquid, pure vapour and a transition region between the vapour and liquid. It is here where the main differences in approach occur.

The most simplified method is to use a ‘barotropic model’ which describes the mixture density in the transition zone between liquid and vapour by linking it to the local static pressure using a barotropic law, $\rho(P)$.

Another, more popular, approach is to estimate the properties of the mixture, including the density using liquid-vapour ratios based on volume fraction (α) or mass fraction (f) and either Rayleigh-Plesset bubble dynamics or empirical parameters.

The theoretical advantage of these models is that they can simulate a wide range of cavitation types including sheet, travelling and super-cavitation whilst including turbulence models.

2.5.2.1 Barotropic Models

The simple barotropic approach solves only one equation relating to the mixture itself, directly coupling the density with pressure, based on the general equation of state as in (2.15).

$$\rho = f(P, T) \quad (2.15)$$

As the cavitation process is assumed to be isothermal, the mixture density is considered to be a function of local pressure only, $\rho = f(P)$, allowing models to be based on three states as seen in (2.16) and Figure 2.11.

$$\rho = \begin{cases} \rho_l & \text{if } p > p_v + \Delta p \\ \rho_v & \text{if } p < p_v - \Delta p \\ \rho(p) & \text{if } p_v - \Delta p < p < p_v + \Delta p \end{cases} \quad (2.16)$$

where Δp is the half-width of the transition from vapour to liquid.

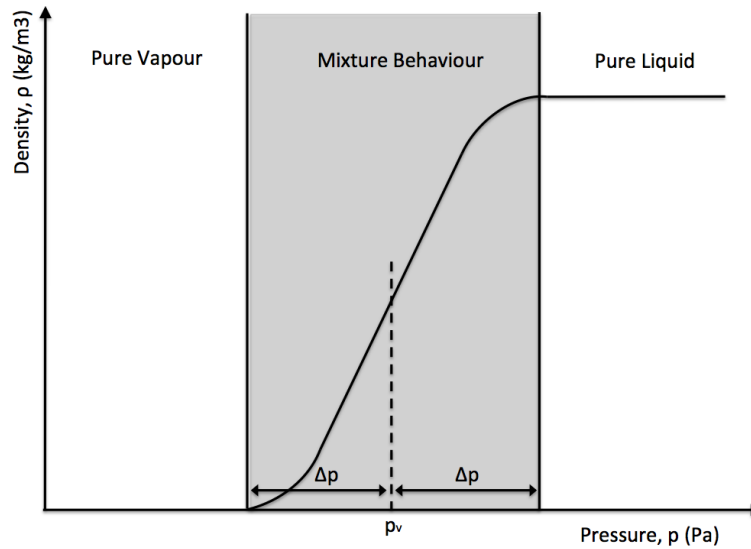


Figure 2.11 - Transition between Vapour and Liquid

These equations represent an incompressible state representing pure liquid, a compressible state representing the pure vapour and a transition component. It is this transitional component which is of interest and by which the models can be differentiated.

The barotropic approach was initially proposed by Delannoy and Kueny [24] who established that the compressibility of the vapour and liquid was only important during the final collapsing phase, representing the transition component by the sine law (2.17).

$$\rho = \rho_v + \frac{1}{a_{min}^2} \frac{2}{\pi} (p_l - p_v) \sin\left(\frac{p - p_v}{p_l - p_v} \frac{\pi}{2}\right) \quad (2.17)$$

where a_{min} is the minimum speed of sound in the mixture. Hoeijmakers et al. [25] also used a similar method.

A different technique was adopted by Chen and Heister [21] who derived a time and pressure dependent density as in (2.18).

$$\frac{d\rho}{dt} = C(P - P_v) \quad (2.18)$$

where $C = C_0 L^* U^*$ (2.19)

with C_0 being defined as an empirical, dimensional parameter for a given liquid and L^* and U^* representing characteristic length and velocity respectively.

Yet another approach was taken by Song and He who utilised a fifth order polynomial, shown in (2.20), to describe the cavitation process [26].

$$\rho = \sum_{i=0}^5 A_i P^i \quad (2.20)$$

where coefficients A_i were selected in order that the density decreased rapidly when the pressure dropped below the vapour pressure.

These methods lack a cavitation transport equation meaning they are unable to consider the convection and transport phenomenon of the cavitation bubbles and are therefore mainly suitable for attached cavities.

Dular [27] stated that, whilst these models show very good correlation with experimental data, the numerical algorithms lack robustness, leading to numerical instability and poor convergence. Hofmann [28] and Habr [29] also indicated that a significant adjustment to the density function is often required in order to get plausible or converged results.

2.5.2.2 Transport Models

To compensate for the issues associated with barotropic models, significant work has been conducted the using concepts of volume of fluid (VOF) or mass fraction in which a transport equation for either volume or mass fraction is solved, with source terms regulating the mass transfer between phases and the mixture density related directly to the relative fractions of liquid and vapour. Equations (2.21) and (2.22) show the relationships for the mixture density using the volume fraction and mass fraction approaches respectively.

$$\rho = \alpha\rho_v + (1 - \alpha)\rho_l \quad (2.21)$$

$$\frac{1}{\rho} = \frac{f}{\rho_v} + \frac{1 - f}{\rho_l} \quad (2.22)$$

where α is the vapour volume fraction, f is the vapour mass fraction and the subscripts v and l denote the vapour and liquid components respectively. Transport models also tend to incorporate turbulence modelling capability.

The additional transport equation for the vapour fraction parameter is shown in (2.23).

$$\frac{\partial \alpha \rho_v}{\partial t} + \nabla(\alpha \rho_v \vec{u}) = \dot{m}^+ + \dot{m}^- \quad (2.23)$$

where \dot{m}^+ and \dot{m}^- represent the liquid-vapour evaporation and condensation rates respectively.

The difference across this range of models is in determining how to represent these creation and destruction terms. Some consider the bubble dynamics to be the most important and use the Rayleigh-Plesset equation while others tune the behaviour based on experimental results specific to the case in question.

2.5.2.2.1 Rayleigh-Plesset Based Models

As mentioned, a common method is to couple the Rayleigh-Plesset equation to the flow solver in order to represent the dynamics of the bubbles contained in the mixture.

Kubota et al. [30] originally used the linear part of the Rayleigh-Plesset equation to simulate the evolution of the bubble radius as a function of surrounding pressure [27]. Since then, more complex models have been developed by Schnerr and Sauer [31] and Frobenius [32] that include quantities such as bubble number density and initial bubble

diameter. However, these are extremely difficult or impossible to determine due to their connection to specific setup conditions and varying impurities present within the liquid.

2.5.2.2.2 Kubota et al. [30]

Within this work, a new cavity model termed bubble two-phase flow (BTF) was developed, investigating the nonlinear interaction between viscous flow with large-scale vortices and microscopic cavitation bubble dynamics; considering the effects of bubble nuclei on cavitation inception and development and expressing unsteady characteristics of vortex cavitation [30].

The mixture is treated as two parts: macroscopic and microscopic. The macroscopic (local homogeneous) model treats the inside and outside of the cavity as a single continuum, regarding the cavity flow as a compressible viscous fluid whose density can vary greatly.

The governing equations of the macroscopic flow field are continuity and the Navier-Stokes conservation equation for momentum, in (2.24) and (2.25) respectively.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \quad (2.24)$$

$$\frac{\partial(\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v} \mathbf{v}) = -\nabla P + \frac{1}{Re} \mu \left\{ \nabla^2 \mathbf{v} + \frac{1}{3} \nabla(\nabla \cdot \mathbf{v}) \right\} \quad (2.25)$$

where P is the pressure in the mixture, μ is the viscosity of the mixture and Re is the Reynolds number. The liquid is assumed to be incompressible.

This model differs from others by assuming that the water-vapour mixture is actually replaced by a fluid of variable density, with the mass and momentum of the vapour being neglected, resulting in different equations to represent the density and viscosity of the mixture, seen in (2.26) and (2.27) respectively.

$$\rho = (1 - f_g)\rho_L \quad (2.26)$$

$$\mu = (1 - f_g)\mu_L + f_g\mu_G \quad (2.27)$$

where ρ_L is the density of the liquid, f_g is the local void fraction, μ_L is the liquid viscosity and μ_G is the vapour viscosity.

From a microscopic perspective, the cavitation is treated as bubble clusters by way of a mean field approximation approach, treating the cavity as a local homogeneous cluster of spherical bubbles, where bubble number density and a typical radius are assumed locally.

The local void fraction, f_g , used in the ‘macroscopic model’ is determined by coupling the bubble density with the bubble radius as in (2.28).

$$f_g = n \frac{4}{3}\pi R^3 \quad (0 < f_g < 1) \quad (2.28)$$

where n is the bubble number density and R is the typical bubble radius.

With the detail of the derivation of the model, and its associated assumptions in the literature, (2.29) shows the final form of Kubota et al’s locally homogeneous equation.

$$\begin{aligned} (1 + 2\pi\Delta r^2 n R) R \frac{D^2 R}{Dt^2} + \left(\frac{3}{2} + 4\pi\Delta r^2 n R \right) \left(\frac{DR}{Dt} \right)^2 + 2\pi\Delta r^2 \frac{Dn}{Dt} R^2 \frac{DR}{Dt} \\ = \frac{P_v - P}{\rho_L} \end{aligned} \quad (2.29)$$

By analysing the results, this cavity model is useful for investigating vortex cavitation characteristics, as it was able to clarify particularly complicated interactions such as those between large-scale vortices caused by separation as well as bubble dynamics. However, the convection of and distribution of the bubbles, variation of the viscosity of the mixture and the slip between bubbles and liquid have all been identified as parameters that could

be incorporated in the future to more accurately predict the behaviour of the cavitation clouds.

2.5.2.2.3 Schnerr and Sauer [31]

Schnerr and Sauer developed another model based on the creation and destruction of vapour bubbles, modelling the surrounding pressure and temperature conditions, with the slip between phases being neglected [33].

The VOF methodology was used in conjunction with the mixture concept and was validated by comparing results with the experimental results from flow over a NACA0015 hydrofoil, with the void fraction and velocity field determined from (2.30) and (2.31).

$$\frac{\partial \alpha}{\partial t} + \frac{\partial(\alpha u)}{\partial x} + \frac{\partial(\alpha v)}{\partial y} = \left(\frac{n_0}{1 + n_0 \cdot \frac{4}{3}\pi R^3} \right) \frac{d}{dt} \left(\frac{4}{3}\pi R^3 \right) \quad (2.30)$$

$$\nabla \cdot \mathbf{u} = - \frac{\rho_v - \rho_l}{\alpha \rho_v + (1 - \alpha)\rho_l} \frac{d\alpha}{dt} \quad (2.31)$$

where α is the vapour fraction, n_0 is the nuclei concentration per unit volume of pure liquid and R is the bubble radius.

As it is assumed here that the vapour consists of mini spherical bubbles, the authors calculate the vapour fraction from (2.32).

$$\alpha = \frac{V_v}{V_{cell}} = \frac{n_0 \cdot \frac{4}{3}\pi R^3}{V_v + V_l} = \frac{n_0 V_l \cdot \frac{4}{3}\pi R^3}{n_0 V_l \cdot \frac{4}{3}\pi R^3 + V_l} = \frac{n_0 \cdot \frac{4}{3}\pi R^3}{1 + n_0 \cdot \frac{4}{3}\pi R^3} \quad (2.32)$$

Under the assumptions that bubble-bubble interactions as well as bubble coalescence can be neglected, and that the bubbles remain spherical, the energy equation combined with the Rayleigh-Plesset equation becomes (2.33).

$$R \frac{d^2 R}{dt^2} + \frac{3}{2} \left(\frac{dR}{dt} \right)^2 = \frac{p(R) - p_\infty}{\rho_l} - \frac{2\sigma}{\rho_l R} - 4 \frac{\mu}{\rho_l R} \frac{dR}{dt} \quad (2.33)$$

The authors then control bubble growth by using the Rayleigh-Plesset relation as in (2.34).

$$\dot{R} = \sqrt{\frac{2}{3} \frac{p(R) - p_\infty}{\rho_l}} \quad (2.34)$$

The results obtained for the standard NACA0015 channel grid and free surface grid provided results in agreement with the experimental data.

2.5.2.3 Empirical Based Models

Another well-established method is to directly simulate the vaporisation and condensation rates between the phases to be included in the mass conservation equation (2.35) or (2.36).

$$\frac{\partial \alpha \rho_v}{\partial t} + \nabla(\alpha \rho_v \vec{u}) = (\dot{m}^- + \dot{m}^+) \quad (2.35)$$

$$\frac{\partial \rho_m f_v}{\partial t} + \nabla(\rho_m f_v \vec{u}) = (\dot{m}^- + \dot{m}^+) \quad (2.36)$$

where \dot{m}^- and \dot{m}^+ represent the evaporation and condensation processes respectively.

This type of model was originally proposed by Merkle et al. [31] and is mainly driven from dimensional argument, eliminating the estimation of quantities required in the Rayleigh-Plesset based models by using empirical laws for the source terms. Similar work has since been investigated by many authors including Kunz et al. [34], Singhal et al. [35], Senocak and Shyy [36] and Owis and Nayfeh [37].

2.5.2.3.1 Kunz et al. [35]

One significant body of work is by Kunz et al. who sought to develop the multiphase technology similar to Merkle et al. [38], initially focussing on sheet and super cavitating flows around submersible vehicles.

The model uses the Navier-Stokes equations to define a three species differential formulation where separate equations are provided for the transport/generation of volume fraction of liquid (which can exchange mass with condensable vapour), volume fraction of non-condensable gas (NCG) and the mixture volume. A mixture momentum equation is also provided.

The governing differential equations are cast in Cartesian coordinates, in a reference frame rotating with constant angular velocity, ω_i , and are stated as (2.37) to (2.40).

$$\left(\frac{1}{\rho_m \beta^2}\right) \frac{\partial p}{\partial \tau} + \frac{\partial u_j}{\partial x_j} = (\dot{m}^+ + \dot{m}^-) \left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \quad (2.37)$$

$$\begin{aligned} \frac{\partial}{\partial t}(\rho_m u_i) + \frac{\partial}{\partial \tau}(\rho_m u_i) + \frac{\partial}{\partial x_j}(\rho_m u_i u_j) \\ = \frac{\partial}{\partial x_i} + \frac{\partial}{\partial x_j} \left(\mu_{m,t} \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right] \right) + \rho_m g_i \end{aligned} \quad (2.38)$$

$$\frac{\partial \alpha_l}{\partial t} + \left(\frac{\alpha_l}{\rho_m \beta^2}\right) \frac{\partial p}{\partial \tau} + \frac{\partial \alpha_l}{\partial \tau} + \frac{\partial}{\partial x_j}(\alpha_l u_j) = (\dot{m}^+ + \dot{m}^-) \left(\frac{1}{\rho_l}\right) \quad (2.39)$$

$$\frac{\partial \alpha_{ng}}{\partial t} + \left(\frac{\alpha_{an}}{\rho_m \beta^2}\right) \frac{\partial p}{\partial \tau} + \frac{\partial \alpha_{an}}{\partial \tau} + \frac{\partial}{\partial x_j}(\alpha_{an} u_j) = 0 \quad (2.40)$$

where α_l , ρ_v and α_{ng} represent the liquid, vapour and non-condensable gas (NCG) volume fractions.

The density of each constituent is assumed to be constant, shown in (2.41).

$$\rho_m = \rho_l \alpha_l + \rho_v \alpha_v + \rho_{ng} \alpha_{ng} \quad (2.41)$$

and the turbulent viscosity is defined by (2.42).

$$\mu_{m,t} = \frac{\rho_m C_\mu k^2}{\varepsilon} \quad (2.42)$$

The formation and collapse of a cavity is again modelled as a phase transformation, with \dot{m}^- and \dot{m}^+ representing evaporation and condensation processes respectively.

The transformation of liquid to vapour, \dot{m}^- , is modelled as being proportional to the liquid volume fraction and the amount by which the pressures below the vapour pressure, similar to the model used by Merkle et al. The transformation of vapour to liquid, \dot{m}^+ , is modelled using a simplified version of the Ginzburg-Landau potential, giving (2.43) and (2.44).

$$\dot{m}^- = \frac{C_{dest} \rho_l \alpha_l \text{MIN}[0, p - p_v]}{(\frac{1}{2} \rho_l U_\infty^2) t_\infty} \quad (2.43)$$

$$\dot{m}^+ = \frac{C_{prod} \rho_l \alpha_l^2 (1 - \alpha_l)}{t_\infty} \quad (2.44)$$

where C_{dest} and C_{prod} are constants obtained through validation studies by Kunz et al.

The model was able to accurately predict the pressure distributions associated with natural sheet cavitation over a hemispherical body across a range of cavitation numbers considered. However, the cone and blunt forebody analyses were weak, partly attributed to an inability to capture more complex cavitation processes away from the body.

Of particular interest is Kunz's work with Medvitz [34] where this methodology was utilised to model cavitation in centrifugal pumps, neglecting the NCG volume fraction. In the instance of centrifugal pumps, values of $C_{dest} = 100$, $C_{prod} = 1000$ were chosen.

Whilst this work takes the modelling a stage further, limitations and improvements were identified, including such as adapting the mass transfer model to accommodate thermal effects, particularly relevant on cavitation breakdown; implementing time-varying pressure and mass flow boundary conditions to accommodate pumping system dynamics; and increasing the validity of the fully 3D capability.

2.5.2.3.2 Singhal et al. [36]

Singhal et al. declared that they had succeeded where previous modelling attempts, including their own, had failed by developing robust numerical algorithms that could be used as a general solver across a range of areas.

Their model uses the standard Navier-Stokes equations for variable fluid density with a conventional k- ϵ turbulence model. A simplified Rayleigh-Plesset equation is used to account for the bubble dynamics, neglecting viscous damping and surface tension terms as well as second-order derivative of the bubble radius and the mixture density was defined as a function of the vapour mass fraction, governed by the vapour transport equation, (2.45).

$$\frac{\partial}{\partial t}(\rho f) + \nabla \cdot (\rho \vec{V} f) = \nabla \cdot (\Gamma \nabla f) + R_e - R_c \quad (2.45)$$

where \vec{V} is the velocity vector and Γ is the effective exchange coefficient. The terms R_e and R_c are the evaporation and condensation rate respectively and are functions of a range of fluid properties such as pressure, flow characteristic velocity, liquid and vapour phase densities, saturation pressure and liquid-vapour surface tension, given by (2.46) and (2.47).

$$R_e = C_e \frac{V_{ch}}{\sigma} \rho_l \rho_v \left[\frac{2 p_v - p}{3 \rho_l} \right]^{\frac{1}{2}} (1 - f) \quad (2.46)$$

$$R_e = C_e \frac{V_{ch}}{\sigma} \rho_l \rho_l \left[\frac{2p - p_v}{3\rho} \right]^{\frac{1}{2}} f \quad (2.47)$$

where subscripts l and v denote the liquid and vapour phases, V_{ch} is a characteristic velocity reflecting the effect of the local relative velocity between the liquid and vapour, σ is the surface tension of the liquid, p_v is the saturation vapour pressure of the liquid for the given temperature and C_e and C_c are recommended as 0.002 and 0.001 respectively.

The relative velocity between the liquid and vapour phases is expressed using V_{ch} as the square root of the local turbulent kinetic energy, \sqrt{k} . The effects of non-condensable gases (NCGs) are also included in the refined model, allowing the ‘full cavitation model’ to be expressed in (2.48).

$$\frac{1}{\rho} = \frac{f_v}{\rho_v} + \frac{f_g}{\rho_g} + \frac{1 - f_v - f_g}{\rho_l} \quad (2.48)$$

where ρ_g is the NCG density and the evaporation and condensation rates are represented as (2.49) and (2.50).

$$R_e = C_e \frac{\sqrt{k}}{\sigma} \rho_l \rho_v \left[\frac{2p_v - p}{3\rho_l} \right]^{\frac{1}{2}} (1 - f_v - f_g) \quad (2.49)$$

$$R_c = C_c \frac{\sqrt{k}}{\sigma} \rho_l \rho_l \left[\frac{2p - p_v}{3\rho} \right]^{\frac{1}{2}} f_v \quad (2.50)$$

In terms of this ‘full cavitation model’, these values proved to have good agreement with the experimental data for the hydrofoil and submerged cylindrical body. Athavale et al. also use this model to simulate cavitating flows in three machines: a two-stage axial pump, a centrifugal water pump and a high-performance rocket pump inducer [39]. The results in each case were plausible with robust and stable convergence behaviour.

Whilst impressive, the full cavitation model has limitations in that it assumes isothermal flow and a uniform mass concentration of NCGs. The work by Athavale et al. also highlights the requirement for further calibration of the coefficients of C_e and C_c across applications.

2.5.2.3.3 Senocak and Shyy [37]

Senocak and Shyy also use the transport methodology to allow a direct interpretation of empirical parameters, developing a method from the initial consideration of a liquid-vapour phase change interface, referred to as interfacial dynamics models (IDMs).

The authors validated their model against existing approaches and experimental data of flow around an axisymmetric, hemispherical projectile and an NACA66MOD hydrofoil. Using this technique, the mass and momentum equations are written as (2.51) and (2.52).

$$\rho_L(V_{L,n} - V_{I,n}) = \rho_V(V_{V,n} - V_{I,n}) \quad (2.51)$$

$$P_V - P_L = \gamma \left(\frac{1}{R_1} + \frac{1}{R_2} \right) + 2\mu_V \frac{\partial V_{V,n}}{\partial n} - 2\mu_L \frac{\partial V_{L,n}}{\partial n} \quad (2.52)$$

$$+ \rho_L(V_{L,n} - V_{I,n})^2 - \rho_V(V_{V,n} - V_{I,n})^2$$

Once again, the energy equation is not present as thermal effects are neglected.

The mixture density is defined using the liquid volume fraction. This interfacial condition is then coupled with the transport equation of liquid volume based on dimensional argument and normalised with a characteristic timescale chosen based on the characteristic length scale and free stream velocity, enabling the mass transfer rate of the bubble cluster, as opposed to individual bubbles, to be defined in (2.53).

$$\dot{m} = \frac{\alpha_L}{t_\infty} = \frac{\rho_L(P_L - P_V)\alpha_L}{\rho_V(V_{V,n} - V_{I,n})^2(\rho_L - \rho_V)t_\infty} + \frac{(P_L - P_V)(1 - \alpha_L)}{(V_{V,n} - V_{I,n})^2(\rho_L - \rho_V)t_\infty} \quad (2.53)$$

As the model is based on an existing interface, conditions must be applied to this source term and is discussed in the paper. The resultant terms are then coupled to the transport equation of $\bar{\alpha}_L$, producing (2.54) that represents both evaporation (first term) and condensation (second term).

$$\begin{aligned} \frac{\partial \bar{\alpha}_L}{\partial x} + \nabla \cdot (\bar{\alpha}_L \tilde{\mathbf{u}}) \\ = \frac{\rho_L \text{MIN}(\bar{P} - P_V, 0) \bar{\alpha}_L}{\rho_V(V_{V,n} - V_{I,n})^2(\rho_L - \rho_V)t_\infty} + \frac{\text{MAX}(\bar{P} - P_V, 0)(1 - \bar{\alpha}_L)}{(V_{V,n} - V_{I,n})^2(\rho_L - \rho_V)t_\infty} \end{aligned} \quad (2.54)$$

This interfacial dynamics-based statement is consistent with existing models developed by Merkle et al and Singal et al and, although not completely empiricism-free, it has the distinct advantage of the empirical constants having physicality that can be represented as (2.55) and (2.56).

$$\frac{C_{dest}}{0.50\rho_L U_\infty^2} = \frac{1}{(\rho_L - \rho_V)(V_{V,n} - V_{I,n})^2} \quad (2.55)$$

$$\frac{C_{prod}}{0.50\rho_L U_\infty^2} = \frac{1}{(\rho_L - \rho_V)(V_{V,n} - V_{I,n})^2} \quad (2.56)$$

In the three configurations tested, the IDM managed to produce qualitatively comparable results, in wall pressure distributions, to the other empirical cavitation models. However, it did not perform as well when predicting the pressure and density distributions, particularly in the closure region unless a time-accurate simulation was used. The sudden change in

density profile across the interface also has implications on the near-wall treatment with the $k - \varepsilon$ turbulence model, resulting in numerical stability issues.

2.5.3 'Hybrid' Models

In some instances, work has sought to create 'hybrid' models by combining the most effective aspects of more than one cavitation model.

2.5.3.1 Huang and Wang [40]

Huang et al. [37] utilise work by Kubota et al. [40] and Senocak and Shyy [37], evaluating the differences in modelling the vaporisation and condensation processes and developing a modified density based model (MDBM).

The Huang et al. model here differs slightly from the model seen earlier and explains the interaction between viscous effects including vortices and cavitation bubbles, using the Rayleigh-Plesset equations for the growth and collapse of the bubble cluster whilst Senocak and Shyy sought to remove the requirement for adjusting the empirical parameters.

The mass transfer terms from Kubota et al. and Senocak and Shyy can be represented as in (2.57).

$$\begin{aligned} \dot{m}_v^k &= C_c \frac{3\alpha_v}{R_B} \left(\frac{2p - p_v}{3\rho_l} \right)^{\frac{1}{2}}, p > p_v & \dot{m}_l^s &= \frac{\rho_l^2 \text{Min}(p - p_v, 0)\alpha_l}{\rho_v (V_{v,n} - V_l)^2 (\rho_l - \rho_v)t_\infty}, p < p_v \\ \dot{m}_l^k &= -C_c \frac{3\alpha_v\alpha_v}{R_B} \left(\frac{2p - p_v}{3\rho_l} \right)^{\frac{1}{2}}, p < p_v & \dot{m}_v^s &= \frac{\rho_l \text{Min}(p - p_v, 0)\alpha_v}{\rho_v (V_{v,n} - V_l)^2 (\rho_l - \rho_v)t_\infty}, p < p_v \end{aligned} \quad (2.57)$$

where condensation term \dot{m}_v^k , and evaporation term \dot{m}_l^k are taken from Kubota et al.'s work and \dot{m}_v^s , and \dot{m}_l^s are taken from Senocak and Shyy's work.

Using both approaches together, an attempt was made to capture the interface between the liquid and vapour at the front of an attached cavity and simulate the vortex shedding in the rear part of the cavitation region by using the mass transfer relationships in (2.58).

$$\dot{m}_l = \chi(b)\dot{m}_l^k + (1 - \chi(b))\dot{m}_l^s, p < p_v \quad (2.58)$$

$$\dot{m}_v = \chi(b)\dot{m}_v^k + (1 - \chi(b))\dot{m}_v^s, p < p_v$$

where $\chi(b)$ is a blending function used to combine the mass transfer rate in and out of the cavity as shown in (2.59).

$$\chi(b) = 0.5 + \tanh \left[\frac{C_1(0.6b - C_2)}{0.2(1 - 2C_2) + C_2} \right] \quad (2.59)$$

with $b = \rho_m/\rho_l$, $C_1 = 4$ and $C_2 = 0.2$.

To validate the model, the author considered two flow configurations: an axisymmetric cylindrical object with a hemispherical headform and a Clark-Y hydrofoil. Whilst the Kubota model sees the cavity being present for a short time and results in a less substantial re-entrant flow and the IDM model does not capture the cloud shedding process, this hybrid model successfully captures the features at every stage from the attached cavity development to the detached cavity at the trailing edge and vortex shedding moving downstream. More importantly, it shows better agreement with experimental results than the individual models, showing a considerable increase on the accuracy of its predictions.

2.5.4 Fully Two-Phase Mixture Models

Another more complex method is to treat the phases present in the flow separately, solving balance equations for each. An approach used by Drew [41], Simonin [42] and Ishii and Hibiki [43], its aim is to overcome the issues associated with single-fluid models such as

unknown parameters or unidentified levels of empirical terms and allows a complete range of distinct physical components and regions to be represented.

2.5.4.1 Mimouni et al. [44]

Mimouni et al. outlined a compressible, unsteady, turbulent 3D model for cavitating and boiling flow, summarising five main points that differentiated their model from single-fluid models:[44]

1. The flow is not assumed to be isothermal, with the energy equations being solved and phase transitions modelled.
2. Compressible and unsteady flow transport equations are solved.
3. The thermodynamic properties of the real fluid are calculated everywhere in the flow and at each time step.
4. Cavitation may occur even without pre-existing NCGs and/or vapour bubbles being present in the flow.
5. The 'six-equation model' is used to solve the balance equations of mass, momentum and energy balance for each field/phase, outlined below:

Two mass balance equations:

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \nabla \cdot (\alpha_k \rho_k \underline{V}_k) = \Gamma_k \quad k = l, v \quad (2.60)$$

where t is time, α_k , ρ_k and \underline{V}_k represent the void fraction, averaged density and velocity of the void fraction of phase k , Γ_k is the interfacial mass transfer per unit volume and time.

The phase index k assumes the value of l for liquid and v for vapour.

Two momentum balance equations, given as (2.61).

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \nabla \cdot (\alpha_k \rho_k \underline{V}_k \underline{V}_k) \quad (2.61)$$

$$= -\alpha_k \nabla p + \underline{M}_k + \alpha_k \rho_k \underline{g} + \nabla \cdot [\alpha_k (\underline{\tau}_k + \underline{\tau}_k^T)] \quad k = l, v$$

where p is the pressure \underline{g} is the acceleration due to gravity, \underline{M}_k is the interfacial momentum transfer per unit volume and time and $\underline{\tau}_k$ and $\underline{\tau}_k^T$ represent the molecular and turbulent stress tensors respectively.

Two total enthalpy balance equations, given as (2.62).

$$\begin{aligned} \frac{\partial}{\partial t} \left[\alpha_k \rho_k \left(h_k + \frac{V_k^2}{2} \right) \right] + \nabla \cdot \left(\alpha_k \rho_k \left(h_k + \frac{V_k^2}{2} \right) \underline{V}_k \right) & \quad (2.62) \\ = \alpha_k \frac{\partial p}{\partial t} + \alpha_k \rho_k \underline{g} \underline{V}_k + \Gamma_k \left(h_{ki} + \frac{V_k^2}{2} \right) & \\ + q''_{ki} A_i + q''_{wk} - \nabla \cdot [\alpha_k (\underline{q}_k + \underline{q}_k^T)] & \quad k \\ = l, v & \end{aligned}$$

where h_k is the phase-averaged enthalpy for phase k and h_{ki} is the interfacial-averaged enthalpy.

The terms Γ_k , \underline{M}_k and $q''_{ki} A_i$ represent the interfacial transfer terms of mass, momentum and heat respectively, with A_i denoting the interfacial area concentration. The interfacial transfer of momentum is assumed to be the sum of four forces, not discussed at this time.

The development of cavitation is dependent on the existence of nucleation/cavitation sites. In this model, the nuclei come from wall nucleation or are pre-existing in the flow. The generated vapour bubbles are carried by the flow and expand in the regions where the local pressure is below the saturation pressure.

The wall nucleation model used in this work was presented by Jones et al. [45] using test cases with varying permutations of vapour generation between the wall and pre-existing cavitation nuclei in nozzle and orifices, with satisfactory agreement with experimental data of pressure and void fraction with the nozzle and good agreement with experimental results with the orifice.

Regarding improvements, the authors also explicitly suggest some including consideration of NCGs and the development of a second order turbulence model.

2.6 Conclusions

There has been a significant amount of work conducted in the area of cavitation modelling, with models of varied accuracy and complexity being developed over the last two decades. Each of these approaches has their own advantages and drawbacks depending on the application, type of cavitation present and level of accuracy required.

By looking at the models from the perspective of applying them specifically to centrifugal pumps, the interface tracking models discussed in Section 2.5.1 are insufficient for the level of complexity involved in rotating machinery due to their inability to capture ‘unsteady’ cavitation phenomena. Mixture models based on the barotropic approach as discussed in Section 2.5.2.1 are also deemed to be insufficient due to their lack of robustness and numerical instabilities even when modelling simplified geometries.

Whilst the fully two-phase approach highlighted in Section 2.5.4 would be useful in certain applications, the increased levels of accuracy may not be worth the additional computing power required.

The most effective option in this instance is suggested to be the use of the mixture models that utilise transport equations to represent the interaction between the vapour and liquid

phases as discussed in Section 2.5.2.2. If a specific focus on centrifugal pumps is desired, for pump manufacturers and related service companies, removing the requirement for a more general solver than a focus on the empirically based sub-group may be the best suited, allowing the parameters and constants used in the transfer terms to be modified based on historical experimental data and in-house expertise.

In terms of using OpenFOAM to model cavitation phenomenon, work has already been conducted in this area by Erney who validated the mixture models of Kunz, Merkle and Schnerr and Sauer contained in the software for a flat plate, hemispherical head-form and a NACA0012 hydrofoil [46]. Following successful results, one of the major recommendations for future work was to undertake studies that involved more complex geometries such as marine propulsors.

The particularly difficult aspect of taking this work further to accurately simulate centrifugal pumps is related to the modification of the solver code in order to incorporate transient analyses that require a moving mesh to represent the rotation of the impeller.

Following an evaluation of the potential use of OpenFOAM for modelling single-phase flow in centrifugal pumps in Chapter 5, an initial investigation into the inclusion of the Kunz, Merkle and Schnerr and Sauer cavitation models by using modified transient version of the interPhaseChangeFoam solver is conducted in Chapter 6.

Chapter 3

3. Cavitation Erosion Modelling

This chapter discusses the modelling techniques that have been developed to model cavitation erosion. Unlike the 'standard' cavitation models discussed in Chapter 2, these seek to model the phenomenon from a quantitative perspective in terms of material damage and erosion. Section 3.1 outlines the basic concept of cavitation erosion and is followed by an overview of the related energy considerations and mechanisms involved in Section 3.2 and Section 3.3. Methods of modelling cavitation erosion are discussed in greater detail in Section 3.4 and an insight into the potential of coupling an erosion model with a CFD code is given in Section 3.5, with conclusions drawn being discussed in Section 3.6.

3.1 Cavitation Erosion

When the cavities formed in a flowing liquid are subjected to increased pressures downstream, the growth of these regions stop, reverses and results in the collapse and disappearance of the cavities and potential erosion of the solid walls.

This erosion is due to the extremely violent collapsing processes that, despite lasting only a few nanoseconds, emit pressure waves across tiny areas within the region of cavitation through spherical bubble collapse, vortex collapse or micro-jet formation.

The energy concentration associated with these pressure waves produce large stress levels on the surface that can exceed the yield strength of the relevant material, forming

indentations of a few micrometres or 'pits'. If the existence of cavitation remains in the machine, the repetitive nature of the cavity collapses and subsequent 'pitting' can lead to significant material loss and failure of the component.

Whilst there are similarities between cavitation erosion and 'standard' erosion by liquids, there are important differences including that cavitation erosion is a much milder process, with material particles only being detached per millions of collapses in comparison to every few thousand impact droplets in 'standard' erosion wear. If both processes are present simultaneously, the wear rate of a material can be significantly increased. Should the additional issue of corrosion exist, this wear rate can be increased even further [9].

3.2 Energy Considerations in Cavitation Erosion

Modelling cavitation erosion is difficult as it involves complicated flow phenomena coupled with material reactions. In order to assess the magnitude of cavitation erosion, Hamitt [47] considered the theory of energy conversion as shown in Figure 3.1, suggesting that cavitation damage would occur when the potential energy contained within a shed collapsing cavity (E_c) exceeded a damage threshold (E_s), a function of the material properties and not the cavitation type.

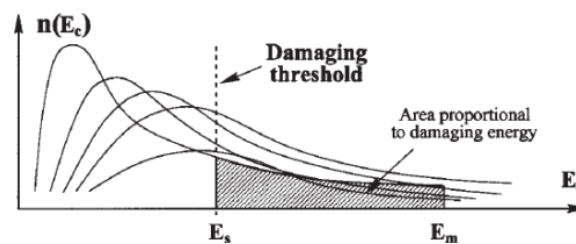


Figure 3.1 - Energy Spectra in Relation to Cavitation Erosion [47]

By considering this, the 'energy cascade' concept was identified by Fortes-Patella et al. [48], outlining how the energy contained within the micro cavity structure is converted to

radiating acoustic pressure waves, through the conversion of potential to kinetic energy during the final collapse phase of the macro scale cavity.

The risk of cavitation erosion was also considered in terms of energy by Bark et al. [49] who stated that the kinetic energy density is at a maximum at the cavity interface and will increase as the cavity collapse proceeds. This concept can also be used for groups and sequences of cavities.

3.3 Cavitation Erosion Mechanisms

In order to predict the effects of cavitation erosion, impact pressures associated with the different phenomena relating to each form of cavity collapse must be established.

Depending on the situation, the 'energy cascade' process can include many underlying mechanisms as outlined by Franc and Michel [5]:

- Collapse and rebound of single spherical bubble
- Micro-jet from spherical bubble
- Collective micro-bubble collapse
- Cavitating vortices

3.3.1 Collapse and Rebound of a Spherical Bubble

Various studies have shown that high values of temperature and pressure exist in the final moments of the spherical bubble collapse, followed by the emission of a high intensity pressure wave which can propagate towards the material surface [5].

At the centre of the bubble, the duration of this pressure wave is in the region of one microsecond, with the amplitude of the wave approximately 100 MPa.

3.3.2 Micro-jet

When a bubble collapses under non-symmetrical conditions close enough to a wall surface, a micro-jet is directed towards this wall at velocities in the region of 100-150 m/s. This mechanism is considered to be the most common mechanism in cavitation erosion [50].

The micro-jet phenomenon was originally investigated by Plesset and Chapman [51] who numerically modelled two cases: a bubble initially in contact with the wall and a bubble that is initially half its radius from the wall. The time history of the shape of these two bubbles and generation of the micro-jet are given by surfaces A to J in Figure 3.2 (a) and (b) respectively.

The solid wall was found to influence the bubble early in the collapse, mainly by reducing the upward motion of the lower portion of the bubble. It is noted that the bubble in case (b) still moves upwards towards the bubble centre but because this motion is reduced, the bubble moved towards the wall.

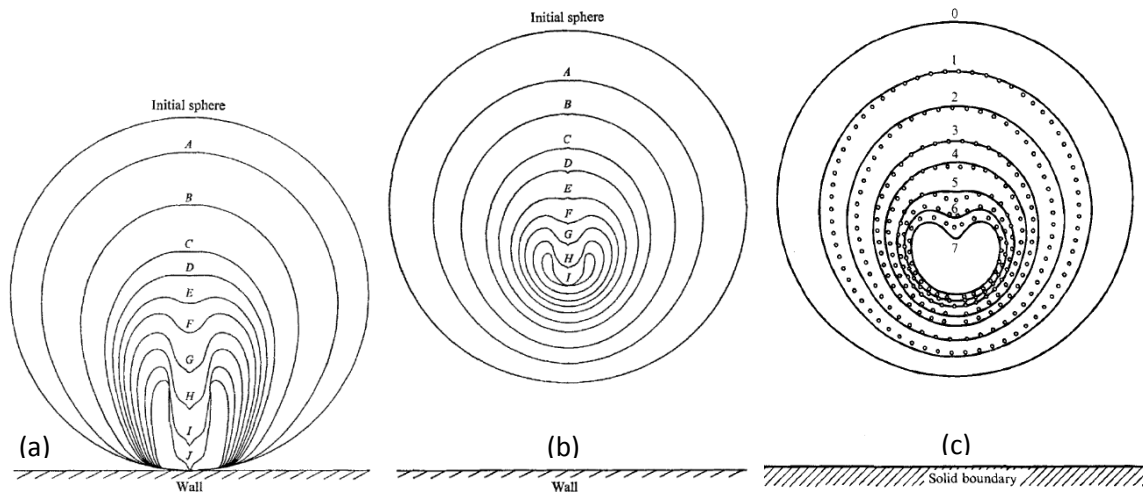


Figure 3.2 - Micro-jet Formation Due to Bubble Collapse Close to Solid Wall [51] [52]

The kinetic energy gained by the bubble in this process is concentrated in the upper portion of the bubble, eventually forming a jet. The speed of this jet was calculated by Plesset and

Chapman to be 130 m/s in Case (a) and 170 m/s in Case (b). Whilst the speed of the jet in Case (a) is lower, more damage is caused as it hits the surface directly as opposed to travelling through the liquid for a distance of approximately five times its diameter before reaching the surface.

This theory was confirmed experimentally by Lauterborn and Bolle [52] who compared their results with case (b) as seen in Figure 3-2 (c).

The pressure rise due to the impact of the micro-jet at such high speeds can be approximated using the water hammer formula (3.1) developed by Joukowski and Allievi [5].

$$\Delta p = \rho c v \quad (3.1)$$

where ρ and c are the density and sonic velocity of the liquid and of sound and v is the velocity of the jet.

This corresponds to typical impact pressures around 150 MPa, with the duration of the pressure pulse being fixed by the jet diameter, d , and in the order of $d/2c$. For a given bubble diameter of 1 mm, the jet diameter is typically 0.1 mm, leading to small durations in the region of $0.03\mu s$. It is highlighted that both of these hydrodynamic mechanisms give rise to pressure pulses with the same order of magnitude as the yield strength of 'usual' metals.

3.3.3 Collective Collapse

Collective effects are present when a cloud of bubbles collapse. Following the initial collapse of a single bubble close to a solid wall, the micro-jet piercing the bubble can result in the formation of a 'vapour torus', forming smaller bubbles that undergo subsequent collective collapses and cause cascades of implosions [5].

The pressure waves emitted by the collapsing and rebounding of such bubbles increases the impact velocities of the surrounding bubbles as well as the amplitude of their own pressure waves.

3.3.4 Cavitating Vortices

Cavitating vortices tend to appear in shed flows, such as submerged jets, as well as at the rear of partial cavities as seen in hydrofoils and pump impeller blades.

They too appear to be responsible for significant erosion in fluid machinery with typical collapse velocities higher than 100 m/s and impact pressures of the same order of magnitude as shock waves and micro-jets mentioned previously.

Franc and Michel state that there are two significant attributes which appear to cause such high erosion due to cavitating vortices: the formation of a 'foamy cloud' at the end of the axial collapse in which cascade mechanisms can occur; and the relatively long time period of high pressure impact (loading period) which typically last several tens of microseconds.

A review of the impact loadings related to the different mechanisms is shown in Table 3.3-1.

Table 3.3-1 – Impact Loads for Various Cavitation Mechanisms [5]

Mechanism	Type of Loading	Amplitude (MPa)	Duration (μs)
Micro-bubble Collapse	Pressure Wave	100	1
Micro-jet (from 1mm bubble)	Impacting Jet	150	0.03
Collective Micro-bubble Collapse	Pressure Waves	>>100	>>1
Cavitating Vortices	Impacting Jet	>100	>10

3.4 Predicting Cavitation Erosion

3.4.1 Empirical Methods

Many empirical methods are used in industry to predict cavitation erosion damage and are extremely useful tools in the design of machines, particularly in terms of material selection. However, the correlations between erosion resistance of materials and mechanical properties are often only valid for a given range of flow rates and cavitation parameters. Ideally a universal model would be used, with a computational method that fully predicts the levels of cavitation erosion without the requirement for model tests being the ultimate aim.

3.4.2 Cavitation Erosion Models

In order to address the issues associated with empirical methods, there have been many attempts to improve our understanding of the phenomenon and predict the levels of risk associated with cavitation erosion through numerical modelling. However, this has proven to be difficult due to the combination of complex flow phenomena and material reactions related to cavitation erosion [53]. The following highlights several risk assessment models that attempt to provide solutions to this problem.

3.4.2.1 Kato et al [54]

The model presented by Kato et al follows a six-phase cavitation development process and assumes that the primary mechanism for cavitation erosion is the shock wave caused by the collapse of bubbles separated from the sheet cavity. The assessment process used is:

Stage 1: Cavity type and extent

Stage 2: Cavity generation rate

Stage 3: Number and size distribution of cavity bubbles

Stage 4: Characteristics of collapsing bubbles

Stage 5: Impact force/pressure distribution on solid wall due to cavity bubble collapse

Stage 6: Amount of erosion caused by successive impact forces

The cavity generation rate in Stage 2 is predicted using the airflow rate into a ventilated cavity, assuming that the flow rate necessary to maintain the length of the cavity should be the same as for a vapour cavity. The quantities of impact force/pressure in Stage 5 are established and correlated using the pit distribution. Stage 6 establishes the material deformation and removal and is studied from a metallurgical perspective.

It is clearly difficult to establish the number and size distribution of cavity bubbles in the flow as their size changes rapidly, however Kato et al estimate this information from the measurement of air bubble distribution downstream of the cavity collapse region as these are the 'remains' of the cavity bubbles and their distribution should thus be similar (Figure 3.3).

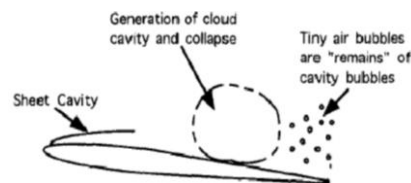


Figure 3.3 - Generation and Collapse of Cloud Cavity [54]

The impact pressures on the foil surface of the hydrofoil used in this study are predicted using a model that simulates a single bubble in infinite space, neglecting interference effects in bubble clouds and wall diffraction effects.

The force/pressure spectrums of the impacting collapses are established by combining the cavity generation rate and number and size of the bubbles. The region of cavity collapse and collapse ambient pressure are estimated and the cavity collapse rate at a specific

location determined. This lengthy process is simplified by assuming that only bubbles in the effective layer have impact forces significant enough to damage the material surface.

This allows the trajectory of the bubbles to be shown against a reference trajectory (Figure 3.4) with three reference length scales being established: bubble layer thickness (h_b), effective layer thickness (h_e) and reference trajectory (h).

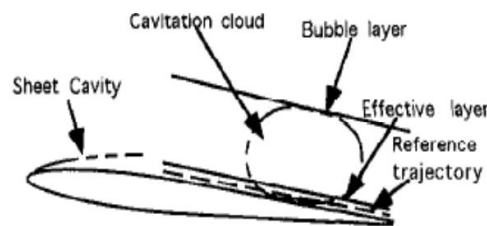


Figure 3.4 - Bubble Layer, Effective Layer and Reference Trajectory [54]

Whilst this model does provide a quantitative prediction of the cavitation erosion present in fluid flow without the requirement of a model test, it has clear limitations in that numerous parameters are assumed or neglected including the initial pressure within the bubbles, change of ambient pressures surrounding the bubbles during collapse, spatial distribution of collapsing bubbles as well as bubble-bubble and bubble-wall interactions.

3.4.2.2 Bark et al [55]

Working in line with the European EROCAV project, Bark et al developed a model based on the principle that cavitation erosion is mostly the result of an 'accumulated energy transfer from macro scale cavities to collapsing cavities close to a solid surface.'

A conceptual model is created to 'sharpen the visual interpretation of observations of cavitation processes by high-speed video' and a 'systematic nomenclature' is given to describe and classify the cavitation behaviour with respect to this focusing and the generation of erosion. The small cavities that result from this focussing cavity are assumed to cause the pitting in the material. They are considered to be spherical at the start of the

collapse but if they are close enough to the wall surface towards the end of the collapse, a high speed micro-jet is formed and hits the surface.

Both the jet and the local pressure wave generated by the collapsing cavities are considered as contributors to the deformation and removal of the material surface and the authors state that either one of these mechanisms can dominate dependent on the specific conditions in question.

The authors highlight that the most violent collapses of the cavities are associated with the collective collapse of vapour cloud cavitation by way of an energy transfer from the peripheral bubbles to the innermost bubbles, concentrating or 'focussing' the energy into a small volume.

Considering the energy cascade model, the cavitation erosion process is decomposed by physically identifiable sub-processes:

- Creation of a transient (usually travelling) cavity from the global cavity
- 'Main-focusing' collapse – early collapse motion of the transient cavity that can be observed by the selected recording technique
- 'Micro-focusing' collapse - last part of the collapse, not resolved in detail by high-speed recordings.
- Rebound

Several steps are used to assess cavitation erosion effects by visual observation. The first step is to search for violent rebounds and estimate their aggressiveness. The cavity is then tracked back to its origin and the vapour content is assessed.

Information regarding the 'focussing' efficiency is also obtained using the degree of disintegration, acceleration of collapse motion, shape and symmetry of collapse motion

and the cyclic behaviour of the focussing in relation to forced oscillations. These items mainly concern the focussing cavity from initial development to rebound, regarded as the indicator of violent collapse. There is no detailed observation of the micro-focussing processes, but it is identified as an area for future consideration.

The cavitation erosion model proposed by Bark et al can be applied to large scale cavities relatively easily. However, when more complex behaviour is present in the fluid flow, it becomes more challenging, with the potential for over-estimation or under-estimation of the erosion prediction occurring due to scale effects, lack of experience or full scale correlation. The time and costs associated with the experimental high-speed video recording are acknowledged as being more limiting than the analysis itself.

3.4.2.3 Fortes-Patella et al [48]

Fortes-Patella et al proposed a model by evaluating the energy transfer between the cavitating flow and the material surface and is based on several phases:

- Collapse of the vapour structures in the cavitating flow
- Emission and propagation of the pressure wave during collapse of vapour structures in the cavitating flow
- Interaction between pressure waves and neighbouring solid surface
- Damage of material exposed to the pressure wave impacts

Similarly to the model proposed by Bark et al, it uses the energy cascade concept, with the potential power contained within the macro cavities being converted into acoustic power produced by the collapsing cloud of microbubbles.

The pressure waves emitted during the collapse of these vapour structures are suggested to be the main contributor of cavitation erosion and can be generated in one of two ways: spherical bubble or vortex collapse or by micro-jet formation.

The input for this model was taken as the development of macro cavities which can be seen through experiments or simulated using multiphase CFD. The volume damage rate can also be calculated as the output from the model.

Instantaneous Potential Power

The instantaneous potential power of the cavitating flow is derived by considering the macroscopic cavity structure as represented in (3.2).

$$P_{tot} = \Delta p \left(\frac{dV_{vap}}{dt} \right) \quad (3.2)$$

where $\Delta p = p_{\infty} - p_{vap}$, p_{∞} is the surrounding pressure p_{vap} is the vapour pressure and V_{vap} is the vapour volume at a given time t .

Flow Aggressiveness Potential Power

The flow aggressiveness potential power is derived from the potential power specifically related to the aggressiveness of the erosion prior to the cavity collapse, giving (3.3).

$$P_{pot}^{mat} = \eta^{**} P_{tot} \quad (3.3)$$

where η^{**} is the energy transfer efficiency and is a function of the hydrodynamic characteristics of the main flow V_{ref} and σ , the reference velocity and cavitation number respectively, and the distance between the collapse centre and the material surface, L .

The flow aggressiveness power is affected by the type, unsteadiness and geometry of the cavitating flow. However, as the potential power P_{tot} already considers this information,

the predominant factor is the distance between the collapse centre and the material surface.

Pressure Wave Power

The pressure wave applied to the material surface upon the bubble collapsing is defined in (3.4).

$$P_{waves}^{mat} = \eta^* P_{pot}^{mat} \quad (3.4)$$

where η^* is the efficiency determined by the collapsing of the spherical vapour and gas bubbles and is mostly dependent on the variation in the surrounding pressure p_∞ relative to the pressure at initial cavity generation, from which the potential power is determined, as well as the air content in the flow.

Volume Damage Rate

The authors also measured the volume damage rate, V_d , using a 3D laser profilometer, relating the results to the flow aggressiveness, $P_{pot}^{mat} / \Delta S$ as in (3.5).

$$V_d = \frac{\eta^* P_{pot}^{mat}}{\beta \Delta S} = \frac{P_{waves}^{mat}}{\beta \Delta S} \quad (3.5)$$

where ΔS is the analysed sample unit surface area and β is a function strongly dependent on the characteristics of the specific material in question such as surface hardness and yield stress.

Overall, this model has the advantage that it follows the description of the physical energy transfer processes. However, as the model is directly dependent on the establishment of two efficiencies, neither of which are currently available in open literature, further work is required in order for the model to be sufficiently useful.

3.4.2.4 Dular et al [53]

The model developed by Dular et al. is based on the damage caused to the material surface when a bubble collapses in the close vicinity of the solid surface.

The authors suggest that there is a strong relationship between cavitation erosion of the material surface and visual cavitation structures. A model is created based on theoretical and some empirical considerations obtained during previous studies by different authors, embracing the theories of cavitation cloud collapse (Shimada et al. [56] and Brennan [6]), attenuation of the pressure wave (Beranek [57]), micro-jet formation (Plesset and Chapman [51]) and pit formation (Lush [58]).

The model was tested against experimental pit count measurements found on copper foil coatings on hydrofoils by Dular et al. [59] and Bachert et al. [60] as well as a radial pump impeller geometry, using cameras to capture the mean value and standard deviation of grey level in the flow, based on the level of illumination of the fluid from a light source.

The complex cavitation process leading to pit formation is presented by the authors as in Figure 3.5.

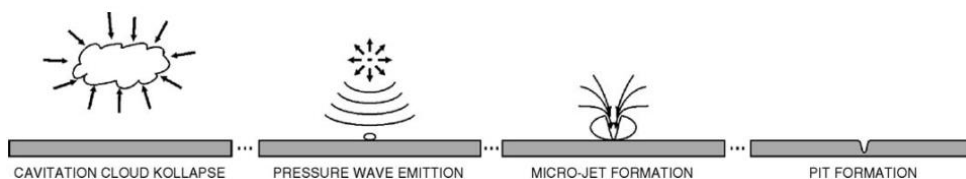


Figure 3.5 - Events Included in Cavitation Modelling Process [53]

- Collapse of the cavitation cloud causes a shock wave that radiates into the fluid
- The magnitude of the shock wave is attenuated as it travels towards to solid surface

- Single bubbles present near the solid surface begin to oscillate and a micro-jet phenomenon will occur if the bubbles are close enough to the wall
- The damage (single pit) is caused by a high energy velocity liquid jet impacting the solid surface

In this work, only the incubation period, where the surface is plastically deformed but no material is lost, is considered.

The power and magnitude of the emitted pressure wave are closely related to the rate of cavitation cloud collapse and the surrounding pressure, corresponding to the instantaneous potential power, P_{tot} , as previously defined by Fortes-Patella et al in (3.2). From acoustics, the magnitude of the emitted pressure wave is proportional to the square root of the acoustic power.

It is assumed that the pressure difference in (3.2), Δp , remains approximately constant. Taking this into consideration, the spread of the mean change in cavitation cloud volume across the hydrofoil surface reveals the associated mean distribution of the pressure wave emitted by the collapsing cloud.

Due to the inability to capture the instantaneous change of cavitation cloud volume (the vapour shedding frequency was too high for the image capturing equipment), the authors instead use the standard deviation of grey level, s . This parameter is used as it is hypothesised that there is a relationship between the time derivative of vapour cloud volume and the distributions of standard deviation of grey level from both top and side views as in (3.6).

$$grey\ level = f(V) \Rightarrow s \propto \left(\frac{dV}{dt} \right) \quad (3.6)$$

In terms of the damage to the material surface, the authors begin by using the theory developed by Plesset and Chapman and discussed in Chapter 2 that the presence of a solid surface can influence the bubble collapse process, using (3.7) to determine the micro-jet impact velocity [51].

$$v_{jet} = 8.97\gamma^2 \sqrt{\frac{p - p_v}{\rho}} \quad (3.7)$$

where γ is the non-dimensional distance from the bubble centre to the surface ($\gamma = H/R$, where H is the distance and R is the bubble radius).

Using theory developed by Plesset and Chapman again, the ‘water hammer’ pressure applied to the material at the impact of the micro-jet is defined in (3.8).

$$p \approx v_{jet}\rho_0c_0 \quad (3.8)$$

where ρ_0 and c_0 are the density and sonic velocity of the liquid.

In this instance, deformation due to the impacting jet only occurs when the velocity is high enough to induce a yield stress p_y after which the surface responds as a perfectly plastic solid. The expression (3.9) derived by Lush [58] is used to determine this critical velocity at which the stress reaches p_y and cause the plastic flow of the material.

$$v_{crit} = \sqrt{\frac{p_y}{\rho_l} \left(1 - \left(1 + \frac{p_y}{B} \right)^{-\frac{1}{n}} \right)} \quad (3.9)$$

where p_y is the material yield stress, ρ_l is density, $B=300$ MPa and $n=7$ (based on the impact of water [58]).

The authors then determine the depth of the pit due to the micro-jet impact by assuming that, after reaching the required pressure for plastic flow, the remainder, p_{def} given in (3.10) is converted into deformation energy.

$$p_{def} \approx v_{def} \rho c = (v_{jet} - v_{crit}) \rho c \quad (3.10)$$

The duration of the water hammer stress is taken as the time for the impact signal to travel the full radius of the jet, r_{jet} , and is given in (3.11).

$$t_{def} = \frac{r_{jet}}{c} \quad (3.11)$$

which allows the maximum depth of the pit to be determined from (3.12).

$$d_{pit} = v_{def} t_{def} \quad (3.12)$$

The authors are then able to calculate the area of the pit based on a ratio between the pit radius and pit depth. Whilst they use a ratio of 26.7 based on previous laser profilometry on the copper material, this can vary depending on the material used.

Overall, the model presented in this paper is able to predict the aggressiveness of the cavitation erosion on a hydrofoil as well as a more complicate setup involving a pump impeller across a range of running conditions.

3.5 Complete Quantitative and Qualitative Modelling in CFD

Whilst the development of erosion models is beneficial in terms of predicting the erosion rates for material surfaces, the ultimate aim is to be able to model the full effects in one process for design and analysis of turbomachinery and other devices. At present, Dular and Coutier-Delgosha appear to be the only authors to publish such work [50].

3.5.1 Dular and Coutier-Delgosa

This work couples an 'in-house' CFD code using a barotropic cavitation modelling approach and an erosion model derived from the work by Dular et al. [59], analysing a hydrofoil with copper foil coating in two dimensions. The coupling is performed by transferring the information from the CFD simulation to the erosion model at each time step, enabling a time-evolution of the cavitation erosion as well as the final extent of damage.

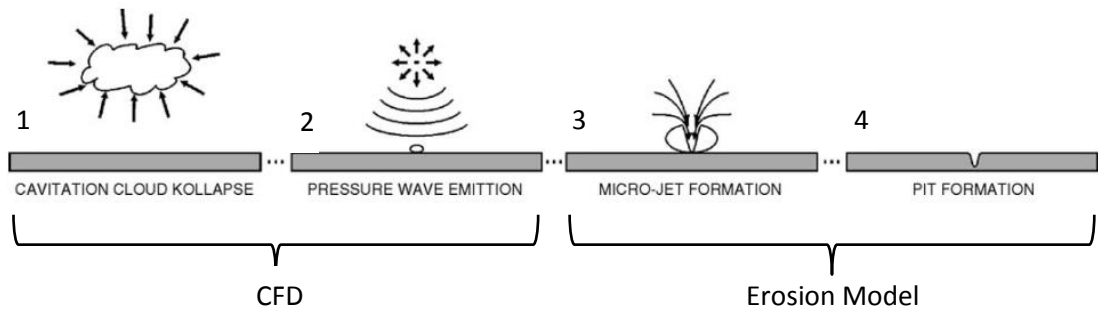


Figure 3.6 - Analysis Process for Coupled Method [50]

Items 1 and 2 in Figure 3.6 involve the prediction of the pressure peaks on the solid surface due to the cavity collapse and are provided by the CFD calculations. Items 3 and 4 are established using the erosion model presented by the authors as in Section 3.4.2.4. The full process created by Dular and Coutier-Delgosa for the prediction of cavitation erosion is shown in Figure 3.7.

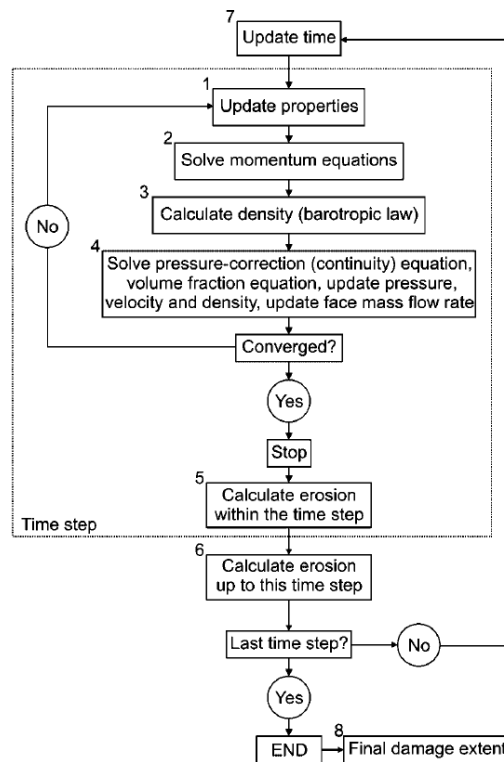


Figure 3.7 - Solution Algorithm for Coupled Method [50]

At each time step, the solution algorithm solves the governing equations sequentially. The authors outline the following steps for a single iteration:

1. Fluid properties are updated, based on the current solution. If the calculation has just begun, the fluid properties are updated based on the initialized solution.
2. The momentum equations are each solved in turn using current values for pressure and face mass fluxes, in order to update the velocity field.
3. The density and the speed of sound are calculated according to the barotropic state law.
4. Since the velocities obtained in the second step may not satisfy the continuity equation locally, an equation for the pressure correction is derived from the continuity equation and from the linearized momentum equations. This pressure correction equation is then solved

to obtain the necessary corrections for the pressure, velocity and density fields and the face mass fluxes, so that continuity is satisfied.

A check for convergence of the equation set is made. If the convergence criteria are not met steps 1–4 are continued until convergence is obtained. If convergence is obtained, calculation continues with item 5.

5. New damage of the surface within the time step is calculated.

6. The new damage is added to the sum of the damage from the previous time steps.

7. The time is updated and the iteration procedure for the new time step begins. These steps are continued until the last time step is reached.

8. Finally, the desired time of exposure to the cavitation is given and the damage extent is determined through extrapolation.

Due to the complex nature of the case, only 38ms of time was simulated. However this covered 10 cavitation cloud collapses, enough to provide statistically averaged results.

Whilst the model is relatively simple, it shows good agreement between the experimental data and the results predicted by the solution algorithm.

Intensity of the cavitation erosion as well as its downstream extent was predicted correctly in most instances. However, there was disagreement relating to the upstream erosion limits, with the model concentrating the damage over a smaller area than expected. This was attributed to the CFD simulation.

Despite the positive results, there are clearly some improvements that can be made such as modelling a complex 3D geometry and the inclusion of a more accurate cavitation model.

3.6 Conclusions

The models and mechanisms explored briefly in this chapter highlight the complicated nature of the prediction of cavitation erosion through modelling techniques.

Whilst the models discussed have each provided positive results in their own right, it is clear that significant testing is required for more complex geometries in order to fully evaluate these under more realistic situations.

The focus moving forward, as highlighted by Dular et al in Section 3.5.1, should be in attempting to develop coupled CFD methodologies based on accurate cavitation and cavitation erosion models. Whilst these are both research areas in their own right, initial investigations could look to develop the existing approach developed by Dular et al.

Based on the limitations highlighted by the authors and the models discussed in Chapter 2, the process seen in Figure 3.7 could potentially be enhanced by using a more complex cavitation model in order to capture the collapse region more accurately than the barotropic state law used. Following positive results, this could be taken further to model more complicated geometries such as turbomachinery.

As the focus of this work is to investigate the applicability of OpenFOAM for modelling centrifugal pumps with a specific focus on cavitation, the cavitation erosion aspect is not taken any further at this stage. However, it is conceivable that a coupled modelling approach could be taken by creating a user-defined function (UDF) in the well-established ANSYS Fluent CFD program. In terms of OpenFOAM, investigations relating to its potential use for such applications would be yet another stage further.

Chapter 4

4. CFD Methodology

This chapter discusses the governing equations and turbulence model utilised in the centrifugal pump case study. As the research focuses on the applicability of OpenFOAM to model such turbomachinery, the vast majority of discussion will surround this software package, however supplementary information is provided that relates to the commercial CFD packages.

The structure of OpenFOAM will be discussed as well as the time and convection discretisation schemes used to determine the most efficient and effective method for modelling the centrifugal pump. The solvers used to perform the steady state (frozen rotor), transient single-phase and transient cavitating flow will also be outlined and the additional utilities used to improve the understanding of the effectiveness of these solvers will be discussed. In addition, a summary of the centrifugal pump details is provided and use of ANSYS CFD-Post post-processing software used in for the studies is discussed.

4.1 General Transport Equation

The conservation equations of mass, momentum and energy can be represented in a standard form by using the general transport equation:

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{\text{Temporal Derivative}} + \underbrace{\text{div}(\rho\phi\mathbf{u})}_{\text{Convective Term}} - \underbrace{\text{div}(\Gamma(\text{div}\phi))}_{\text{Diffusion Term}} = \underbrace{S_\phi}_{\text{Source Term}} \quad (4.1)$$

where ϕ is a general variable, ρ is the fluid density, \mathbf{u} is the velocity vector and Γ is the diffusion coefficient.

For an incompressible fluid, the fluid density ρ is constant, simplifying (4.1) to:

$$\rho \frac{\partial(\phi)}{\partial t} + \rho(\text{div}(\phi\mathbf{u})) - \text{div}(\Gamma(\text{div}\phi)) = S_\phi \quad (4.2)$$

4.2 k - ω SST Turbulence Model

Whilst there are several turbulence models that can be used to solve CFD problems, the model employed in the following studies is the k - ω SST (Shear Stress Transport) model developed by Menter [61].

The k - ω SST model is a hybrid two-equation model that combines the advantages of the k - ω and k - ϵ models. Illustrated in Figure 4.1, the k - ω model is used close to the wall, down through the viscous sub-layer, switching to k - ϵ within the free-stream to avoid sensitivity issues to inlet free-stream turbulence properties associated with the k - ω model.

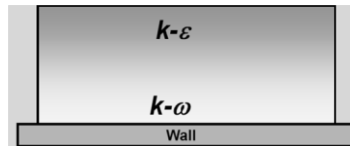


Figure 4.1– Illustration of The k - ω SST Model

The model solves equations for turbulent kinetic energy, k , which determines the energy contained within the turbulence, and the specific turbulence dissipation, ω , which determines the scale of the turbulence. As the flow is assumed to be incompressible in the following studies, the equations become:

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P_k - \beta^* k \omega + \frac{\partial}{\partial x_j} \left[(v + \sigma_k v_T) \frac{\partial k}{\partial x_j} \right] \quad (4.3)$$

$$\begin{aligned} \frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[(v + \sigma_\omega v_T) \frac{\partial \omega}{\partial x_j} \right] \\ + 2(1 - F_1) \sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i} \end{aligned} \quad (4.4)$$

where v_T is the kinematic eddy viscosity, determined by

$$v_T = \frac{\alpha_1 k}{\max(\alpha_1 \omega, SF_2)} \quad (4.5)$$

and the closure coefficients and other relations can be found in OpenFOAM

OpenFOAM (Open Source Field Operation and Manipulation) is a set of customisable numerical solvers written using the C++ programming language for solving continuum mechanics problems such as computational fluid dynamics (CFD). It also includes functionality enabling both pre- and post-processing.

Originally developed towards the end of the 1980s at Imperial College, London, it was first released into the public domain in 2004 and is now produced by ESI-OpenCFD and distributed by the OpenFOAM Foundation under a GNU General Public License.

This free licence allows users to exploit the maximum computing power at their disposal without the financially restrictive requirement of licensing costs associated with commercial codes.

This, allied to its highly customisable platform that allows users to develop new solvers and modify existing techniques to suit specific research aims, makes it an extremely powerful resource in the CFD arena.

Whilst there are distinct advantages in the OpenFOAM software in comparison to commercial codes, there are downsides, mostly associated with the lack of a dedicated

graphical user interface (GUI). This means that most of the work involved in simulating a case is command-line driven, with setup and options being selected through modifying text in particular files within the program. As this takes time to become accustomed too, there is a steep learning curve for new users of the code that could otherwise have been spent running simulations on alternative packages.

4.2.1 OpenFOAM Cases

When working with OpenFOAM, a user will normally begin with a 'case' directory, essentially a working directory in which all the files and subdirectories relating to a particular problem are stored. As the software provides a vast array of tutorial examples, the technique is to establish which is the most relevant to the problem you are attempting to solve and use it as a template from which you can construct and refine your own case.

4.2.1.1 Case File Structure

Each OpenFOAM study follows the same basic structure, with a minimum number of files required to run each application. An example of the structure is shown in Figure 4.2.

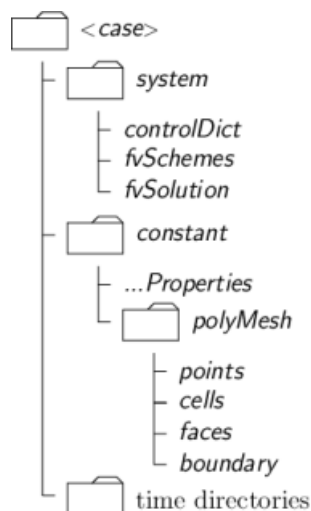


Figure 4.2 - OpenFOAM Case File Structure

There are three main directories that are always present: *constant*, *system* and a minimum of one time directory, for example the default initial time, *0*.

The *constant* directory contains a subdirectory *polyMesh* that defines the entire case mesh as well as files describing the physical properties required for the particular problem such as *turbulenceProperties*.

The *system* directory contains the settings for parameters associated with the solution procedure and always includes a minimum of three files; *controlDict*, *fvSchemes* and *fvSolution*. The *controlDict* is used to control various parameters such as the start/end times, time step size, maximum Courant number as well as functions that can be used for data output. The *fvSchemes* file specifies the discretisation schemes to be used during the solution process. The *fvSolution* file defines the equation solvers, tolerances and algorithm controls for the simulation.

The *time* directory contains all of the individual files containing the data of each type of field being solved in any given simulation such as pressure (*p*), velocity (*U*) and specific kinetic energy (*k*). This information is specified by the user as initial values and boundary conditions when constructing the simulation and written to file by OpenFOAM during the solution process.

4.2.2 Dimensional Units

As with any continuum mechanics software package, properties are defined by using a specific type of unit, such as volume in cubic metres (m^3) or pressure in Pascals ($\text{kgm}^{-1}\text{s}^{-2}$). In OpenFOAM, the units relating to each type of field being calculated during a simulation are specified by using a *dimensionSet*, with the format based on the following base units:

Table 4-1 – Dimensional Units Used in OpenFOAM

No	Property	SI Unit
1	Mass	Kilogram (kg)
2	Length	Metre (m)
3	Time	Second (s)
4	Temperature	Kelvin (K)
5	Quality	Kilogram-Mole (kgmol)
6	Current	Ampere (A)
7	Luminous Intensity	Candela (ca)

In Table 4-1, each value corresponds to the power of each of these base units, so for a velocity term (ms^{-1}), the dimensions would be specified by these seven scalars delimited by square brackets as [0 1 -1 0 0 0 0].

4.2.3 Numerical Schemes

One of the most important ‘dictionaries’ in OpenFOAM is *fvSchemes*, located in the *system* directory. It specifies the numerical schemes for terms such as derivatives in equations relevant to the application being used

OpenFOAM aims to offer an unrestricted choice to the user and includes a variety of terms that are assigned numerical schemes in this dictionary, such as divergence ($\nabla \cdot$) and time schemes. Furthermore, the derivative terms have a choice of discretisation practice, with standard Gaussian finite volume integration being the standard choice. There is also the ability to define what interpolation scheme to use, with specific schemes being designed for particular divergence terms.

4.2.4 Time Discretisation

OpenFOAM provides three time discretisation schemes; the first-order Euler scheme and the second order Backward Differencing and Crank-Nicholson approaches.

By using (4.2), the general form of the transport equation for an incompressible fluid can be expressed as:

$$\int_t^{t+\Delta t} \left[\rho \frac{\partial}{\partial t} \int_{V_P} \phi dV + \rho \int_{V_P} \text{div}(U\phi) dV - \int_{V_P} \text{div}(\Gamma_\phi(\text{div}\phi)) dV \right] dt \quad (4.6)$$

$$= \int_t^{t+\Delta t} \left(\int_{V_P} S_\phi(\phi) dV \right) dt$$

where V_P is the control volume. By assuming that this control volume does not change over time as well as the density and diffusivity within the control volume, this becomes:

$$\frac{\rho\phi_P(t + \Delta t) - \rho\phi_P(t)}{\Delta t} V_P + A[\phi_f(t + \Delta t) + \phi_f(t)] \quad (4.7)$$

$$- B[(\text{div}\phi)_f(t + \Delta t) + (\text{div}\phi)_f(t)] = S$$

where A and B are coefficients.

From here, each of the three time discretisation schemes are discussed in turn.

4.2.4.1 Euler

The Euler implicit method uses only the value of the 'new' time ($t + \Delta t$) in all terms in (4.7) with the exception of the time term, giving

$$\frac{\rho\phi_P(t + \Delta t) - \rho\phi_P(t)}{\Delta t} V_P + A\phi_f(t + \Delta t) - B(\text{div}\phi)_f(t + \Delta t) = S \quad (4.8)$$

where it is clear that the flux of node P is related to the face flux at time $t + \Delta t$ only, highlighting the first order accuracy of this approach. Whilst the lack of inclusion of the difference between the times t and $t + \Delta t$ results in a method that is not as accurate as the higher order schemes available, it does have the advantage of often being more stable.

4.2.4.2 Crank-Nicholson

The Crank-Nicholson is also an implicit method that utilises the simplified transport equation in (4.7) this time relating the flux at point P to the face flux at both t and $t + \Delta t$ and resulting in an approach that is second-order time accurate.

Of particular interest is the fact that OpenFOAM provides a blending coefficient, allowing a combination of Euler and Crank-Nicholson methods to be used. A value of 1 represents a pure Crank-Nicholson, whilst a value of 0 represents a purely Euler time discretisation. Selecting a value between 0 and 1 allows a weighted combination of these schemes.

Whilst second order accurate, this method is typically less stable than the first-order Euler scheme. However, if the time step used in the simulation is small enough then the additional adjustment of the face flux at t should allow better results in transient simulations.

4.2.4.3 backward

The other second-order scheme available uses the backward differencing approach and also neglects the variation of the flux value at the face of the cell. It is obtained using the Taylor series expansion of the flux values $\phi(t)$ and $\phi(t - \Delta t)$:

$$\phi(t) = \phi(t + \Delta t) - \frac{\partial \phi}{\partial t} \Delta t + \frac{1}{2} \frac{\partial^2 \phi}{\partial t^2} \Delta t^2 + O(\Delta t^3) \quad (4.9)$$

$$\phi(t - \Delta t) = \phi(t + \Delta t) - \frac{\partial \phi}{\partial t} (2\Delta t) + \frac{1}{2} \frac{\partial^2 \phi}{\partial t^2} (2\Delta t)^2 + O(\Delta t^3) \quad (4.10)$$

By rewriting (4.10) as

$$\phi(t - \Delta t) = \phi(t + \Delta t) - 2 \frac{\partial \phi}{\partial t} \Delta t + 2 \frac{\partial^2 \phi}{\partial t^2} \Delta t^2 + O(t^3) \quad (4.11)$$

the following relationship can be derived:

$$\frac{\partial \phi}{\partial t} = \frac{\frac{3}{2}\phi(t + \Delta t) - 2\phi(t) + \frac{1}{2}\phi(t - \Delta t)}{\Delta t} \quad (4.12)$$

Finally, this provides the final time discretised equation for the backward differencing method:

$$\frac{\frac{3}{2}\phi(t + \Delta t) - 2\phi(t) + \frac{1}{2}\phi(t - \Delta t)}{\Delta t} V_P + A\phi_f(t + \Delta t) - B(\text{div}\phi)_f(t + \Delta t) = S \quad (4.13)$$

4.2.5 Convection Discretisation

The convection terms in the general transport equation (4.1) can be discretised in several ways. Two of the most common methods, used in the studies herein, are described below.

4.2.5.1 upwind

The upwind discretisation approach is a first-order convection scheme that solves the differential equations using a differencing method that is biased towards the direction of the flux. The method can be described by considering a one-dimensional control volume in which the node P is centred, with neighbouring nodes E and W, shown in Figure 4.3.

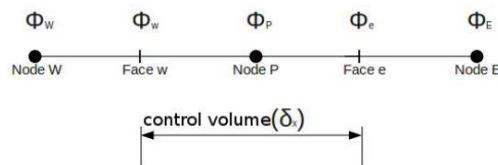


Figure 4.3 - upwind Discretisation Approach

The flux at Face E is determined by the direction of the flow:

$$\phi = \begin{cases} \phi_P & \text{if the flux is directed out of the control volume} \\ \phi_E & \text{if the flux is directed into the control volume} \end{cases} \quad (4.14)$$

This scheme is always bounded and can be used as an initial step in transient simulations in order to aid the convergence of the simulation.

4.2.5.2 linearUpwind

The accuracy of the first order upwind scheme can be improved by using an approach with second order accuracy such as the linearUpwind scheme. This method includes the first-

order estimation ϕ_P as before but also includes an additional correction in the form of an assumption of linear variation of flux between nodes P and W as shown in Figure 4.4.

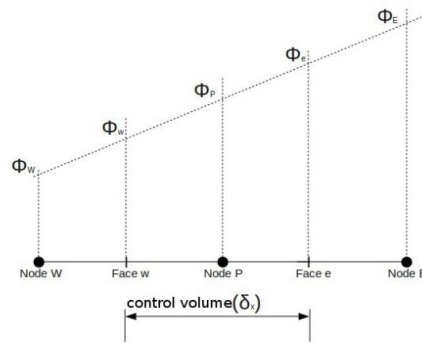


Figure 4.4 - linearUpwind Discretisation Approach

In this instance, the flux at the east face of the control volume can now be calculated from:

$$\phi_e = \phi_P + \frac{(\phi_P - \phi_W) \delta x}{\delta x} \frac{\delta x}{2} = \phi_P + \frac{1}{2}(\phi_P - \phi_W) \quad (4.15)$$

4.2.6 Arbitrary Mesh Interface (AMI)

An important technique used in problems involving one or more interfaces is the Arbitrary Mesh Interface (AMI). It allows modelling across mesh domains that are disconnected but adjacent, irrespective of whether stationary or moving relative to each other. The AMI methodology is available in the selection of the boundaries within the model being solved and can be used for unmatched/non-conformal cyclic patch pairs, mapped patches for coupling simulations between different mesh domains such as bulk and surface film flow as well as sliding interfaces for applications such as turbomachinery.

4.2.7 Solvers and Utilities

OpenFOAM contains a wide variety of solvers ranging from compressible and non-compressible to electromagnetism and combustion. Due to OpenFOAM's modifiable nature, users can, with an understanding of the C++ code and the relevant physics

associated to the problem, customise existing solvers to meet their needs. The following section outlines the solvers utilised within this study, highlighting the theory behind each as well as modifications made in order to satisfy the specific case study investigated.

4.2.7.1 Steady-State Solver: MRFSimpleFoam

MRFSimpleFoam is a steady state solver for incompressible turbulent flow and uses the SIMPLE algorithm for pressure-velocity coupling. Based on the Multiple Reference Frame (MRF) approach, it assumes that no relative mesh motion is present between the rotating and stationary components and is therefore often referred to as the Frozen Rotor approach. It approximates the solution by modelling the fluid zone within the rotating region as a rotating frame of reference, with the surrounding zones being treated as stationary frames.

The momentum equations are solved using a mixture of inertial and relative velocities in the relative frame with the inclusion of an additional Coriolis term for the rotating component as shown in (4.16) and (4.17).

$$\nabla \cdot (\vec{u}_R \otimes \vec{u}_R) + \vec{\Omega} \times \vec{u}_I = -\nabla \left(\frac{p}{\rho} \right) + \nu \nabla \cdot \nabla (\vec{u}_I) \quad (4.16)$$

$$\nabla \cdot (\vec{u}_I) = 0 \quad (4.17)$$

Whilst the transient effects of the flow are not taken into consideration, the MRFSimpleFoam solver provides fast results that can be used as initial conditions for the unsteady simulations.

4.2.7.2 Unsteady Solver: pimpleDyMFoam

This solver allows transient simulation of incompressible turbulent flow of Newtonian fluids in cases containing dynamic meshes and is included in the OpenFOAM-2.1.x distribution.

Based on the PISO pressure correction method, it provides accurate transient solutions but suffers from inefficient temporal time marching due a restriction on the maximum time step length. This ultimately results in a solver that is impractical for turbomachinery simulations.

4.2.7.3 Unsteady Solver: transientSimpleDyMFoam

Like pimpleDyMFoam, this solver allows transient simulation of incompressible turbulent flow of Newtonian fluids in cases containing dynamic meshes. However, it is distinctly different in that a SIMPLE-based time-stepping algorithm is utilised, with the turbulence model solution moved within the SIMPLE loop. This allows the solver to be more robust, allowing larger time steps to be prescribed with the knowledge that sufficient iterations will be conducted within each time-step.

4.2.7.4 Cavitation Solver: interPhaseChangeFoam

In order to predict the occurrence of cavitation in the simulations conducted during this work, a multiphase solver capable of modelling the creation and destruction of the vapour phase is required. In this instance, the interPhaseChangeFoam solver was used due to its capability of modelling two incompressible, isothermal immiscible fluids using the volume of fluid (VOF) phase fraction interface capturing approach. As investigated in Chapter 2, this type of model solves the governing equations based upon the fluid properties of the 'mixture' of the two phases. It also allows the usual turbulence models available within OpenFOAM to be used, including the $k-\omega$ SST model.

The governing equations used in this solver are not based upon mass continuity but instead by volume continuity as seen in (4.18).

$$\left(\frac{1}{\rho_m}\right)\frac{\partial p}{\partial \tau} + \nabla \cdot \mathbf{u} = (\dot{m}^+ + \dot{m}^-)\left(\frac{1}{\rho_l} + \frac{1}{\rho_v}\right) \quad (4.18)$$

where p is the pressure, ρ the density, \mathbf{u} the velocity, τ the pseudo-time derivative used for the time iterative solution technique and \dot{m}^+ and \dot{m}^- represent the mass flow between the phases.

The associated momentum equation is shown in (4.19).

$$\frac{\partial \rho_m \mathbf{u}}{\partial t} + \frac{\partial \rho_m \mathbf{u}}{\partial \tau} + \rho_m \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \mu_m \nabla^2 \mathbf{u} + \rho_m \mathbf{g} \quad (4.19)$$

where μ_m is the mixture dynamic viscosity and \mathbf{g} is the gravitational acceleration.

In order to track the interface between the liquid and vapour phases, a phase continuity equation is also employed as seen in (4.20).

$$\frac{\partial \alpha}{\partial t} + \left(\frac{\alpha}{\rho_m}\right)\frac{\partial p}{\partial \tau} + \frac{\partial \alpha}{\partial \tau} + \nabla \cdot \alpha \mathbf{u} = (\dot{m}^+ + \dot{m}^-)\left(\frac{1}{\rho_l}\right) \quad (4.20)$$

The `interPhaseChangeFoam` solver provides three options by which to calculate the mass transfer rates \dot{m}^+ and \dot{m}^- . These are housed in the `phaseChangeTwoPhaseMixtures` sub-directory and are known as *Kunz*, *Merkle* and *SchnerrSauer*, representing the cavitation modelling techniques, discussed in Chapter 2, that the OpenFOAM code is based upon [31] [34] [35].

4.2.7.4.1 Cavitation Solver: `interPhaseChangeDyMFoam`

Whilst the `interPhaseChangeFoam` solver is very useful, additional code is required in order to properly assess cases involving moving meshes, resulting in the `interPhaseChangeDyMFoam` solver that was included for the first time in the OpenFOAM 2.2.x. This solver was therefore used to conduct the initial work undertaken in relation to modelling cavitation the centrifugal pump.

4.2.7.5 Utility: turboPerformance

This useful library was developed by prominent OpenFOAM contributors from the Turbomachinery Special Interest Group (SIG); Mikko Avuvinen (Helsinki University of Technology, Finland); Hakan Nilsson (Chalmers University of Technology, Sweden); and David Boger and Bryan Lewis (Penn State University, USA).

It provides the capability of calculating various properties of particular interest in turbomachinery [62]. By specifying simple items of information in the *controlDict* such the names of the inlet and outlet patches, rotating regions such as impellers as well as the density of the fluid and rotational velocity, the utility calculates related forces and information within the case. Of great significance is the additional ability to print generated head, absorbed power, hydraulic power and hydraulic efficiency of the unit either on the screen or graphically via plotting tools such as *gnuplot*. Not only does this allow the user to establish whether the problem is being solved correctly, it also allows a visual way of determining if the solution has properly converged. This was used in all simulations performed in order to obtain the relevant performance output figures for comparison.

4.3 Centrifugal Pump Case Study

The specific test case considered in the following simulations is based on a model of a Weir Warman Horizontal Slurry Pump 8/6 AH, the construction options of which are seen in Figure 4.5. Whilst this type of centrifugal pump is typically used to transport slurries in the minerals industry, performance test data available for clear water was used to determine the accuracy of the simulations performed in OpenFOAM and the commercial solver ANSYS CFX [63].

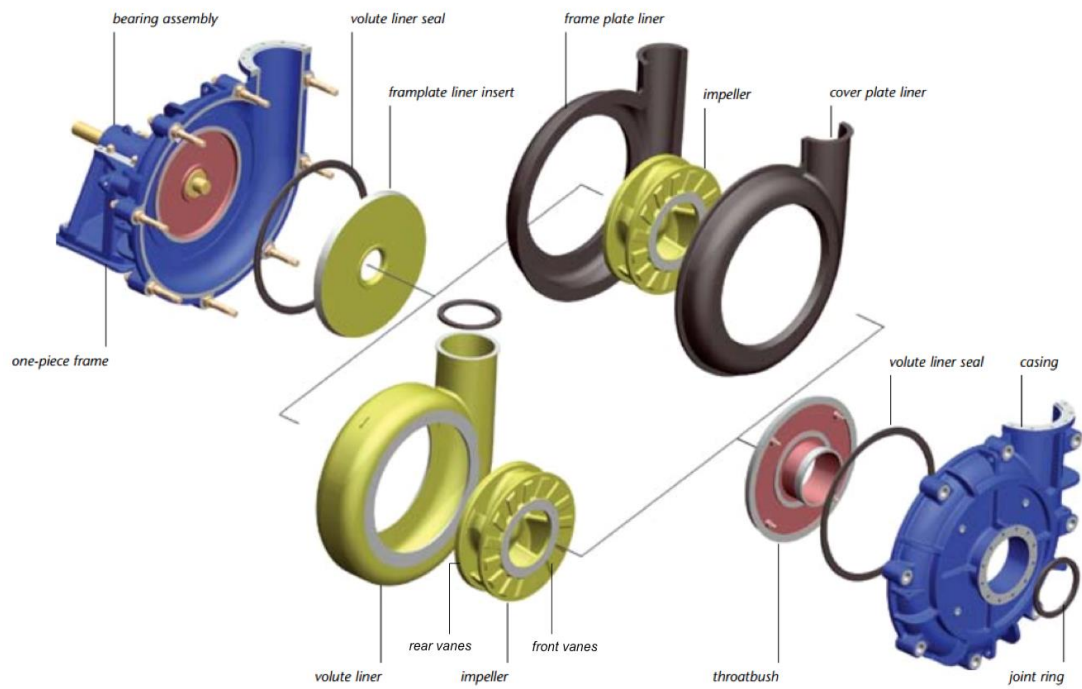


Figure 4.5 - Construction Options for Weir Warman 8/6 AH

The general specifications and operating conditions are also given in Table 4-2. During the analyses, the best efficiency flow rate, Q_{opt} , was used for initial studies, before examining off-design conditions at 70% and 130% Q_{opt} .

Table 4-2 - Pump Specification and Operating Conditions

Number of Vanes	4
Vane diameter	0.536 m
Inlet (throat bush) diameter, D_1	0.203 m
Inlet (throat bush) diameter, A_1	0.032 m ²
Outlet (discharge pipe) diameter, D_2	0.152 m
Outlet (discharge pipe) diameter, A_2	0.018 m ²
Rotational Speed, n	1100 rpm
Best Efficiency Flow, Q_{opt}	235 l/s (0.235 m ³ /s)
Generated Head @ Q_{opt} , H_{opt}	53.8 m
Fluid Density, ρ	997 kg/m ³
Fluid Kinematic Viscosity, ν	8.9x10 ⁻⁴ N s/m ²
Fluid Temperature, T	298 K

4.3.1 Computational Mesh

The mesh used in this case study was created in ICEM, using a hexahedral mesh of 2.1 m cells across three constituent parts: throat bush, impeller and volute casing as seen in Figure 4.6 which includes the extension of the suction and discharge pipework in order to improve accuracy and stability of the simulation. The mesh utilised here was selected following a previous mesh sensitivity study conducted under related work which compared three meshes with differing levels of refinement [64]. Each differed in quality based on the number of prismatic layers at the wall locations; three, five and seven, which yielded average y^+ values of 900, 150 and 130 respectively. The results of the three meshes were compared using a cross-section of the overall pump including the rotating and stationary components, where static and total pressure contours and velocity contours and streamlines were reviewed. Overall, the medium and high quality meshes were comparable, most noticeably in terms of recirculation zones in the impeller where the low quality mesh yielded differing results. Given that the high quality mesh took 67% longer to solve than the medium quality mesh, the decision was taken to use the medium quality mesh with the five prismatic layers in the subsequent studies.

More detail of the mesh, including the five prismatic layers at the walls to counteract potential false diffusion at these boundaries can also be seen in Figure 4.7.

Using the approach of meshing each domain individually allowed the impeller to be treated as a rotating region in the transient simulations, with rotor-stator interfaces defined between the throat bush and the impeller and the volute and impeller respectively.



Figure 4.6 - Overall Case Mesh Construction

Note that the front and back vanes connected to the impeller in the physical pump were not included and the flow between the front and rear plates and the volute as well as the leakage through the wear rings was not modelled in the following studies. This was due to the additional size and complexity of the mesh and the associated increased computational effort required as well as the overall objective in this instance which had a strong focus on the feasibility of performing the simulations, both single-phase and two-phase using OpenFOAM.

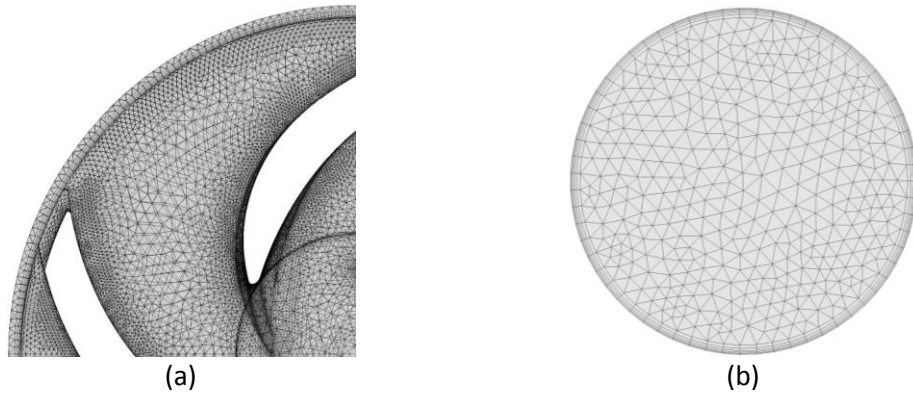


Figure 4.7 – (a) Impeller Mesh and (b) Discharge Pipe Mesh Highlighting Inflation Layers

In order to use these meshes in OpenFOAM, they must first be converted into a format that can be read. This was done by converting the *.msh* files using the ‘fluentMeshToFoam’ command for each component. Following this, ‘zones’ were created for each component in order to define the stationary and rotating parts and the meshes were merged together to construct the overall pump domain. Finally, the boundary conditions were set up for the velocity, pressure, k and ω fields, using the cyclicAMI condition for the interfaces between components.

4.3.2 Boundary Conditions

Before any simulations were carried out, the selection of appropriate boundary conditions was required. Both steady state and transient simulations of the single-phase fluid were performed using the robust approach of using velocity inlet and static pressure outlet boundary conditions. Alternatively, for the two phase simulations, a total pressure inlet and velocity outlet boundary condition pair was employed, with the total pressure being reduced in order to vary to NPSH conditions.

In relation to using the k - ω SST turbulence model, the values of k and ω are also defined at the inlet boundary.

The specific kinetic energy, k , was calculated using the equation

$$k = 1.5(U \cdot I)^2 \quad (4.21)$$

where U is the inlet velocity in m/s and I is the turbulence intensity as a percentage.

The specific turbulence dissipation rate, ω , was calculated using

$$\omega = C_\mu^{-0.25} \frac{\sqrt{k}}{L} \quad (4.22)$$

where C_μ is a constant of 0.09, k is the specific kinetic energy calculated above and L is the characteristic length, taken to be 10% of the inlet hydraulic diameter [65].

A typical turbulence intensity of 5% was used during all analyses [65]. Using the example of an inlet velocity of 7.261 m/s calculated from the best efficiency flow rate, Q_{opt} , of 235 l/s through the 0.203 m inlet produces:

$$k = 1.5(7.261 \cdot 0.05)^2 = 0.198 \text{ m}^2/\text{s}^2 \quad (4.23)$$

$$\omega = 0.09^{-0.25} \frac{\sqrt{0.198}}{0.0203} = 39.989 \quad (4.24)$$

In terms of the wall treatment of components, with the average y^+ of 150 the standard wall functions contained in OpenFOAM were used for k and ω , named `kqRwallFunction` and `omegaWallFunction`.

The conditions used for the following studies are as contained in Table 4-3.

Table 4-3 - Turbulence Conditions Used at Inlet Boundary

% Q_{opt}	Volume Flow Rate (l/s)	Inlet Velocity (m/s)	k (m^2/s^2)	ω (s^{-1})
70%	164.5	5.083	0.097	27.993
100%	235	7.261	0.198	39.989
130%	305.5	9.439	0.334	51.986

Using the Q_{opt} condition of 235 l/s, the boundary conditions used in OpenFOAM are summarised in Table 4-4. For all other flow rates, these were modified accordingly based on Table 4-3.

Table 4-4 – Summary of Boundary Conditions (Single-Phase)

Inlet	
U	$U_{opt(in)} = \frac{Q_{opt}}{A_1} = \frac{0.235}{0.032} = 7.261 \text{ m/s}$
p	'zeroGradient' i.e. normal gradient of p is zero
k	$k = 1.5(7.261 \cdot 0.05)^2 = 0.198 \text{ m}^2/s^2$
ω	$\omega = 0.09^{-0.25} \frac{\sqrt{0.198}}{0.0203} = 39.989 \text{ s}^{-1}$
Outlet	
U	'zeroGradient' i.e. normal gradient of U is zero
p	'fixedValue' of 1000 (p/ ρ)
k	'zeroGradient' i.e. normal gradient of k is zero
ω	'zeroGradient' i.e. normal gradient of ω is zero

4.3.3 Post-processing in ANSYS CFD-Post

Whilst the main objective of the work was to establish whether OpenFOAM is a viable alternative to commercial codes or a worthy addition, one of the most powerful aspects of some licensed software is the post-processing capabilities such as that of ANSYS CFD-Post. In relation to pumps and other turbomachinery, ANSYS also has a specific post-processing tool built directly into the software which is known as the 'Turbo' tool. For this particular work it was decided that ANSYS Turbo would be used as a method of comparing the results of the simulations conducted in both OpenFOAM and ANSYS CFX as it would allow direct comparisons to be made.

In order to do so, some additional steps were required to convert results from OpenFOAM to ANSYS CFD-Post using the *foamMeshToFluent* and *foamDataToFluent* commands. Further details on how this was undertaken are contained in *APPENDIX C – OpenFOAM User Guide*.

Discussing the options available to the user within ANSYS CFD-Post further, the Turbo tool provides the opportunity for specific post-processing operations that are useful in the design and optimisation processes associated with turbomachinery. In addition to automated macros and reporting templates specific to devices such as pumps, fans and compressors, results can also be presented in a useful manner in the meridional view (Figure 4.8) and unrolled blade-to-blade views at any span position between the hub and shroud. There is also the option for viewing blade loading plots (Figure 4.9) and review results plotted from inlet to outlet and from hub to shroud.

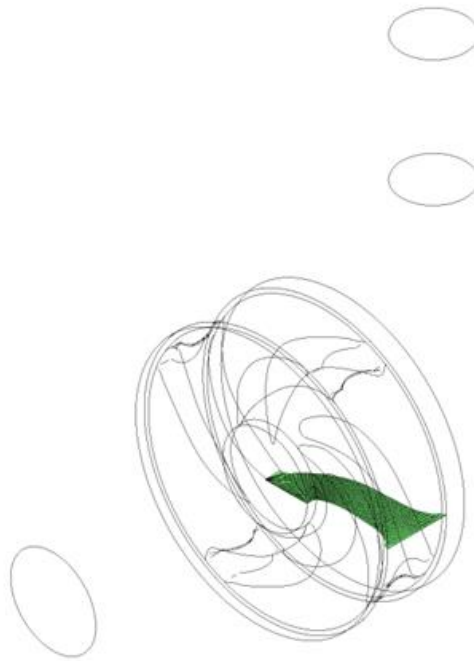


Figure 4.8 – Meridional Profile Automatically Selected within Case Study

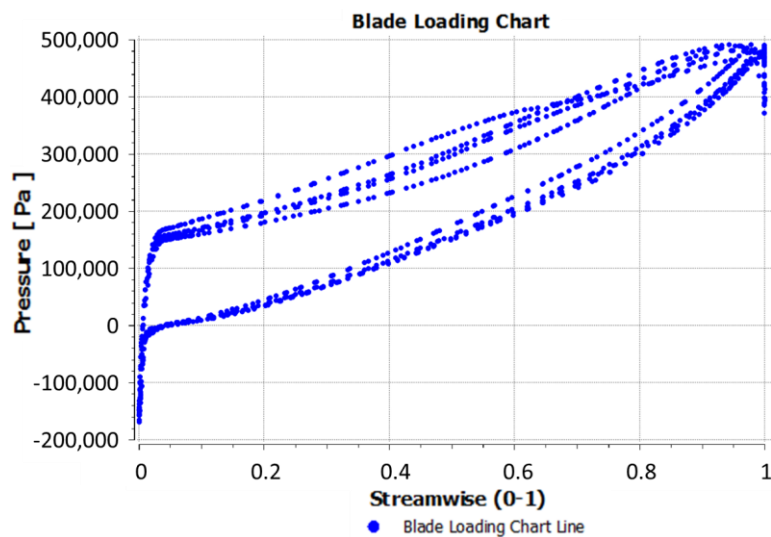


Figure 4.9 – Example of Blade Loading (Static Pressure)

The main tool used in this work was the capability to assess results from the impeller hub to the shroud at three different points through from the inlet to the outlet of the impeller (15%, 50% and 85%) for values of pressure, velocity and turbulence, using 25 points from the hub (0% span) to the shroud (100% span) and calculated using the area averaging process as shown in Figure 4.10.

Hub-to-Shroud Averaging
(Equal Distance)

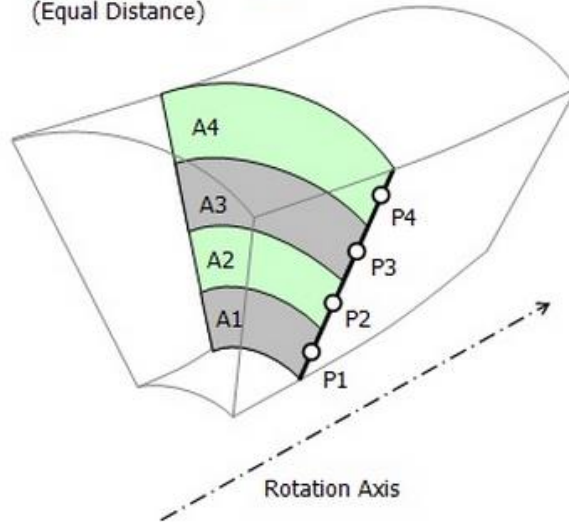


Figure 4.10 – Hub to Shroud Averaging by Area [66]

Additionally, the CFD-Post software was also used to assess these properties along a chosen line in the casing, selected to be close to the cutwater region as shown in Figure 4.11 as Line A-A.

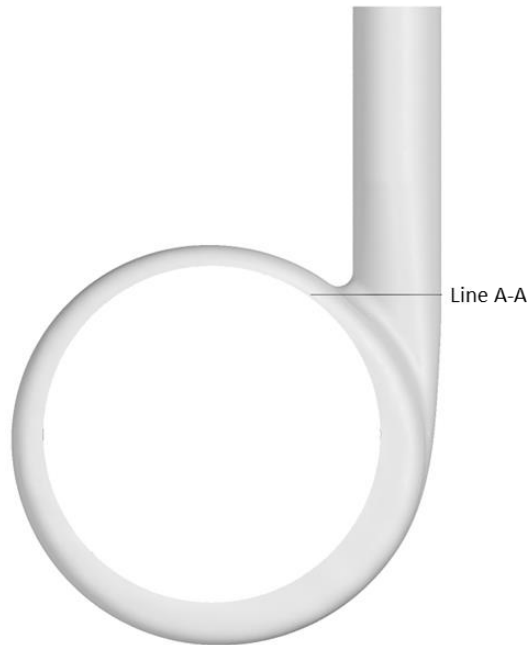


Figure 4.11 - Line A-A used for Reviewing Results in Volute Casing

4.4 Conclusions

In order to provide the relevant background to the case studies undertaken, the most significant aspects of the CFD calculations, setup and approach were outlined.

Following an introduction on the governing equations and turbulence model used in the centrifugal pump case study, the majority of the subsequent discussion surrounded the OpenFOAM code. This initially involved introducing the OpenFOAM platform by outlining the structure of a typical case study and was taken a stage further by discussing the time and convection discretisation schemes that are used in the following analyses to determine the best approach to modelling a centrifugal pump.

In relation to the particular examples to be reviewed, each of the solvers used throughout the subsequent work were outlined for steady state, transient and two-phase analyses. The turboPerformance utility was also introduced, used to allow the calculation and tracking of performance data associated with the pump simulations and assist the confirmation of solution convergence.

In order to give an overall appreciation of the centrifugal pump case study in question, more specific details were outlined in terms of the construction and geometry of the pump itself, the computational mesh used, determination of the boundary conditions to be prescribed and post-processing approach using ANSYS CFD-Post.

Chapter 5

5. Single-Phase Modelling

This chapter will discuss in detail the use of OpenFOAM to model single-phase flow in the Warman 8/6 AH pump using both steady state and two transient solvers. It will outline the investigation to establish the most effective configuration in OpenFOAM for this problem by examining sensitivity to time step size, time discretisation scheme, number of internal corrector iteration loops per time step and velocity convection scheme.

The results obtained using OpenFOAM were converted in order to be post-processed in ANSYS CFD-Post as mentioned in Section 0 which also facilitated a final comparison the ANSYS CFX commercial solver. This is of particular interest as the ANSYS CFX simulation software is considered to be the standard simulation tool for modelling centrifugal pumps within the pump industry, with the results being accepted by industry users.

5.1 Frozen Rotor Simulation

The first step in understanding the effectiveness of using OpenFOAM to model the Warman AH 8/6 centrifugal pump was to perform single-phase simulations using clear water at the best efficiency flow, Q_{opt} , of 235 l/s. The initial step in this process was to run a steady state, or frozen rotor, simulation in order to obtain an approximate solution. Only once this was completed could the more accurate transient simulation be examined, using the steady state results as initial values.

By establishing the most effective method of simulating the pump, off-design conditions of 70% and 130% Q_{opt} could then be investigated and compared against experimental data.

In order to generate the initial results from the simulation, the frozen rotor approach was used, using MRFSimpleFOAM solver, based on the multiple reference frame method. Whilst the results from such simulations are not completely accurate, they do enable a quick validation against the experimental data in order to ensure appropriate initial conditions are used in the transient simulations.

In order to model the rotating frame in the correct manner, an *MRFZones* file was created in the *constant* directory. This allows the 'rotating' region to be defined, with the rotating velocity being 115.19 rad/s (1100 rpm) in this instance.

In order to allow the assessment to be made using multiple cores, the decomposePar tool built in to OpenFOAM was used.

The simulations were set up to allow the best possible potential for a converged solution in this first instance during these simulations, using a 'zeroGradient' inlet and 'fixedValue' outlet for velocity, with 'uniform' total inlet pressure and 'zeroGradient' pressure at the outlet as outlined in Section 4.3.2. As the flow rate is set, this particular setup allows the inlet pressure to be predicated, thus predicting the generated head of the pump.

Prior to the use of the MRFSimpleFoam solver, the simple potential flow solver potentialFoam was used to initialise the starting fields for the simulation.

The setup conditions are outlined in Table 5-1.

Table 5-1 - Summary of Setup Conditions for Frozen Rotor Simulation

Time discretisation scheme	steadyState	
Convection discretisation schemes		
U	linearUpwind	
k, omega	upwind	
Solvers		
p	solver	GAMG
	Smoother	GaussSeidel
	Tolerance	1×10^{-8}
	relTol	0.05
U, k, omega	solver	smoothSolver
	Smoother	GaussSeidel
	Tolerance	1×10^{-7}
	relTol	0.1

Figure 5.1 shows the level of convergence for the frozen rotor simulation which was stopped after 5000 iterations with all the residuals levelling below or marginally above 1×10^{-3} . Due to the simple nature of the solver, the simulation ran quickly, taking 3.5 hours using eight cores.

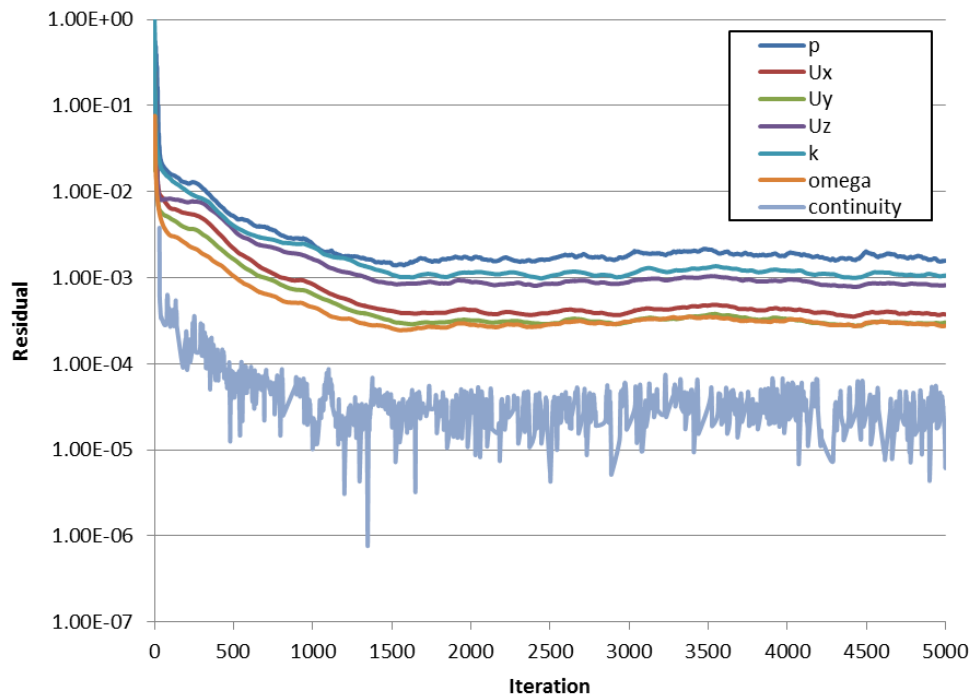


Figure 5.1 – MRFSimpleFoam Residual Convergence

Of interest are Figure 5.2 and Figure 5.3 which show the convergence of the generated head and hydraulic power, absorbed power and hydraulic efficiency respectively. The total generated head from experimental data has also been included for reference purposes [63].

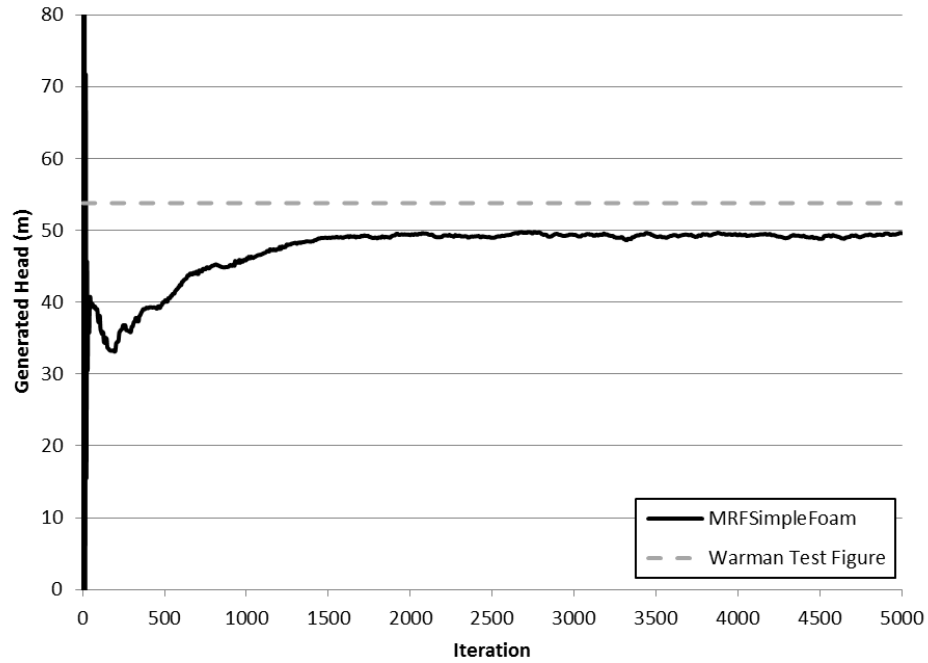


Figure 5.2 – Convergence of Pump Generated Head

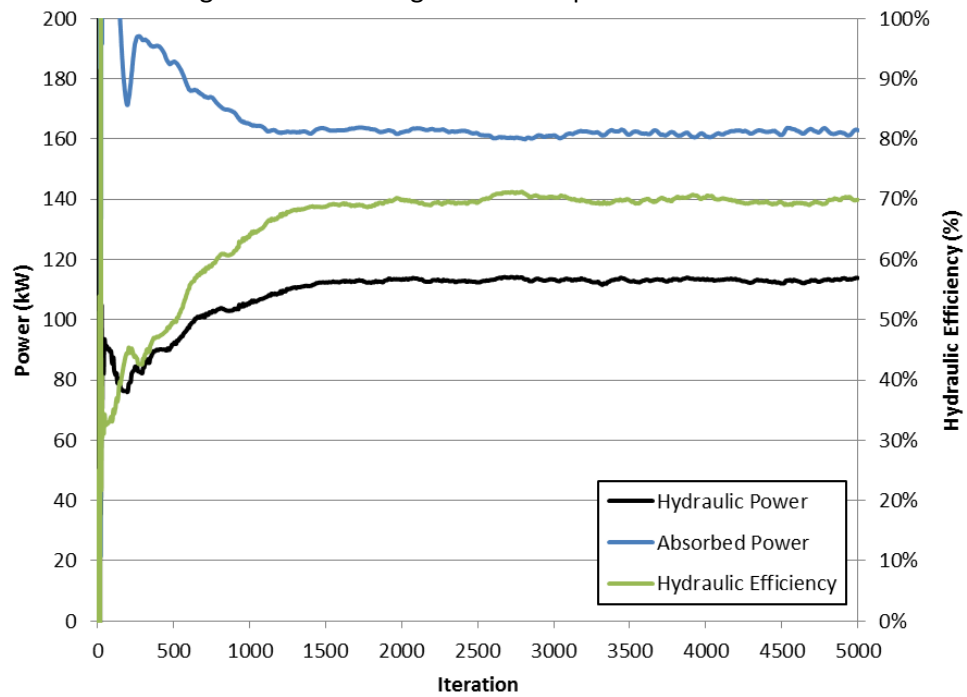


Figure 5.3 - Convergence of Pump Hydraulic Efficiency

It is clear that the MRFSimpleFoam steady state solver underestimates the total head developed by the pump, with the final value of 49.8 m reflecting a 6.9% error from the experimental value of 53.5 m. However this was expected due to the 'snapshot' approach that the solver takes, only producing a result for the fixed position of the mesh and unable to model the variation in generated head as the impeller blade passes a given reference point. The absorbed power and hydraulic efficiency also appear to be within a realistic range; however it is worth noting at this stage that the hydraulic efficiency computed using the turboPerformance utility is based simply on the hydraulic power generated and, due to the simplified model, only the power absorbed by the internal fluid passages of the impeller as the front and rear shrouds/wear plate surfaces are not included. Therefore, the efficiency values provided in the following discussions are greater than the experimental efficiency values from the Warman test data as these also consider the other components of overall pump efficiency; mechanical efficiency and volumetric efficiency. However, they are still extremely useful in determining the effectiveness of the set ups examined.

This is reinforced by the plots of static pressure, total pressure and velocity in the stationary frame shown in Figure 5.4 which have been post-processed using the ANSYS CFD-Post tool following conversion from the OpenFOAM format. This particular slice was taken to be in line with the centre of the impeller but also includes the casing to highlight the interaction between the rotating and stationary components. It can be seen that pressure is generated through the impeller as expected. Of particular interest is the velocity plot which shows a region of reduced velocity on the right hand side of the impeller looking at the image. This is attributed to the cutwater region and 'frozen' nature of the impeller in this simulation.

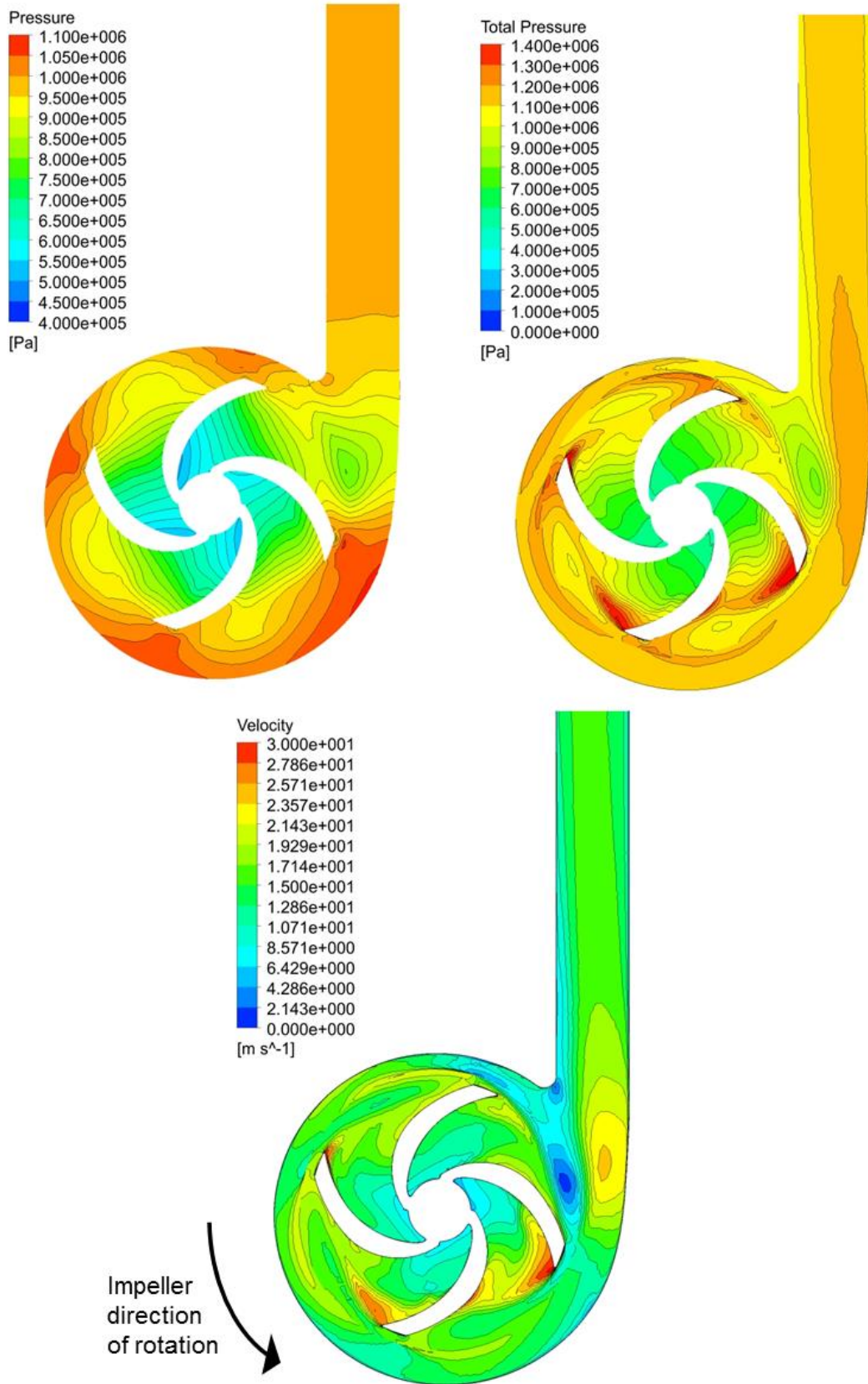


Figure 5.4 – Contour Plots of Static Pressure (top left), Total Pressure (top right) and Velocity in the Stationary Frame (bottom)

Following the completion of this initial simulation, the accuracy of the results was deemed to be sufficient to allow the data to be used as initial values in the more involved transient simulations that would be required in order to assess the performance in greater detail.

5.2 Transient Simulations

The next phase in the process saw the converged solution from the Q_{opt} frozen rotor simulation utilised by the transient solvers available in OpenFOAM.

This section discusses in detail the construction of these OpenFOAM cases, the effects of varying several setup and solver parameters before comparing the results to those obtained through the use of ANSYS CFX.

5.2.1 Setup

In order to progress onto the transient simulations in OpenFOAM, a few important modifications are made. Firstly, unlike the frozen rotor simulations that utilise the *MRFZones* file in the constant directory to specify the 'rotational' properties of the impeller zone, a file called *dynamicMeshDict* was created in which the true dynamic properties of this zone are defined i.e. the rotational velocity and centre of rotation.

The *U* file within the time directory that contains the velocity data for the fields within the model also required alteration such that the boundary type for the impeller walls was changed from *fixedValue* to *movingWallVelocity*.

A summary of the setup and solution properties for the various studies that follow relating to single-phase flow is summarised in Table 5-2.

Table 5-2 - Summary of Setup Conditions for Transient Simulations

Transient Solver	transientSimpleDyMFoam/pimpleDyMFoam	
Time discretisation scheme	Euler/CrankNicholson/backward	
Time Step (s) (per revolution of impeller)	1.36x10 ⁻³ /5.45x10 ⁻⁴ /2.73x10 ⁻⁴ (40/100/200)	
Correctors		
nCorrectors	1 (not required for transientSimpleDyMFoam)	
nOuterCorrectors	5/10/20/40	
nNonOrthogonalCorrectors	0	
Convection discretisation schemes		
U	upwind/linearUpwind	
k, omega	upwind	
Solvers		
p,pcorr,pFinal	solver	GAMG
	smoother	GaussSeidel
	tolerance	1x10 ⁻⁸
	relTol	0
U, k, omega	solver	BiCGStab
	preconditioner	DILU
	tolerance	1x10 ⁻⁷
	relTol	0
Under-relaxation		
p	0.3	
U	0.7	
k, omega	0.4	

5.2.2 Solver Capability

The initial simulations were aimed at comparing the two solvers previously discussed, pimpleDyMFoam and transientSimpleDyMFoam, in order to determine which was the most applicable for this particular purpose and would be used in the sensitivity studies that followed. The influence of the number of time steps per revolution was also investigated at this stage, using values of 40, 100 and 200 (time steps of 1.36x10⁻³ s, 5.45x10⁻⁴ s and 2.73x10⁻⁴ s respectively).

During this initial basic investigation, all other parameters remained constant with first order upwind velocity and turbulence convection, Euler time discretisation scheme and 20 corrector loops per time step.

Following the appropriate modifications of the case files using Table 5-2, the results from the frozen rotor simulation were used as the initial data from which the transient simulations were started using both transientSimpleDyMFoam and pimpleDyMFoam, with each simulation using eight cores. Whilst there were differences in the time taken to reach a converged solution, the analyses typically lasted between five and seven impeller revolutions.

The raw time-varying performance data for the impeller and complete pump obtained using the turboPerformance utility over the final revolution of the pump impeller is shown in Figure 5.5 to Figure 5.9 as they were considerably out with an acceptable range, believed to be due to an issue with the interface between the impeller and casing in this instance. The use of 'T' and 'P' refer to transientSimpleDyMFoam and pimpleDyMFoam respectively and the corresponding numbers refer to the number of time steps per revolution of the impeller. The averaged data over this last impeller revolution is summarised in Table 5-3.

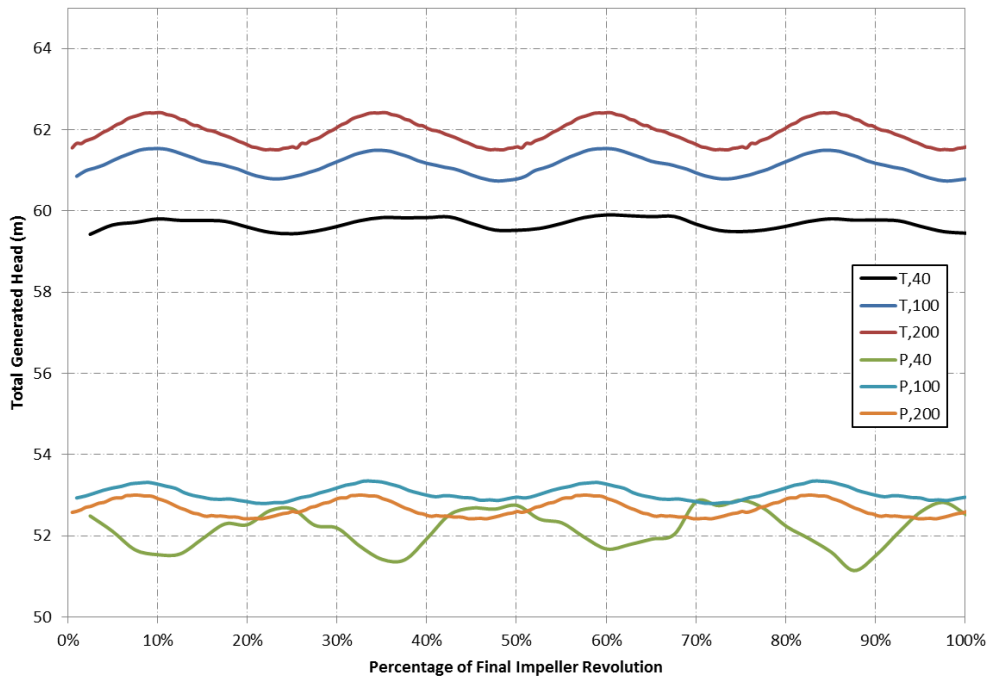


Figure 5.5 - Head Generated by Impeller during Final Revolution of Impeller (Study 1)

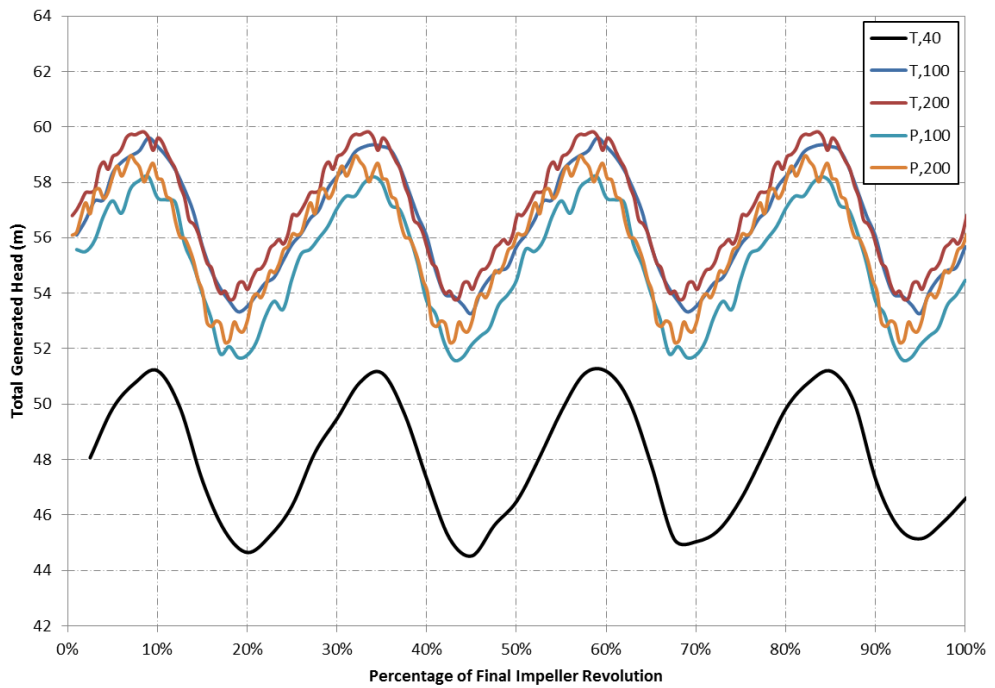


Figure 5.6 – Head Generated by Pump during Final Revolution of Impeller (Study 1)

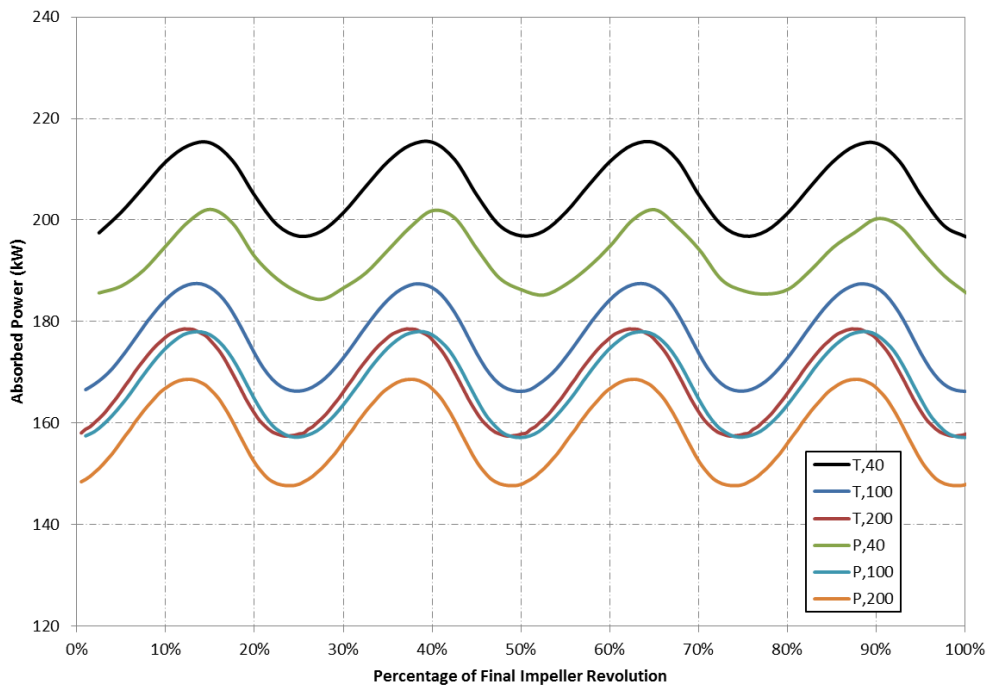


Figure 5.7 – Absorbed Power during Final Revolution of Impeller (Study 1)

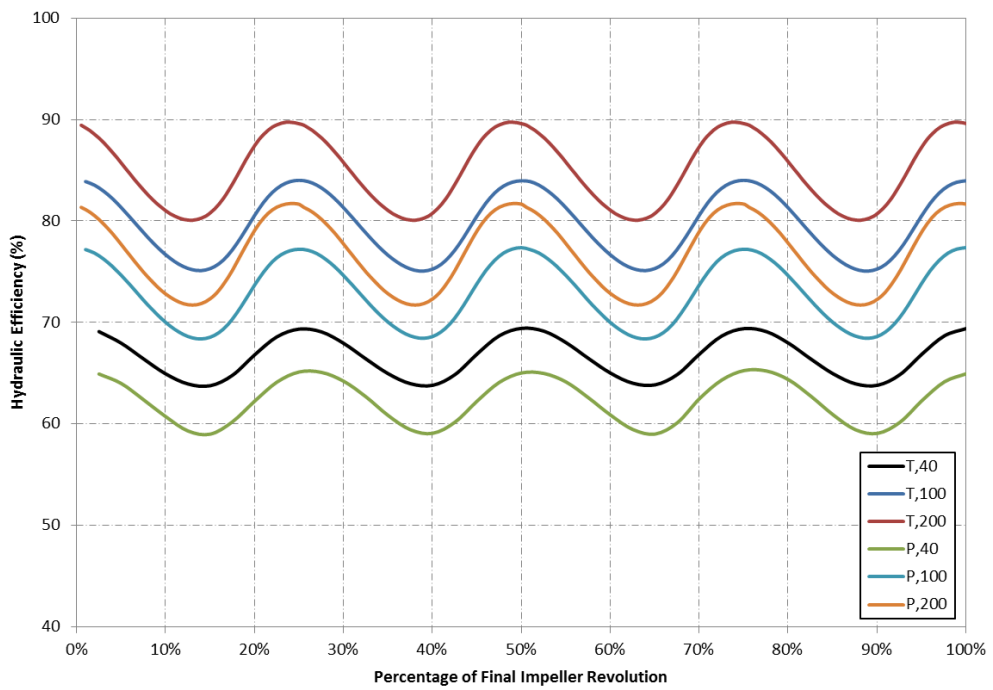


Figure 5.8 – Impeller Hydraulic Efficiency during Final Revolution of Impeller (Study 1)

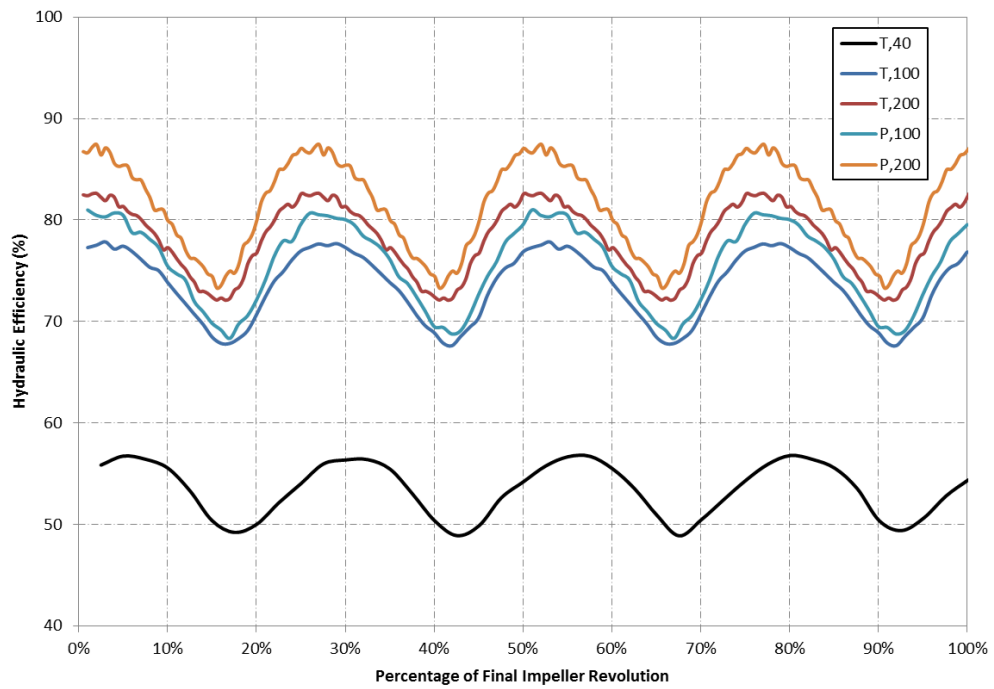


Figure 5.9 – Pump Hydraulic Efficiency during Final Revolution of Impeller (Study 1)

Table 5-3 - Summarised Results for Varying Solver and Time Steps

Setup	Impeller				Pump			Run Time Per Impeller Rev. (hrs)
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)	
T,40	59.7	137.0	206.0	66.5	47.7	109.4	53.1	0.9
T,100	61.1	140.4	176.8	79.4	56.4	129.4	73.2	2.1
T,200	61.9	142.2	167.9	84.7	56.8	130.2	77.5	4.1
P,40	52.2	119.8	192.5	62.2	-34.8	-79.8	-41.5	0.7
P,100	53.0	121.8	167.5	72.7	55.1	126.4	75.5	1.2
P,200	52.7	120.9	148.0	81.7	55.8	128.1	86.6	2.1

Prior to assessing the results, the experimental data showing a total generated head of 53.5 m at the Q_{opt} was considered. As this simulation has been simplified from the physical as-built design with the removal of the front and rear plates, exclusion of leakage paths and negligibility of surface roughness, the expectation of the results was that the generated head would be over-predicted in comparison to the experimental data at a given flow rate.

The initial review of the summarised performance in Table 5-3 and Figure 5.5 to Figure 5.9 shows that in each instance where the largest time step was used (40 per revolution), the results were not accurate enough.

Whilst the T,40 setup did yield reasonable results for the impeller, the total pump generated head of 47.7 m was deemed to be unsatisfactory at this stage as it was below the experimental result as well as considerably below the values seen in the studies with smaller time steps. The P,40 setup predicted a lower impeller head of 52.2 m but the main issue was in relation to the predicted pump head which was -34.8 m, with the larger time setup appearing to cause issues with the interaction between the impeller and casing. Whilst attempts were made to resolve this, the issue remained.

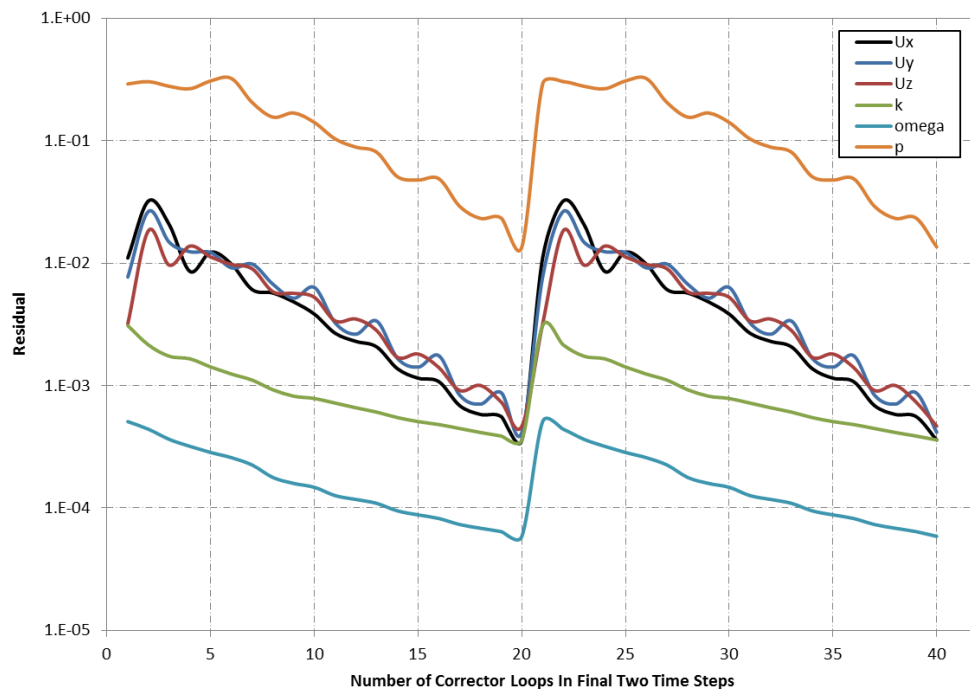
The other two assessments conducted using the pimpleDyMFoam solver, P,100 and P,200, predicted the head generated by the impeller at 53.0 m and 52.7 m respectively, more than 10% lower than the transientSimpleDyMFoam result. These figures are also lower than the overall experimental figure seen for the pump which is clearly not correct. This is only compounded by the fact that the total generated head predicted for the pump in the P,100 and P,200 studies was higher than the figures predicted for the impeller. Again, this is not physically possible as there should be losses associated with the casing, resulting in a decrease in this parameter between the impeller outlet and casing outlet. Based on these issues, the pimpleDyMFoam solver was not considered further.

The other two remaining cases, T,100 and T,200 were similar in their results across the impeller and overall pump assessments. In terms of the impeller results, the difference between the results for generated head, absorbed power and hydraulic efficiency were 1.3%, 5.2% and 6.5% respectively. The differences for the generated head and hydraulic efficiency in relation to the overall pump were 0.7% and 5.7% respectively.

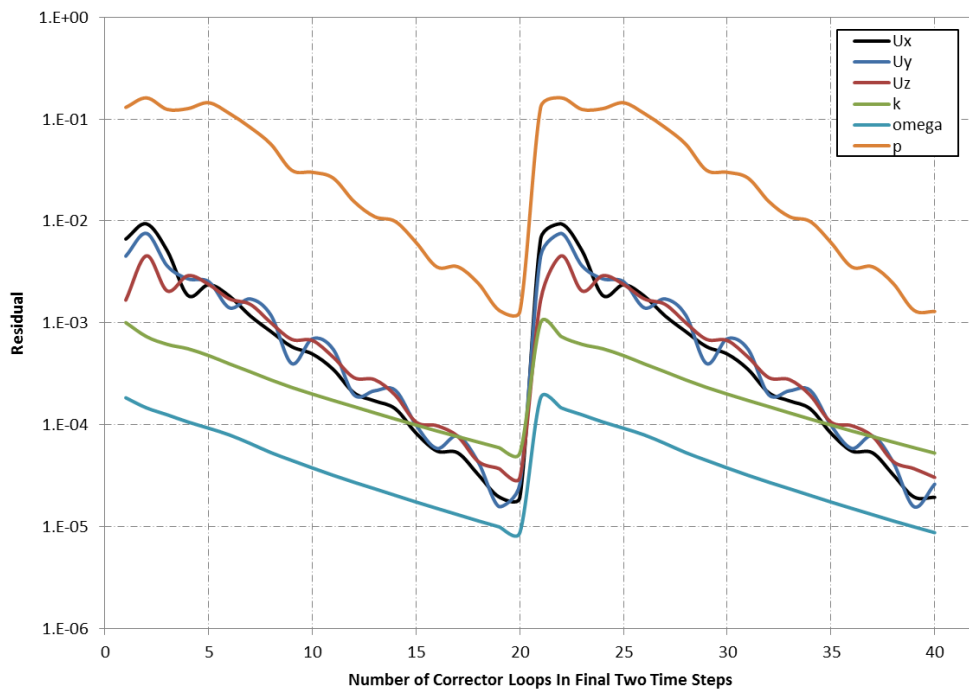
In order to provide a greater understanding of how the set ups compared, flow through the impeller and volute casing was examined as outlined in Section 0. This included comparing circumferentially averaged values of static pressure, total pressure, velocity and turbulent

kinetic energy in the impeller fluid passages and across the defined location at the cutwater region. By reviewing this information it was seen that in all cases the results for each T,100 and T,200 simulation were similar, particularly in relation to the velocity and turbulent kinetic energy through the impeller and static and total pressure within the volute casing close to the cutwater. Data supplementing this can be found in **Figure A1** to **Figure A8**.

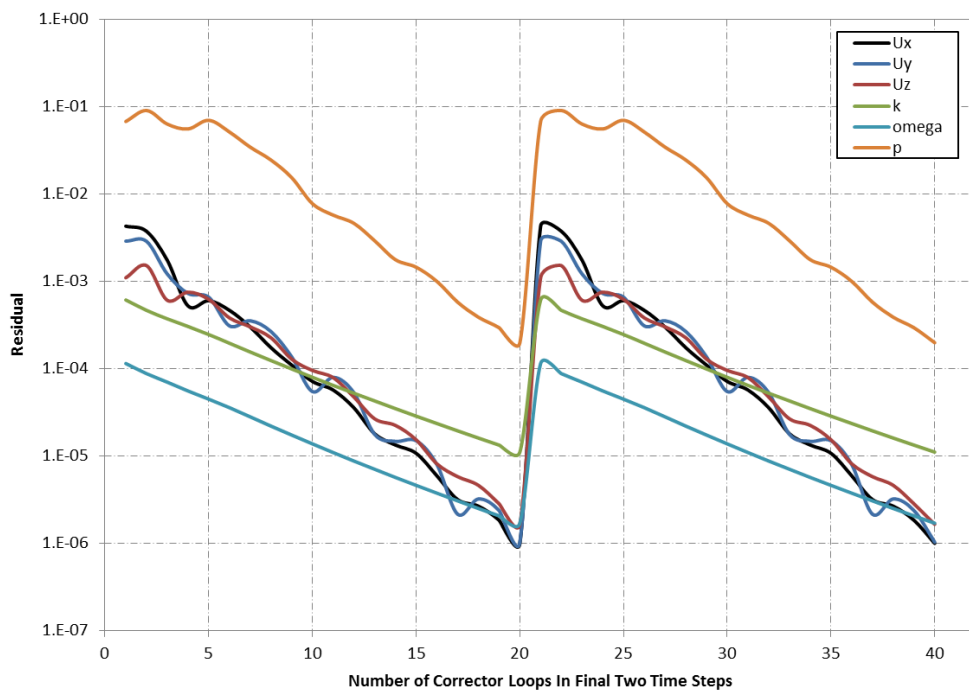
In terms of convergence, Figure 5.10 (b) and Figure 5.10 (c) show the comparison between the T,100 and T,200 setups over the final two time steps. It can be seen that, whilst there is a level of improved convergence using the smaller time step, the gain is not significant.



(a)



(b)



(c)

Figure 5.10 – Convergence Levels using transientSimpleDyMFoam with Time Step of (a) 1.36×10^{-3} s (b) 5.45×10^{-4} s and (c) 2.73×10^{-4} s

Following review of all of the data obtained during this particular assessment, it was decided that the T,100 would be used for the subsequent simulations. Whilst there were

differences in the predicted performance values and levels of convergence between the T,100 and T,200 set ups, when the extent of these were compared alongside the difference in associated run time (2.1 hours and 4.1 hours per impeller revolution respectively) the T,100 approach was deemed to be a more appropriate setup to build upon and refine.

5.2.3 Number of Pressure Correcting Loops (nOuterCorrectors)

Following the determination of the most effective solver and time step size in the previous section, the parameter investigated in this next study relates to the number of corrector loops within each of these time steps, nOuterCorrectors. With the default value of 20 being used in the previous study, values of 5, 10 and 40 were investigated. As with the previous study, all other parameters remained constant.

The raw time-varying performance data for the impeller and complete pump obtained using the turboPerformance utility over the final revolution of the pump impeller was once again reviewed, with the summarised data shown in Table 5-4.

Table 5-4 - Summarised Results for Varying Number of Internal Corrector Loops

nCorr	Impeller				Pump			Run Time Per Impeller Rev. (hrs)
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)	
5	49.0	112.4	161.6	69.6	41.2	94.7	58.6	0.6
10	61.2	140.6	178.7	78.7	39.9	91.6	51.3	1.1
20	61.1	140.4	176.8	79.4	56.4	129.4	73.2	2.1
40	61.2	140.4	176.7	79.5	56.1	128.8	72.9	3.8

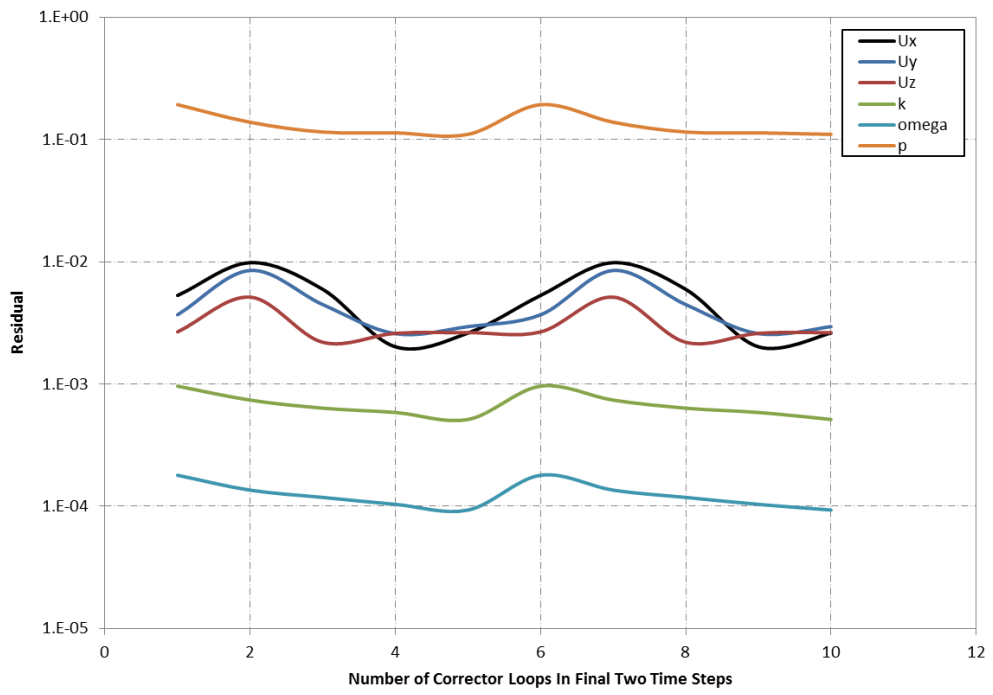
Assessing the impeller performance first, the data in Table 5-4 show that the values of generated head and absorbed power at 49.0 m and 161.6 kW are lower than those for the initial study which utilised 20 corrector loops. However, the values for the studies using 10, 20 and 40 corrector loops are similar. Comparing the studies using 10 and 20 corrector loops, the differences are 0.2%, 1.1% and 0.9% for generated head, absorbed power and

hydraulic efficiency respectively. This difference is even smaller between the studies with 20 and 40 corrector loops, with differences of 0.2%, 0.0% and 0.1% for generated head, absorbed power and hydraulic efficiency respectively.

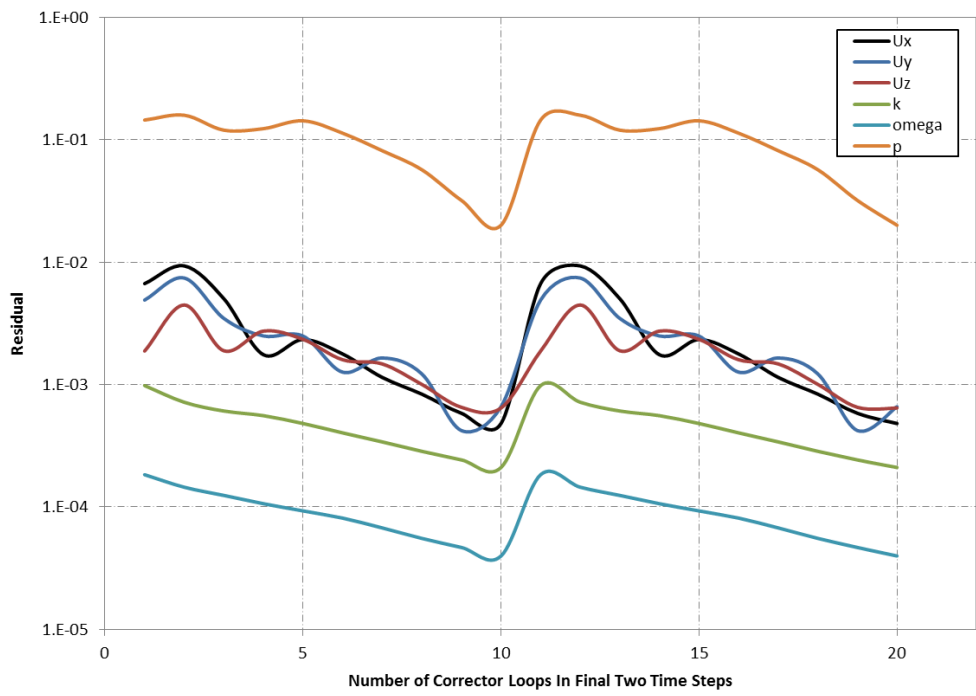
In relation to the overall pump performance, the output data for the studies using 5 and 10 corrector loops are more aligned, with the different 20 and 40 corrector loop studies being significantly different. For example, the differences between the values for the 10 and 20 corrector loop studies are 34.3% and 35.2% for generated head and hydraulic efficiency respectively. Comparing the 20 and 40 loop studies however, the differences are only 0.5% and 0.4% respectively.

This close comparison in output values was also seen when assessing the previously outlined properties at the specified locations across the impeller and volute casing, where the differences of the studies using the 5 and 10 corrector loops are highlighted and similarity of using 20 and 40 corrector loops reinforced.

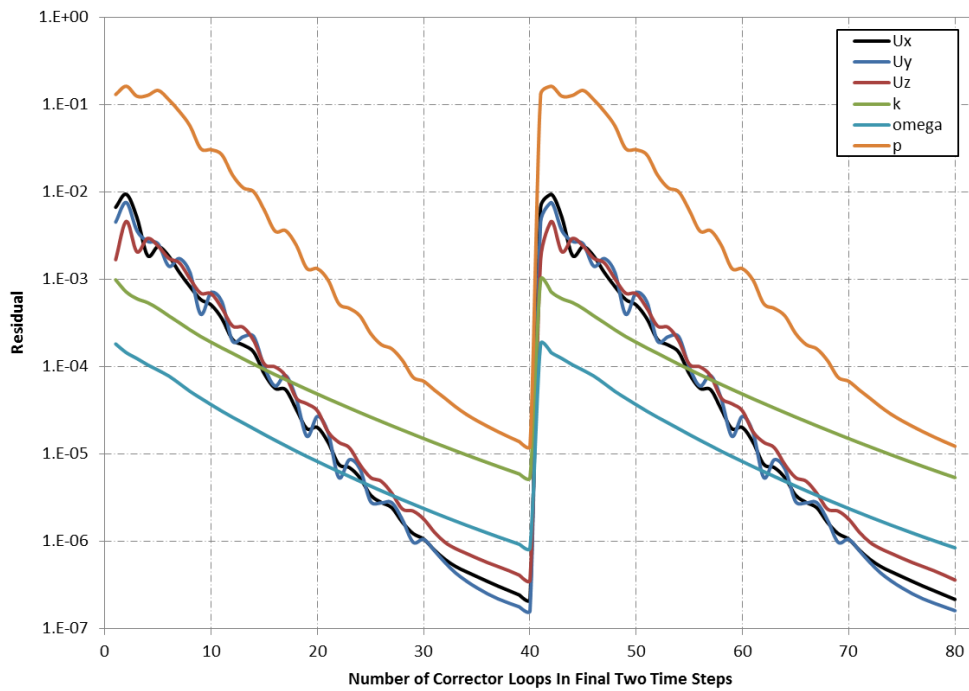
In terms of convergence, a comparison was once again made, this time between Figure 5.10 (b) and Figure 5.11. Whilst there was an improvement in the convergence of the simulation using the 40 corrector loops, the close agreement with the study using the 20 corrector loops and the significant increase in computational time (from 2.1 hours to 3.8 hours per impeller revolution) resulted in a decision to maintain the use of 20 corrector loops for the subsequent studies.



(a)



(b)



(c)

Figure 5.11 – Convergence Levels Using transientSimpleDyMFoam with Number of Corrector Loops of (a) 5 (b) 10 and (c) 40

5.2.4 Time Discretisation Scheme

With the solver, time step and number of internal corrector loops selected, the next study examined the sensitivity to time discretisation scheme by comparing the Euler, Crank-Nicholson and backward schemes discussed previously.

The raw time-varying performance data for the impeller and complete pump obtained using the turboPerformance utility over the final revolution of the pump impeller was once again reviewed, with the summarised data shown below in Table 5-5.

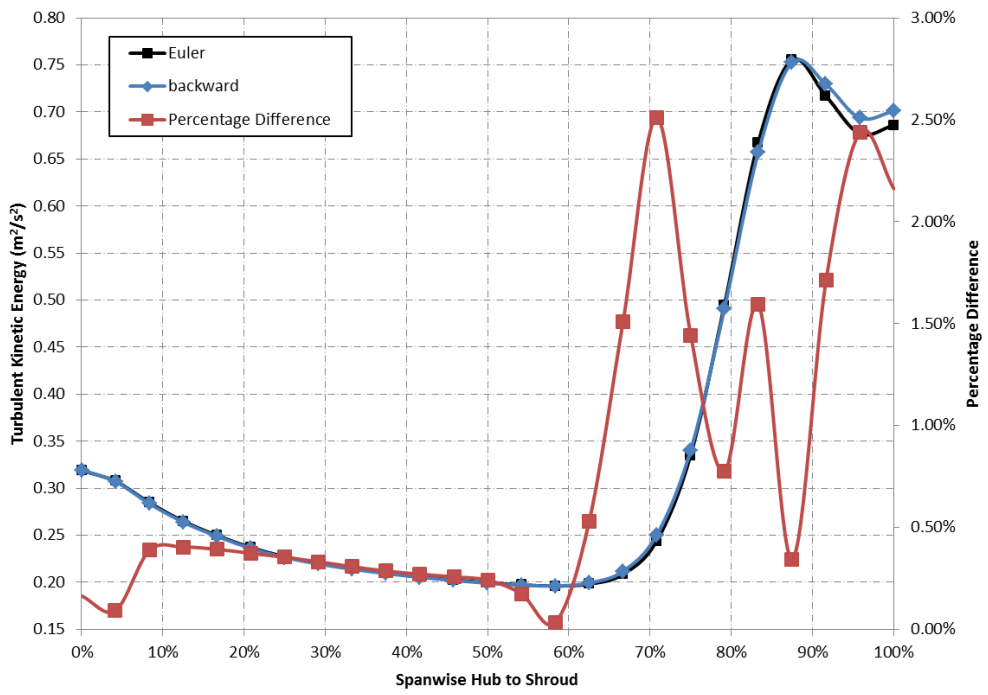
Table 5-5 - Summarised Results for Different Time Discretisation

Scheme	Impeller				Pump			Run Time Per Impeller Rev. (hrs)
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)	
Euler	61.1	140.4	176.8	79.4	56.4	129.4	73.2	2.1
backward	63.2	145.2	159.7	90.9	58.3	133.8	84.8	2.1

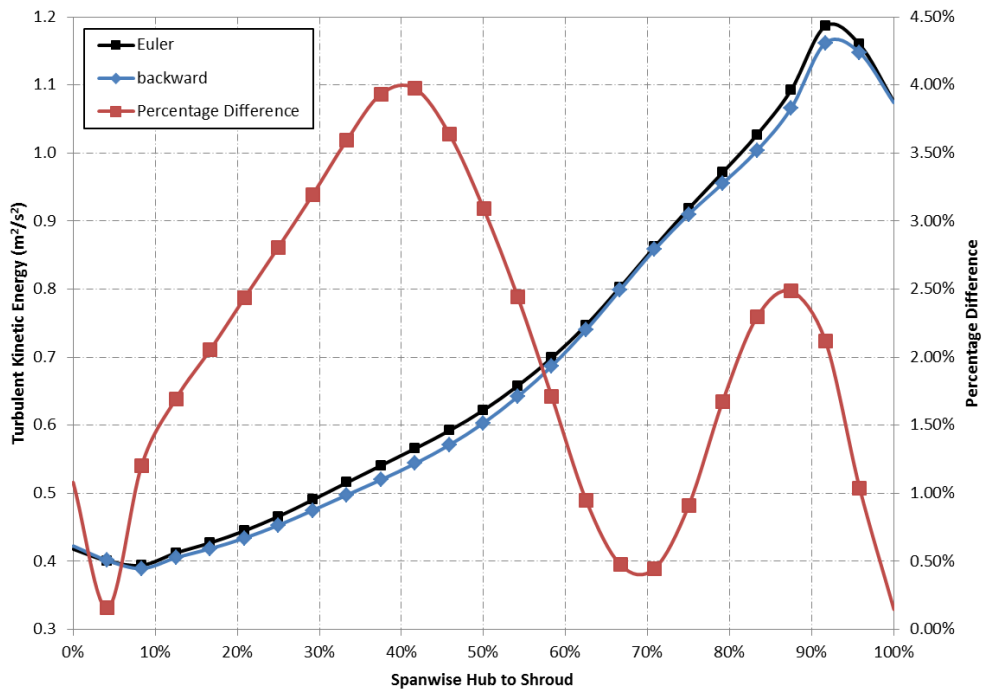
The results obtained for the Crank-Nicholson simulations are omitted as they all diverged significantly after a period of time. Despite several attempts to rectify this, divergence remained an issue throughout, confirming the discussion in Section 4.2.4.2 that highlighted the reduced stability compared to other schemes and requirement of a smaller time step to work effectively. As this was not an option in this case, further attempts were not explored, nor was the investigation into the effect of the blending function which could be a study in its own right.

By examining Table 5-5 it can be seen that a difference was highlighted between these two approaches across all performance parameters. In terms of the impeller performance, the difference was seen to be 3.4%, 10.2% and 13.5% for generated head, absorbed power and hydraulic efficiency respectively, with the study using the backward scheme predicting a lower absorbed power but higher generated head and hydraulic efficiency. Examining the data for the overall pump performance, a similar trend is seen, with differences of 3.3% and 14.7% for generated head and hydraulic efficiency.

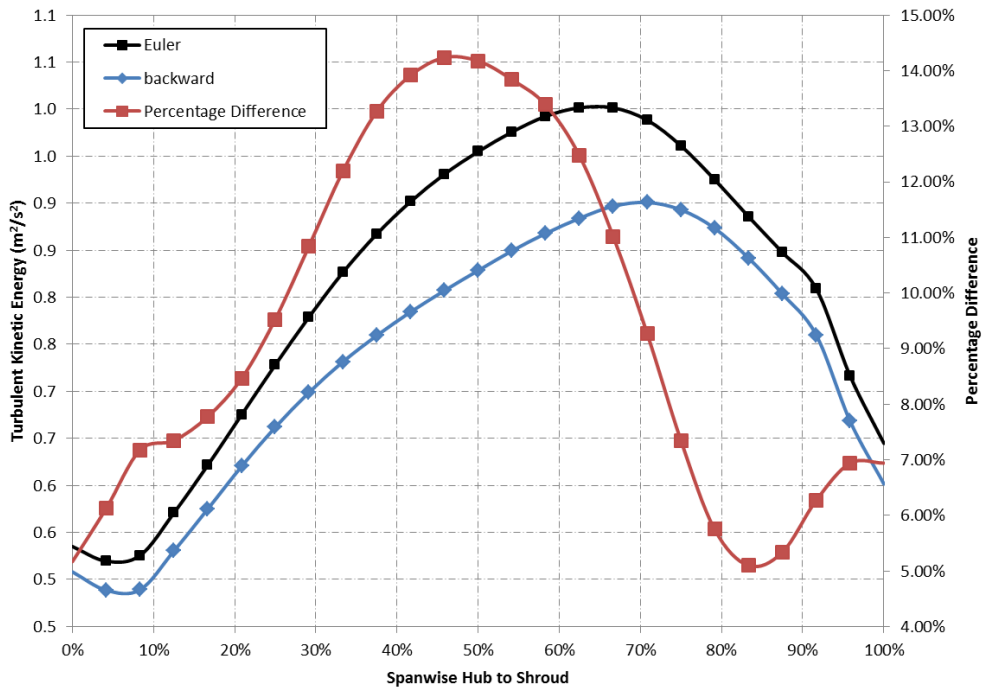
By reviewing the previously outlined properties at the specified locations across the impeller and volute casing it was seen that whilst there were differences across all of the plots, most of these were below 3%. The main differences in the impeller results were seen in Figure 5.12 in which the backward scheme resulted in higher levels of turbulent kinetic energy in the impeller, particularly at 85% stream-wise which saw differences of up to 15%.



(a)



(b)



(c)

Figure 5.12 – Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 3)

The main differences in the volute results were seen in Figure 5.13 and Figure 5.14 which relate to the velocity and turbulence at the cutwater region of the casing.

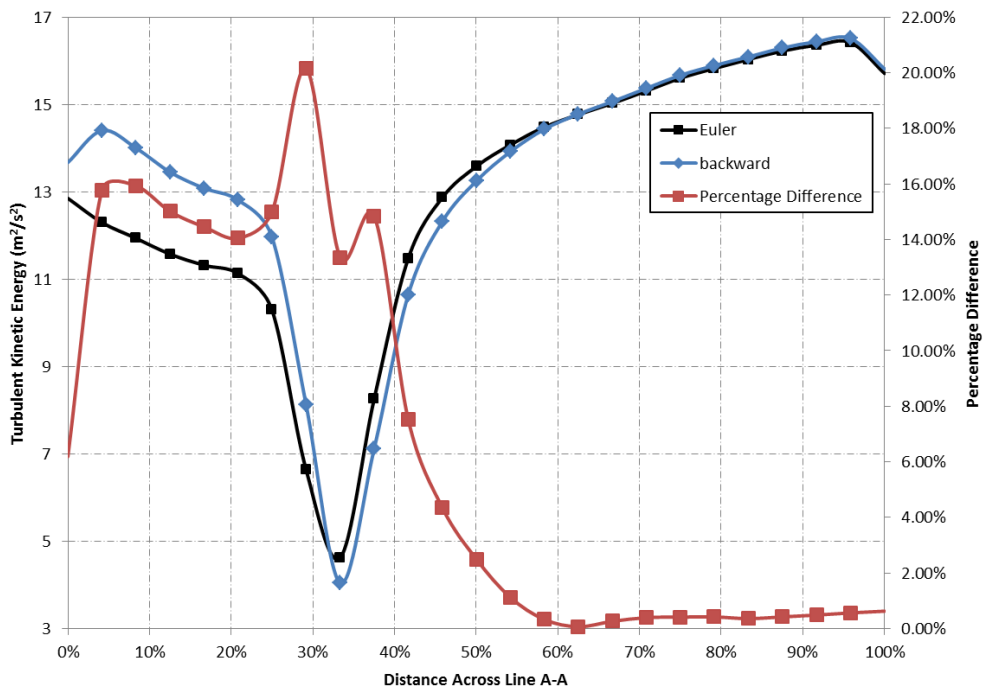


Figure 5.13 – Velocity across Line A-A (Figure 4.11) in Volute Casing (Study 3)

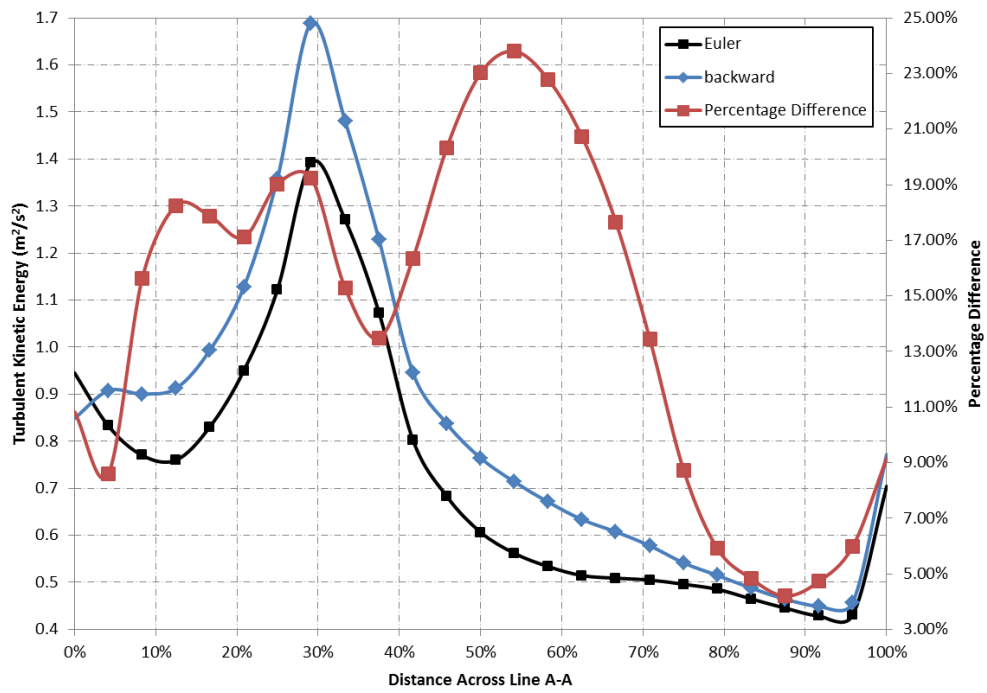


Figure 5.14 – Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Study 3)

A review of the level of convergence in Figure 5.15 shows similar levels to the upwind scheme seen in Figure 5.10 (b).

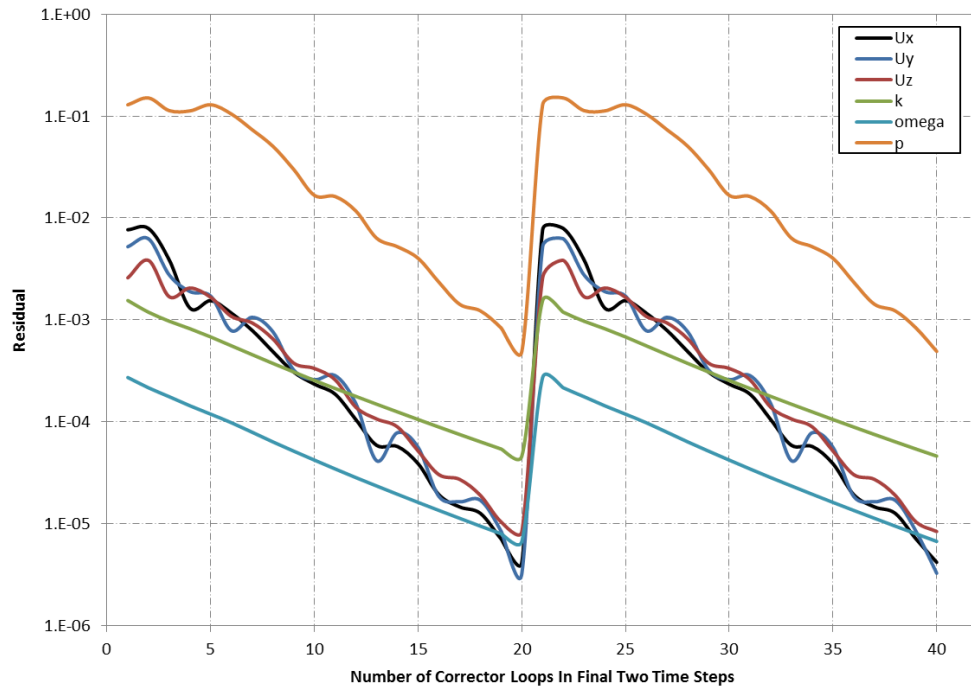


Figure 5.15 – Convergence Levels using backward Time Discretisation (Study 3)

Overall, based on the fact that there were differences of reasonable significance using the backward scheme, similar convergence and a computational time that matched the Euler scheme, the second order backward scheme was concluded to be the best option to use in subsequent investigations.

5.2.5 Velocity Convection Scheme

With the solver, time step, number of internal corrector loops and time discretisation scheme selected, the next study relates to the velocity convection scheme used. With the previous studies utilising the first-order upwind scheme, a comparison with the second-order linearUpwind scheme was made.

The raw time-varying performance data for the impeller and complete pump obtained using the turboPerformance utility over the final revolution of the pump impeller was once again reviewed, with the summarised data shown below in Table 5-6.

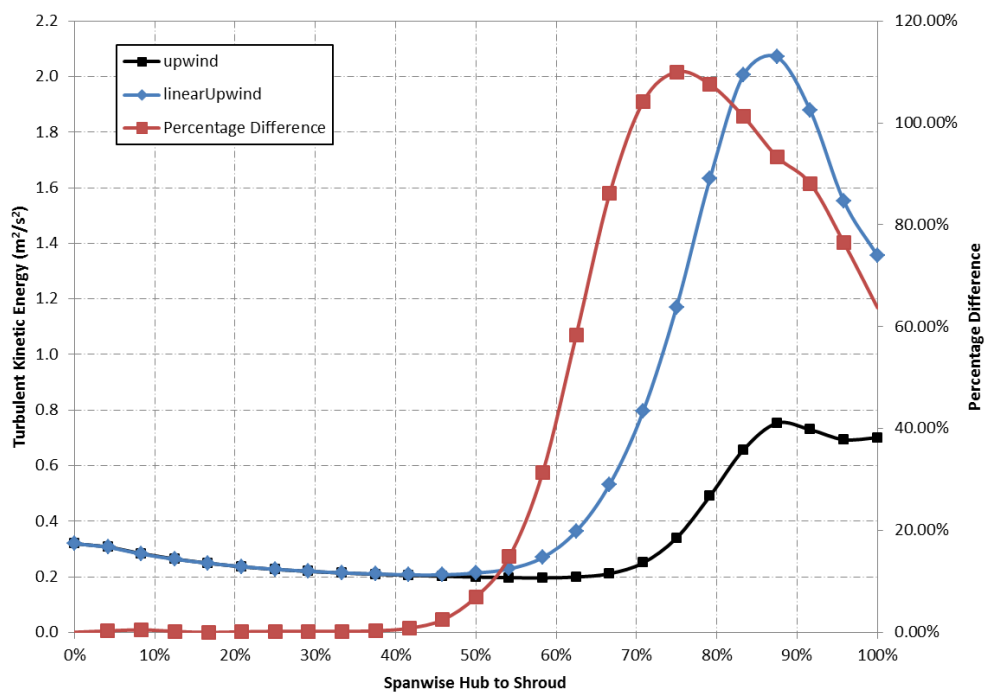
Table 5-6 - Summarised Results Different Velocity Convection Schemes

Scheme	Impeller				Pump			Run Time Per Impeller Rev. (hrs)
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)	
upwind	63.2	145.2	159.7	90.9	58.3	133.8	84.8	2.1
linear Upwind	62.7	144	157.1	91.7	58.8	134.9	85.9	2.1

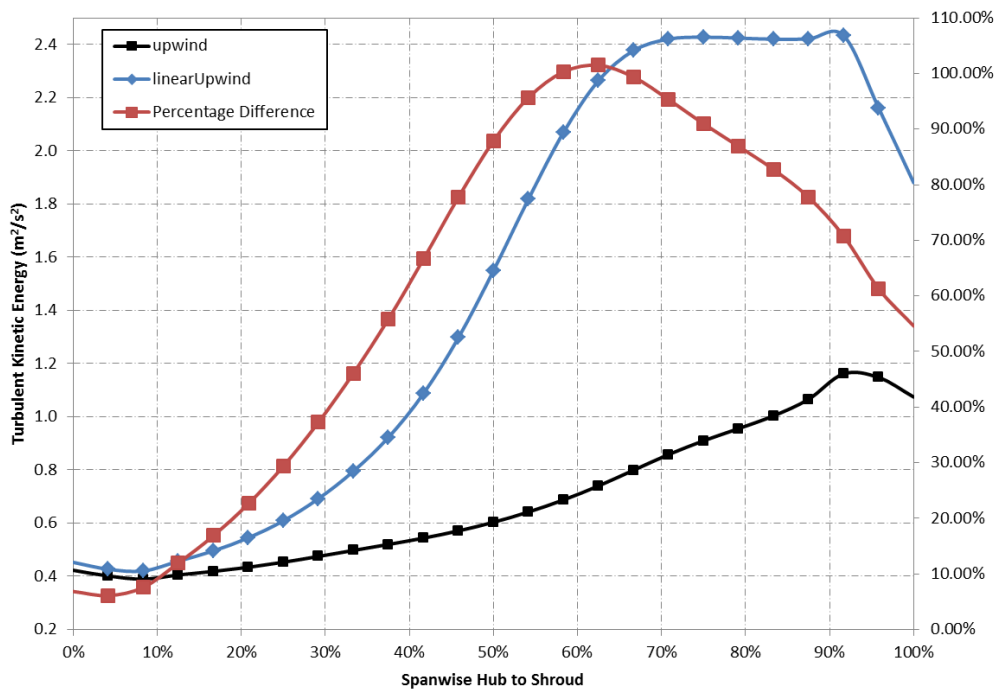
By examining Table 5-6 it can be seen that there is a small difference in the predicted performance of the impeller across the two approaches, with a 0.8%, 1.6% and 0.9% difference for generated head, absorbed power and hydraulic efficiency respectively, with the study using the backward scheme predicting a lower absorbed power but higher generated head and hydraulic efficiency. Examining the data for the overall pump

performance, a similar trend is seen, with differences of 0.9% and 1.3% for generated head and hydraulic efficiency.

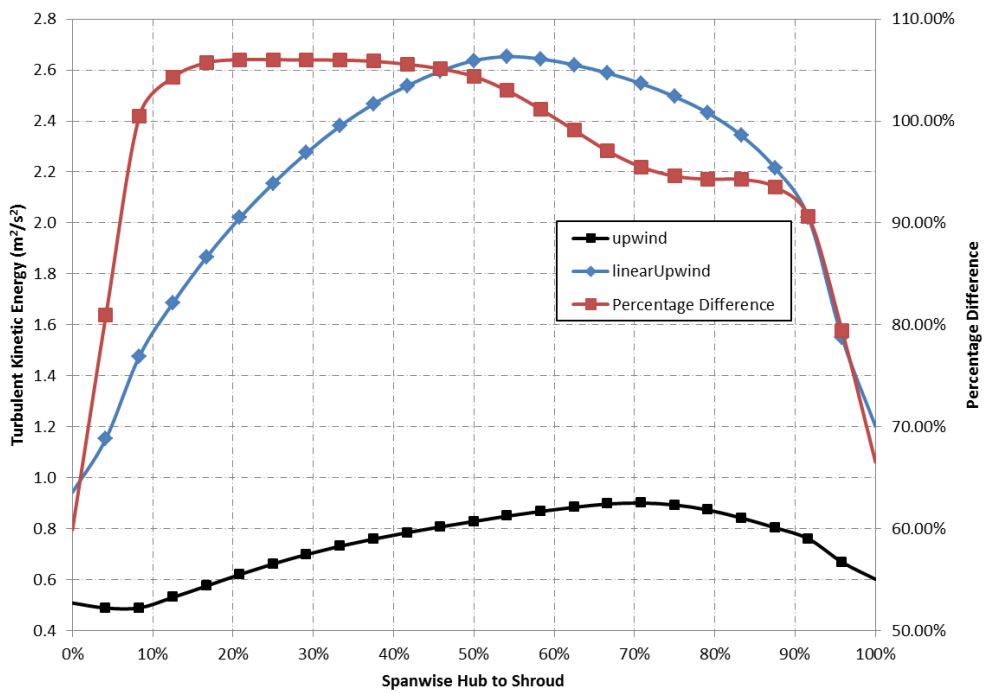
By examining the previously outlined properties at the specified locations across the impeller, the differences in pressure and velocity across each position within the impeller are typically seen to be below 2%. However, there was a significant difference in turbulent kinetic energy between the two approaches seen in Figure 5.16, with differences of up to 100% seen.



(a)



(b)



(c)

Figure 5.16 – Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 4)

Similar differences were seen at the volute section, with comparable results found for pressure in each case and more significant differences in velocity and turbulent kinetic

energy between the approaches. This is highlighted in Figure 5.17 and Figure 5.18, particularly at the cutwater position.

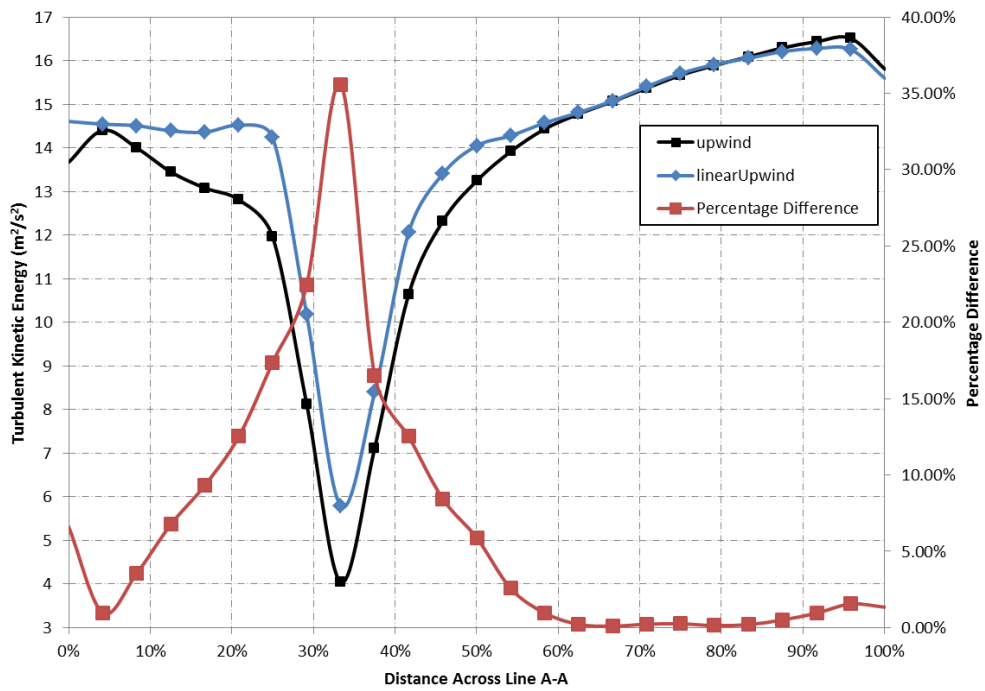


Figure 5.17 – Velocity across Line A-A (Figure 4.11) in Volute Casing (Study 4)

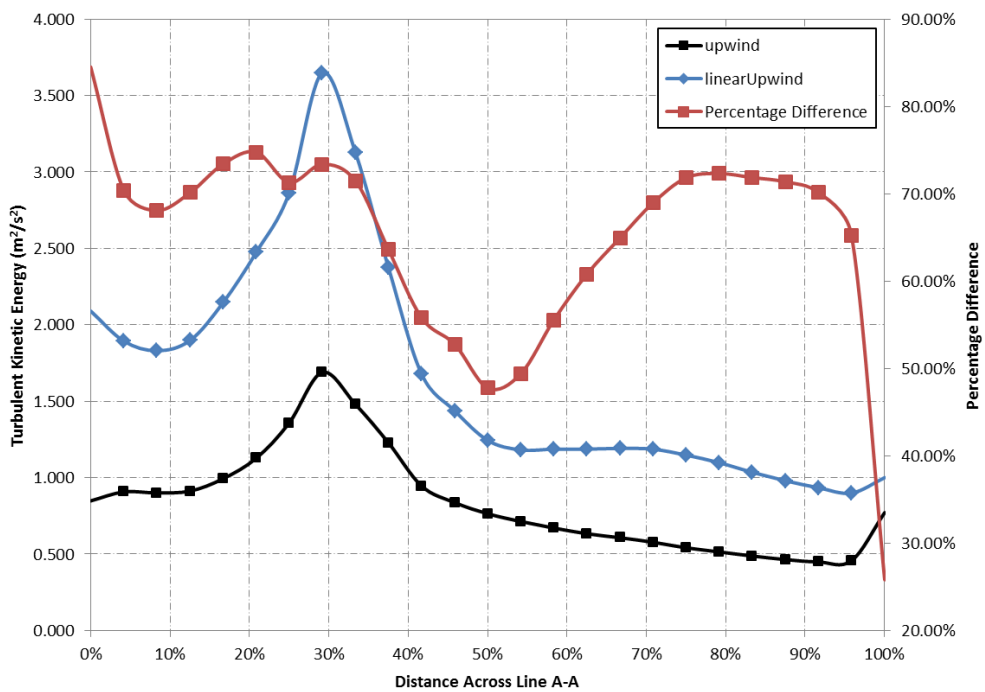


Figure 5.18 – Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Study 4)

As seen previously, there was no real difference between the upwind and linearUpwind approaches in terms of convergence when examining Figure 5.15 and Figure 5.19.

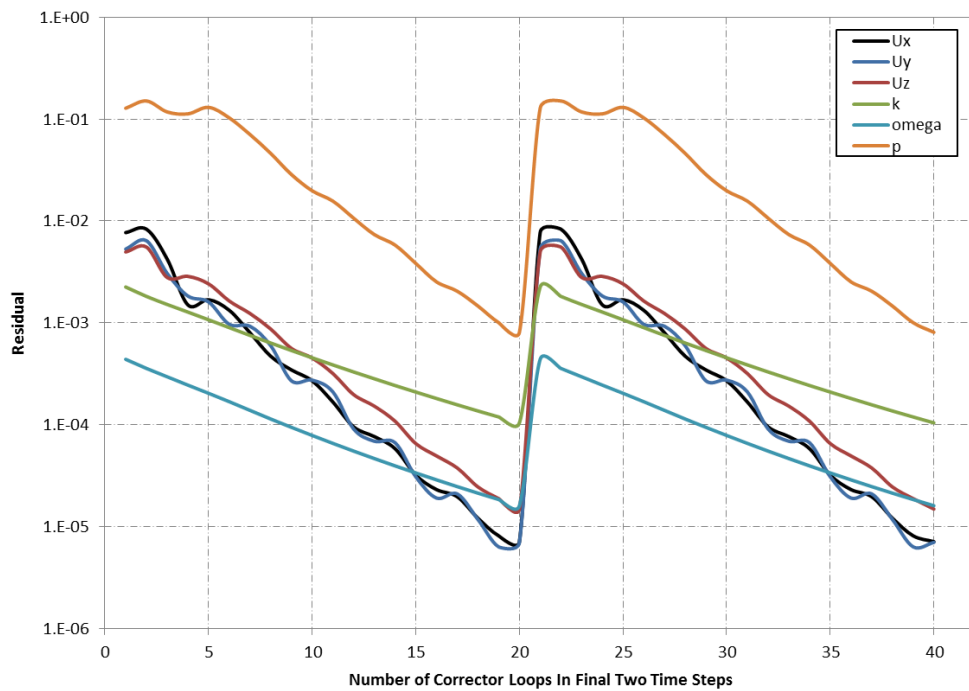


Figure 5.19 – Convergence Levels using backward Time Discretisation (Study 4)

Overall, based on the fact there were differences of reasonable significance using the linearUpwind scheme, particularly in relation to the levels of turbulence and velocity at the cutwater location whilst consistent convergence and a computational time that matched the first-order linear scheme, the linearUpwind scheme was concluded to be the best option to use in subsequent investigations.

5.2.6 Comparisons between OpenFOAM and ANSYS CFX

In order to establish the usefulness of the results obtained through the final OpenFOAM studies in the previous section, the results were compared directly against those obtained using ANSYS CFX. This was done by examining an additional two studies in OpenFOAM at 70% and 130% Q_{opt} as well and running simulations at 70%, 100% and 130% Q_{opt} using ANSYS CFX.

5.2.6.1 CFX Setup Conditions

The setup used in these simulations was similar to that seen in the OpenFOAM studies, with velocity inlet and static pressure outlet boundaries conditions specified as in the OpenFOAM cases. The second order backward differencing time scheme was used with a 'high resolution' second order convection scheme. The maximum number of internal correction loops was set at 15, less than that used in the OpenFOAM studies due to improved convergence seen during initial simulation work. A time step of 5.45×10^{-4} s (100 per revolution) was prescribed and the k-omega SST turbulence model selected.

5.2.6.2 Comparison of Results

The summarised performance results obtained for the three prescribed flow rates are shown in Table 5-7 and Table 5-8 respectively with Table 5-9 highlighting the percentage differences of each of the performance measures.

Table 5-7 - Summarised Results for OpenFOAM across Flow Range

%Q _{opt}	Impeller				Pump		
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)
70	68.2	109.5	122.2	89.6	63.0	101.2	82.8
100	62.7	144	157.1	91.7	58.8	134.9	85.9
130	58.1	173.4	189.4	91.6	54.1	161.4	85.2

Table 5-8 - Summarised Results for ANSYS CFX across Flow Range

%Q _{opt}	Impeller				Pump		
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)
70	68.0	109.3	119.1	91.7	63.3	101.7	85.4
100	63.2	145.0	154.6	93.8	59.8	137.1	88.7
130	58.5	174.5	187.7	93.0	55.1	164.3	87.5

Table 5-9 – Summarised Differences Between OpenFOAM and ANSYS CFX Results

% Q_{opt}	Impeller				Pump		
	Total Head (m)	Hydraulic Power (kW)	Absorbed Power (kW)	Hydraulic Efficiency (%)	Total Head (m)	Hydraulic Power (kW)	Hydraulic Efficiency (%)
70	0.3%	0.2%	2.6%	2.3%	0.5%	0.5%	3.1%
100	0.8%	0.7%	1.6%	2.3%	1.7%	1.6%	3.2%
130	0.7%	0.6%	0.9%	1.5%	1.8%	1.8%	2.7%

The data related to the generated head is also shown in Figure 5.20 in order to provide a comparison between the simulation work and the experimental data.

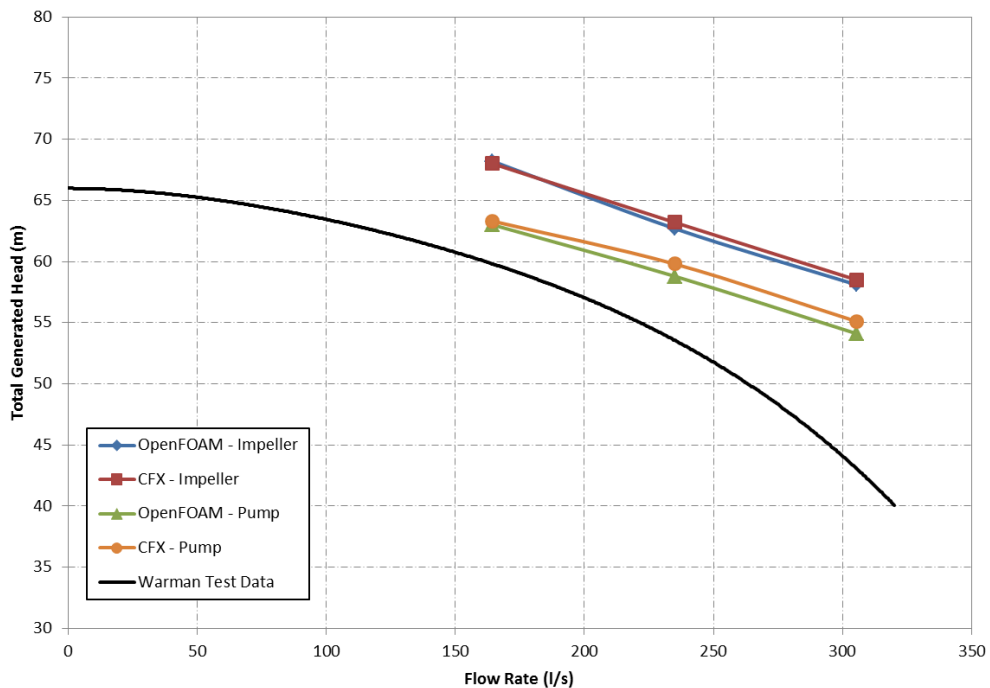


Figure 5.20 – Generated Head Comparison between OpenFOAM and ANSYS CFX

Reviewing the summarised results in in Table 5-7 and Table 5-8 in conjunction with the percentage differences in Table 5-9, it can be seen in terms of performance prediction, the OpenFOAM solver transientSimpleDyMFoam has close agreement with the ANSYS CFX commercial solver.

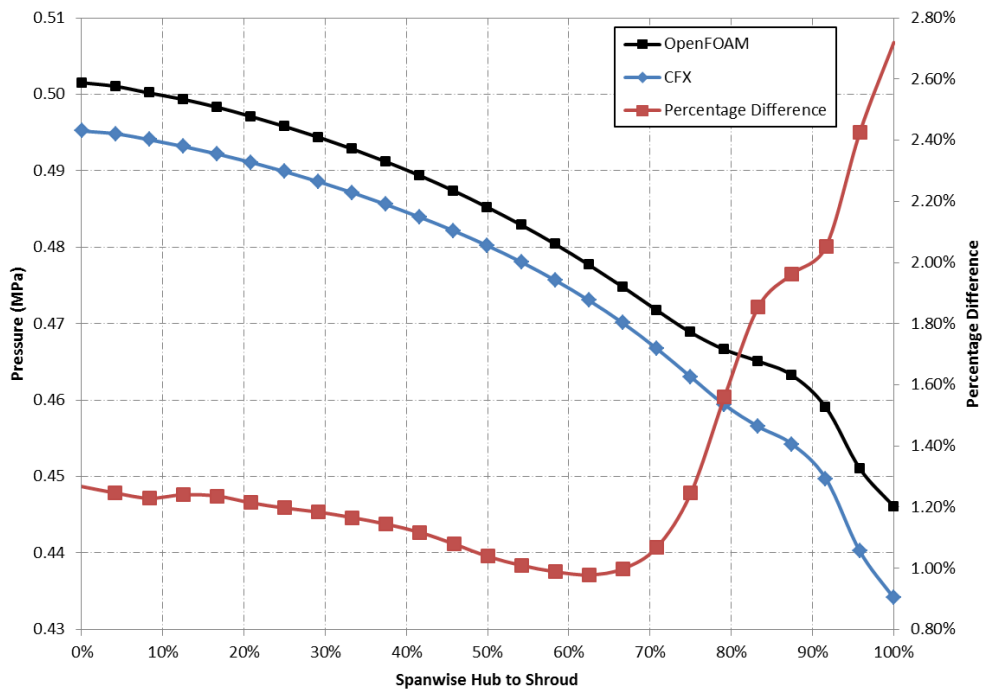
In each instance, the ANSYS CFX solver predicts a higher generated head for both the impeller-only and complete pump output apart from the impeller head at 70% Q_{opt} which

sees OpenFOAM predict a higher head by 0.3%. Again, in each instance, ANSYS CFX predicts a lower power absorbed by the impeller. The combination of each of these two facts results in predicted hydraulic efficiencies for both the impeller and overall pump that is higher using the ANSYS CFX solver.

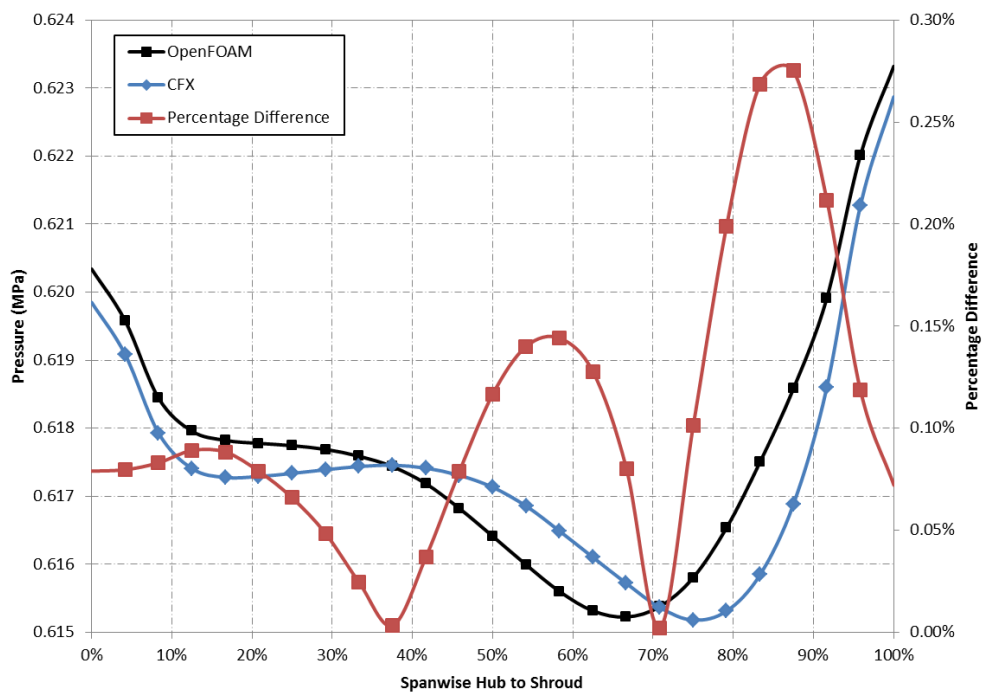
Looking at the impeller performance figures in greater detail, the percentage differences between the solvers across the flow range in terms of predicted generated head and power absorbed by the impeller are seen to be within 1% and 2.6% respectively, leading to a maximum percentage difference of hydraulic efficiency of 2.3%. In relation to the overall pump performance, the percentage differences in predicted head and hydraulic efficiency are seen to be within a 2% and 3.2% respectively.

As with the OpenFOAM investigations, line plots of pressure, velocity and turbulent kinetic energy were reviewed for both the impeller and volute casing, with the results from each solver being compared against each other at 70%, 100% and 130% Q_{opt} .

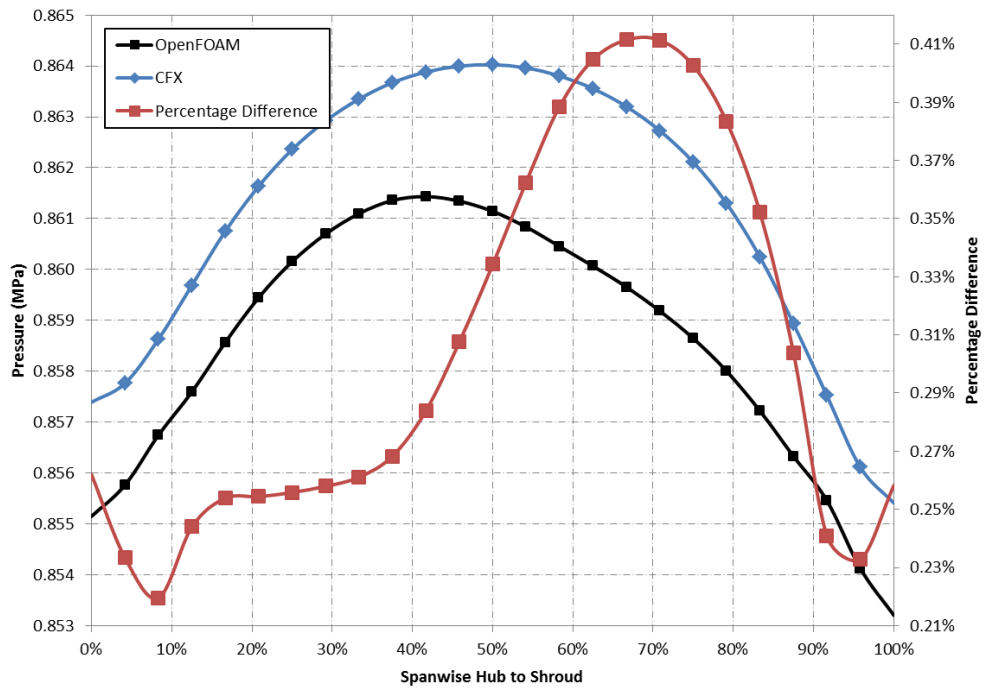
Reviewing the impeller results in general, the percentage difference between the two solvers for the static pressure across the flow range is less than 3%, such as in Figure 5.21, with the total pressure differences generally being less than 5%, for example as shown in Figure 5.22. In terms of the velocity in the stationary frame, the extent of the difference between the solvers does vary across the flow range.



(a)

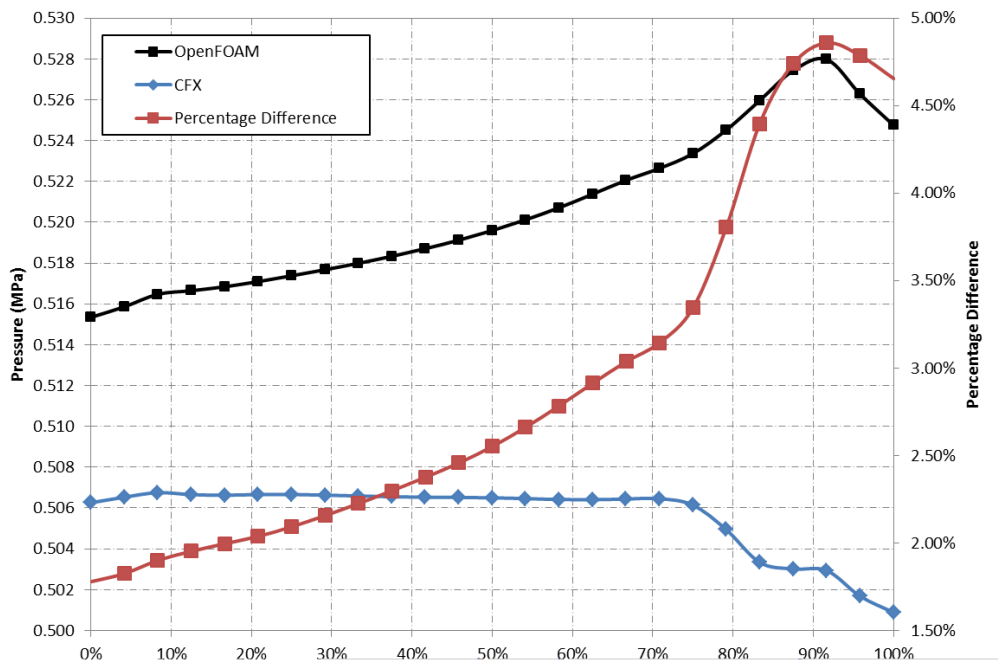


(b)

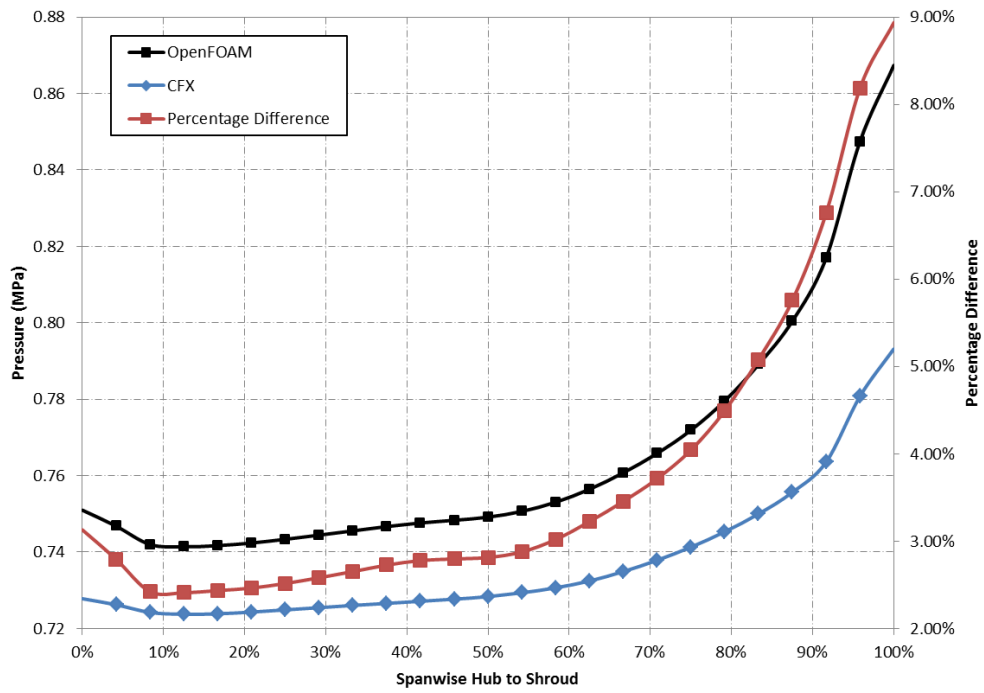


(c)

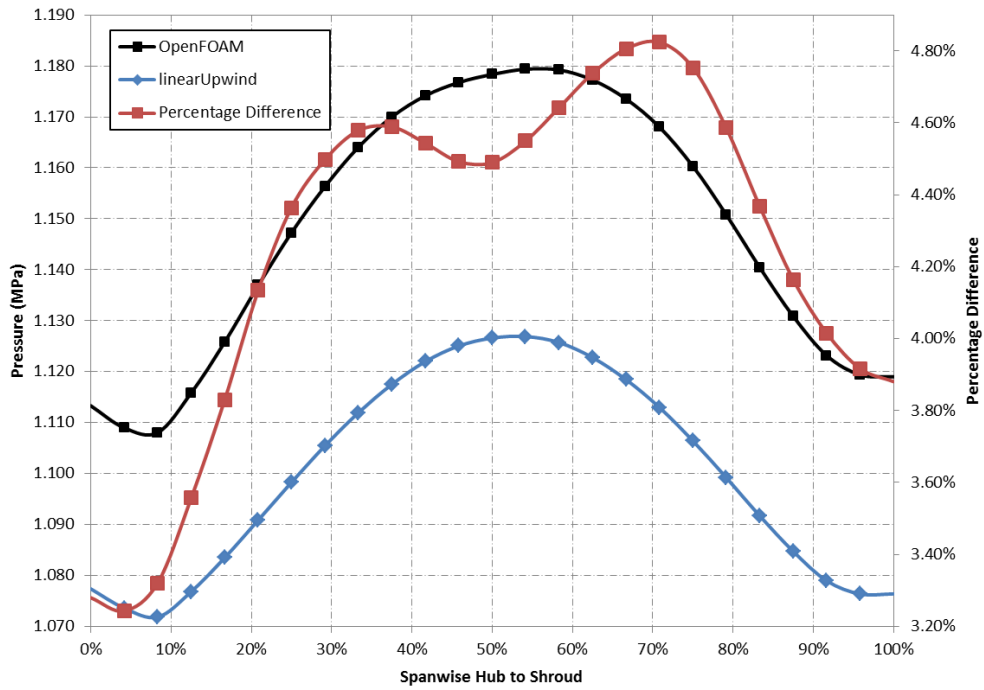
Figure 5.21 – Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 100% Q_{opt})



(a)



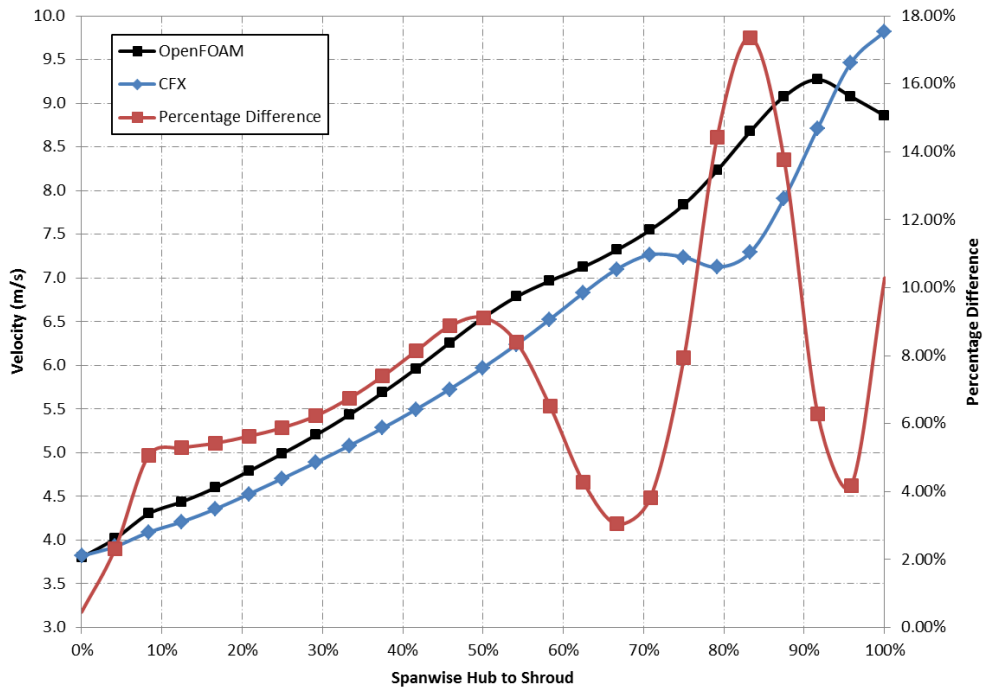
(b)



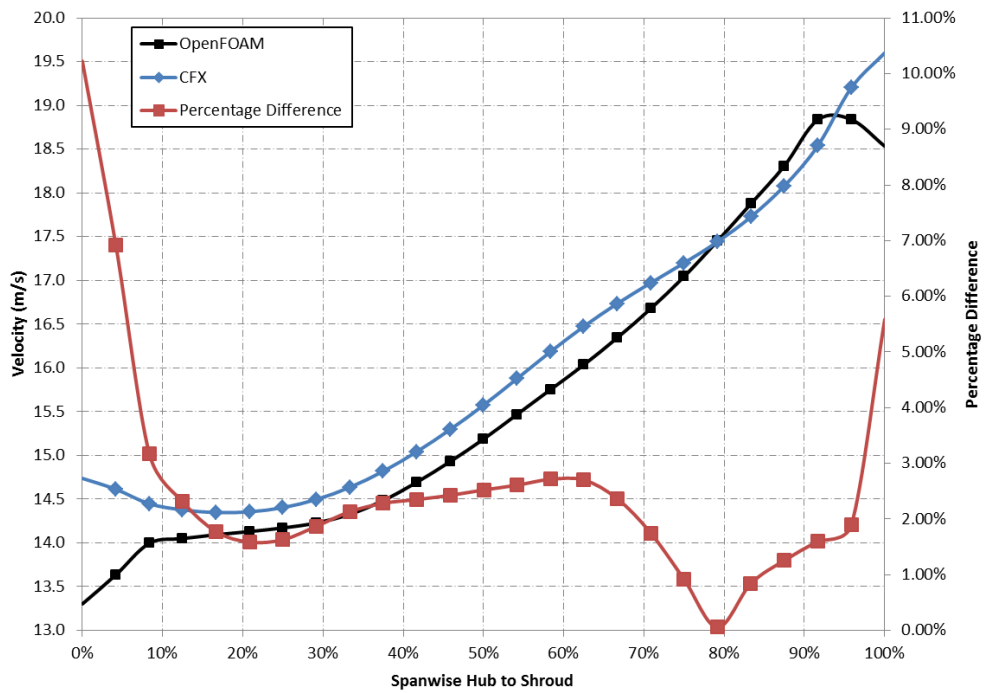
(c)

Figure 5.22 – Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Q_{opt}) (Software Comparison – 100% Q_{opt})

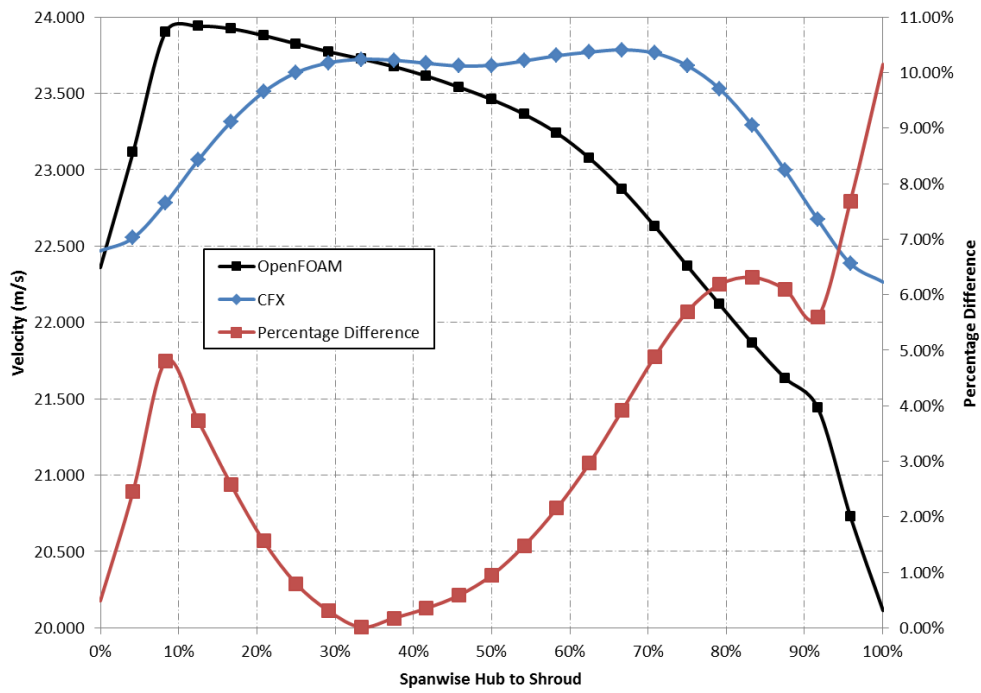
At 100% and 130% Q_{opt} , the velocity difference is generally less than 4% although it does increase at the walls, where the blades attach to the shrouds. At the off-design flow of 70% Q_{opt} , the difference is greater but still generally within 10% as highlighted in Figure 5.23.



(a)



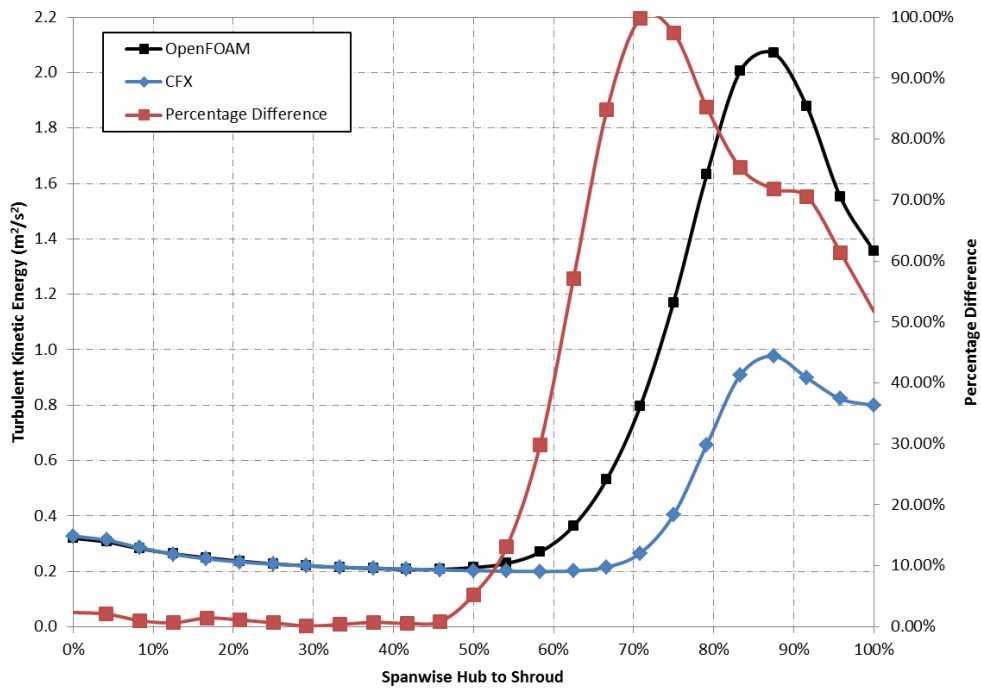
(b)



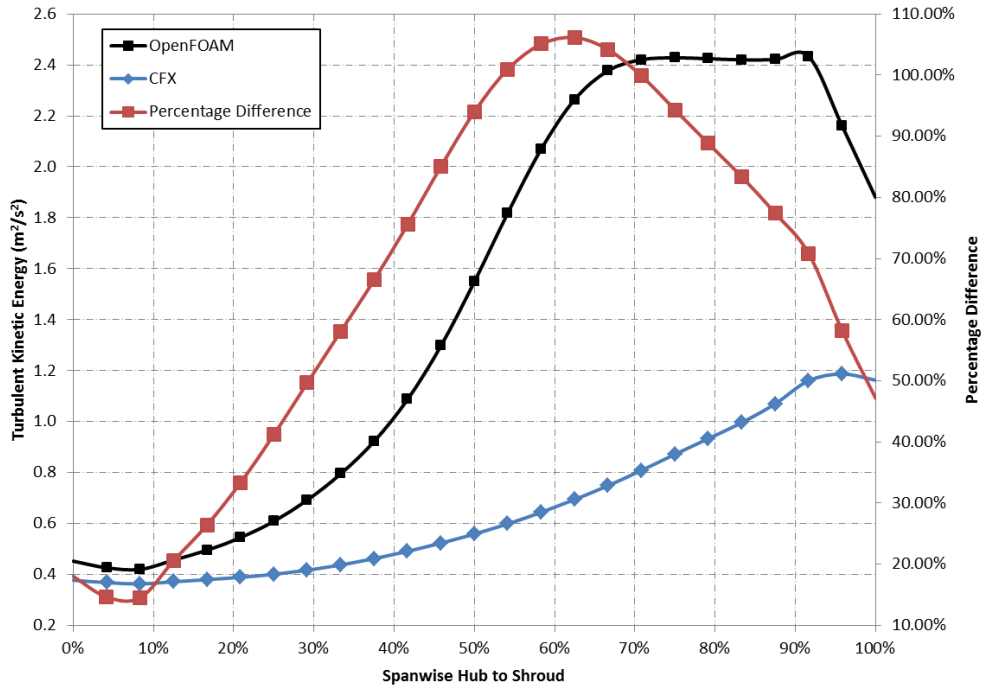
(c)

Figure 5.23 – Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 70% Q_{opt})

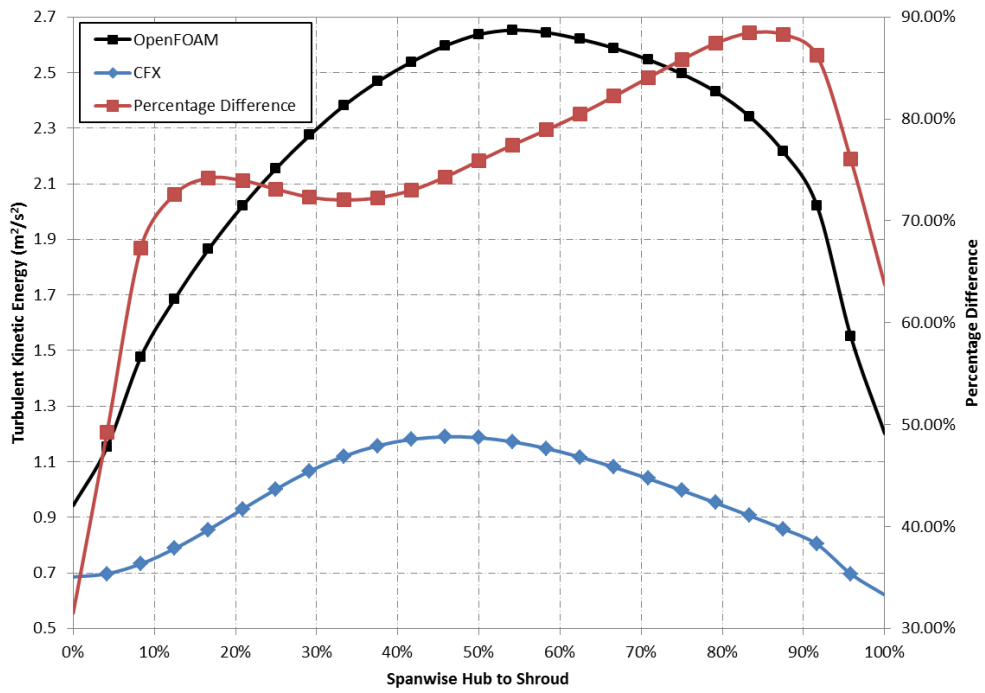
The greatest difference is seen in the turbulence kinetic energy, with up to 100% percentage difference seen between the two solvers across each of the flow rates, with the greatest differences seen at the 15% stream-wise position as highlighted in Figure 5.24.



(a)



(b)



(c)

Figure 5.24 – Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 100% Q_{opt})

Reviewing the volute casing line plots, a similar trend is seen. The percentage difference for static and total pressures is seen to be within 2.5% and 5% in general across each of the flow ranges. The difference in velocity is generally less than 10% across each of the flow rates apart from at the specific location of the cutwater geometry, where this increases to 25% at the off-design flows of 70% and 130% Q_{opt} and to 50% at 100% Q_{opt} as seen in Figure 5.25 to Figure 5.27.

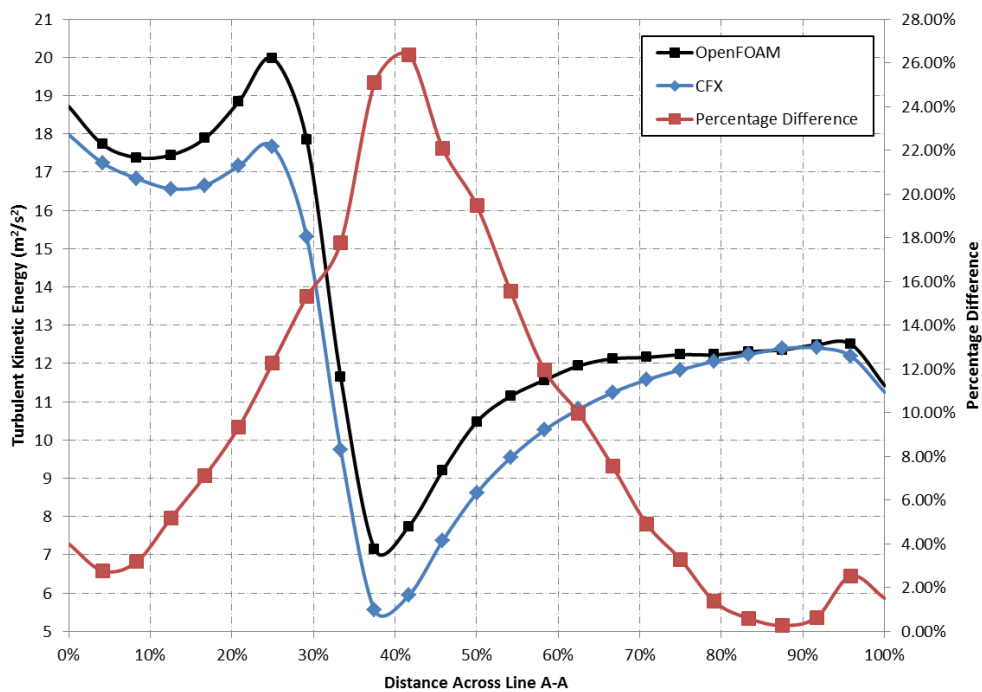


Figure 5.25 – Velocity across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 70% Q_{opt})

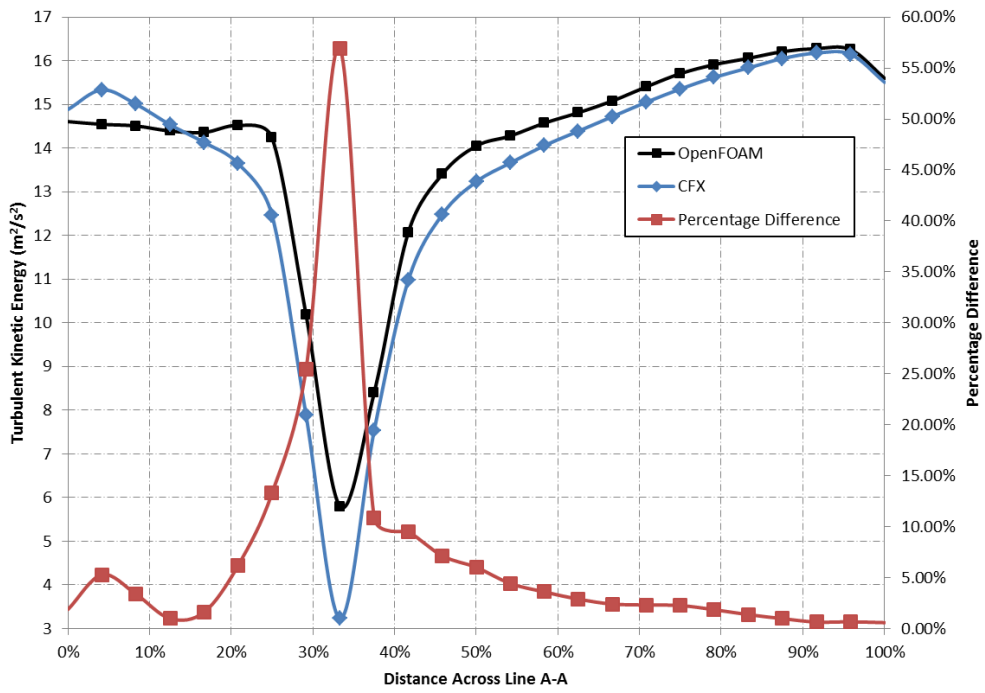


Figure 5.26 – Velocity across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 100% Q_{opt})

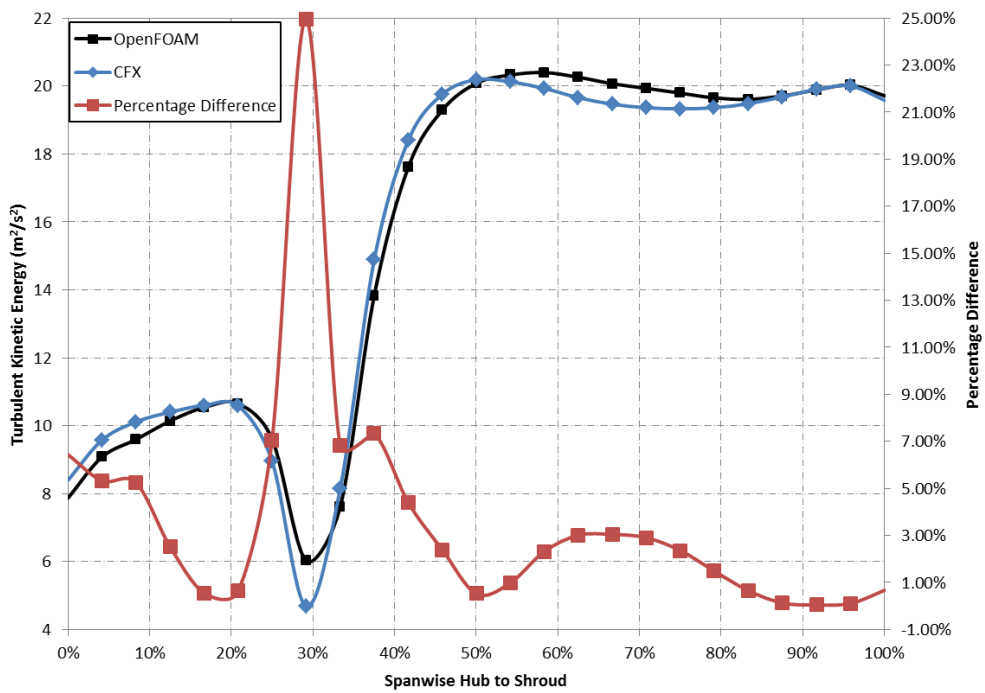


Figure 5.27 – Velocity across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 130% Q_{opt})

As with the impeller review, the turbulent kinetic energy sees a larger difference between the solvers with an average in the region of 30% and a high of 50% as demonstrated in Figure 5.28.

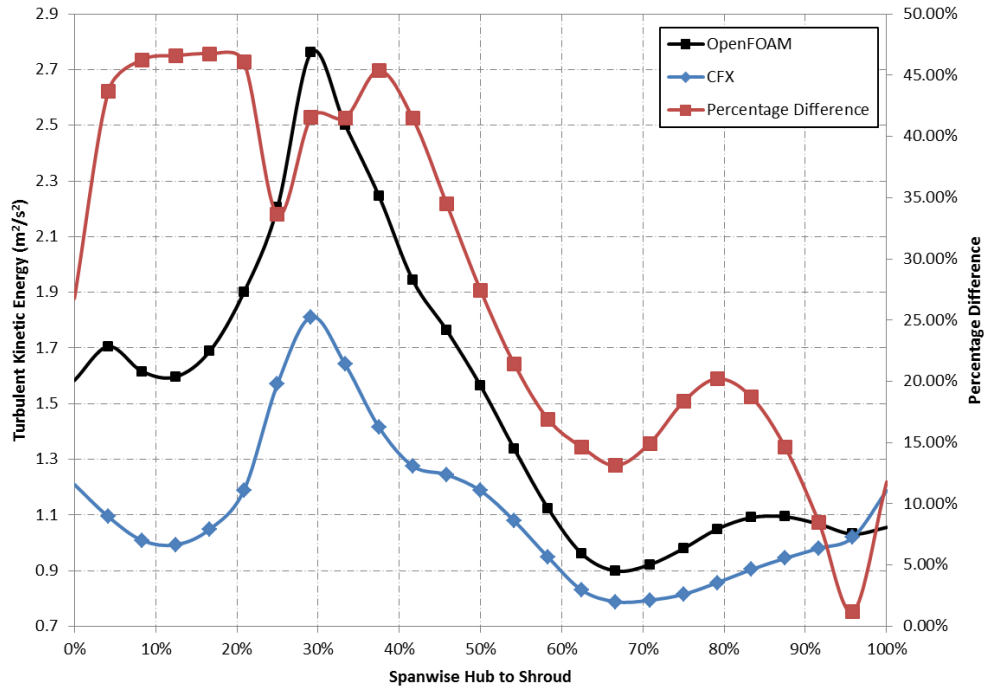


Figure 5.28 – Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 130% Q_{opt})

Whilst this difference is larger, it is not as pronounced as that seen in relation to the impeller data.

Overall, the results of the OpenFOAM transientSimpleDyMFoam solver with this particular setup show good agreement with those obtained using the ANSYS CFX solver.

The most notable difference that was highlighted during the review was the over-prediction of turbulent kinetic energy by the OpenFOAM solver. For clarity, Figure 5.29 shows a slice taken at the centreline of the impeller for the 100% Q_{opt} results. It can be seen that the OpenFOAM solver predicted more turbulent kinetic energy than the ANSYS CFX solver, most evident at the outlet regions of the impeller blade passages and particularly at the passage closest to the cutwater.

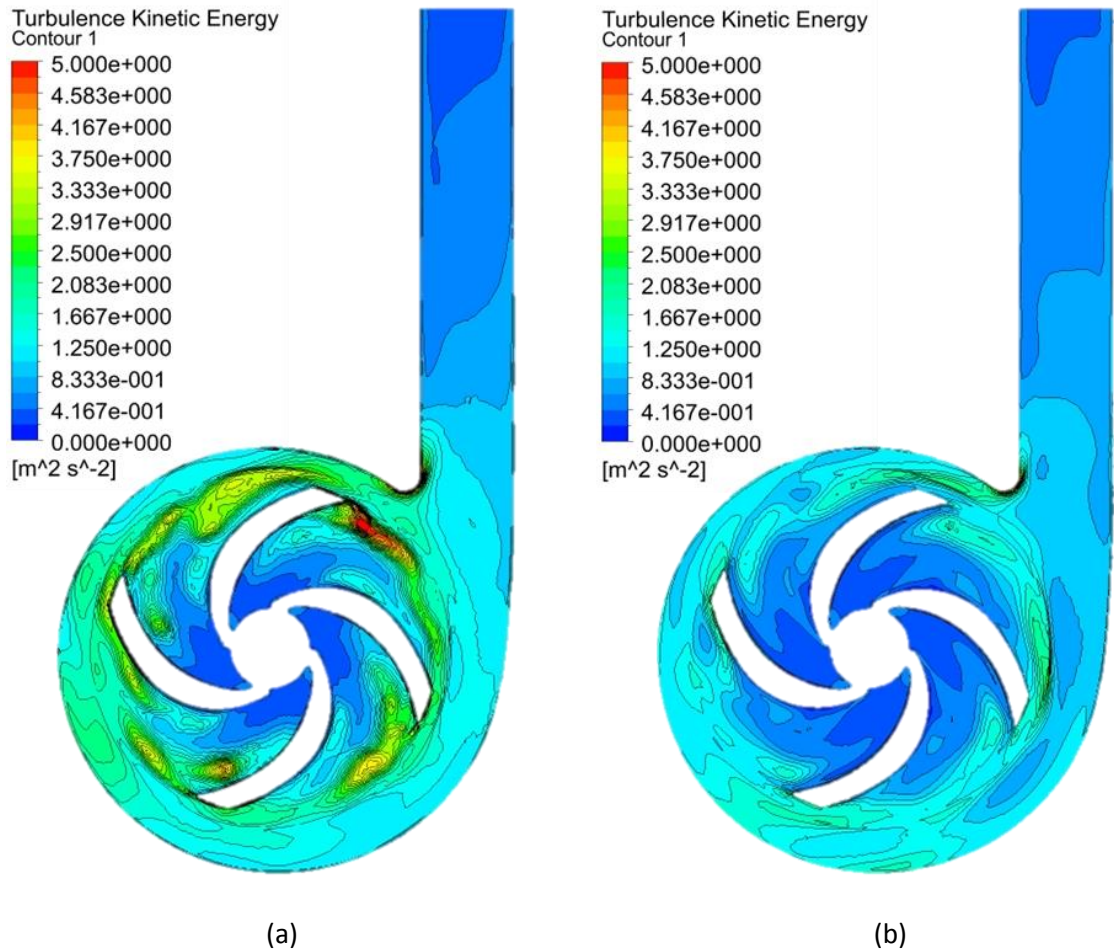


Figure 5.29 – Turbulent Kinetic Energy at 100% Q_{opt} for (a) OpenFOAM and (b) ANSYS CFX

This trend is also seen in Figure 5.30 which highlights the specific dissipation rate (turbulent eddy frequency), where larger rates are predicted by the OpenFOAM solver towards the outlet of the impeller passages and particularly at the outlet tips on the suction/low pressure side of the blades.

Considering that this is the most significant difference seen between the two solvers, further work could be undertaken in order to assess the impact of the turbulence model and associated variables on the results. For example, in relation to the turbulence intensity specified at the inlet boundary condition, whilst there is literature to suggest that this may not be significant, it is also recommended that a sensitivity study to this parameter would be useful [67].

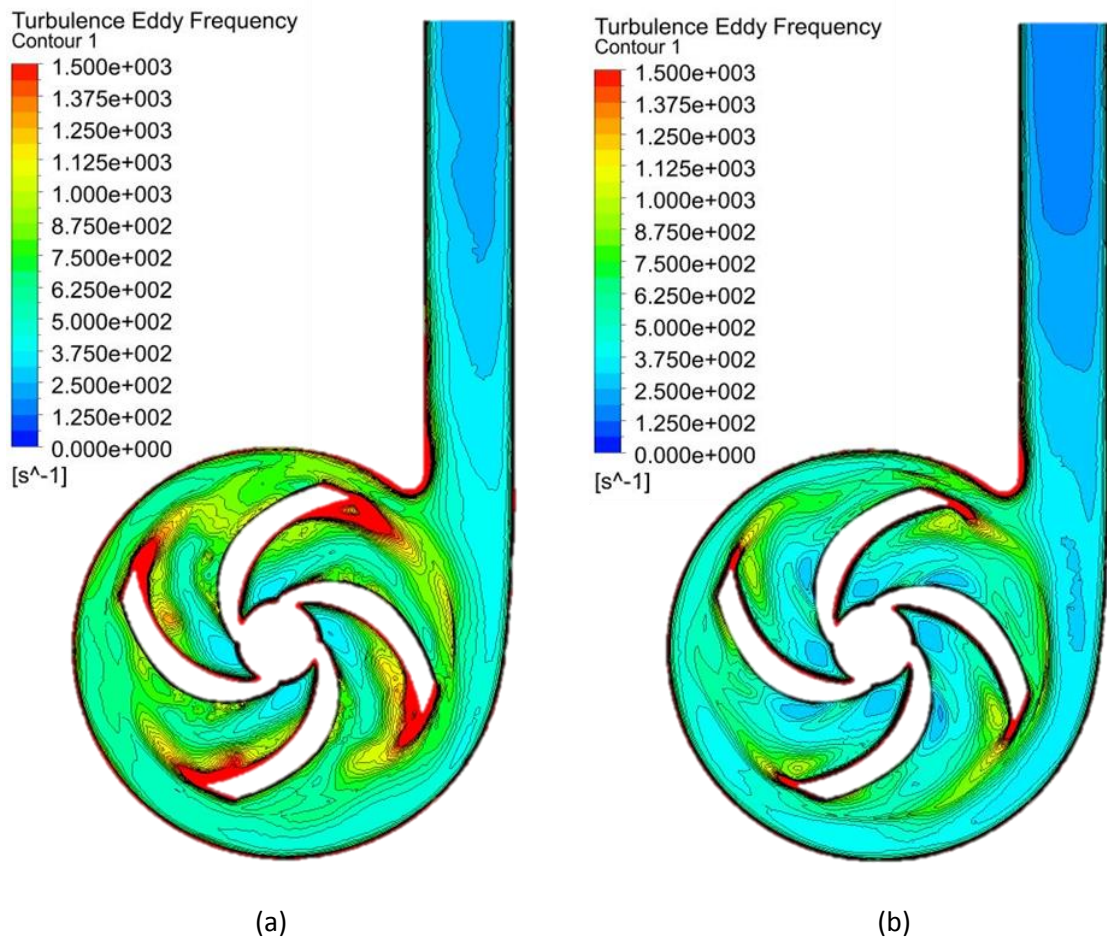


Figure 5.30 – Turbulent Eddy Frequency at 100% Q_{opt} for (a) OpenFOAM and (b) ANSYS CFX

In order to obtain a more detailed insight into the differences in accuracy between the solvers, a more complex model including the front and rear plates and, dependent of computational power, leakage paths, could be included.

5.3 Conclusions

The initial work conducted in this chapter showed how it was possible to assess the steady state performance of a centrifugal pump using a simplified 3D mesh in conjunction with the MRFSimpleFoam solver.

Subsequent transient analyses were conducted using the in-built pimpleDyMFoam solver and turbomachinery-specific downloadable solver transientSimpleDyMFoam. The latter of

these solvers was determined to be the best suited for such simulations and was used in all subsequent analyses. Several investigations were performed to assess the sensitivity of the solver to time step size, time discretisation scheme, number of internal corrector iteration loops and velocity convection scheme. The most suitable set up was deemed to use a time step of 5.45×10^{-4} s, backward time discretisation, 20 internal corrector loops and the second-order linearUpwind velocity convection scheme.

Finally, the results of the case study obtained using OpenFOAM were validated against outputs from the commercial CFD software ANSYS CFX and in general were found to compare favourably in terms of both performance figures and flow regime (pressure, velocity and turbulence) in the impeller and volute casing.

Chapter 6

6. Two-Phase Modelling

This chapter discusses the use of OpenFOAM and ANSYS CFX to perform two-phase flow simulations, specifically in relation to phase change in the form of cavitation.

An initial study is summarised in Section 6.1 in which the fundamental ability of both OpenFOAM's `interPhaseChangeFoam` solver and ANSYS CFX to model the formation of vapour is assessed through the study of a NACA66-212 profile.

Section 6.2 then outlines the tutorial case included in the newer version of OpenFOAM (2.3.x) in which cavitation is modelled on a rotating propeller. This simulation used the `interPhaseChangeDyMFoam` solver which allows for phase change to be modelled but also has the ability to include dynamic meshes in the analysis.

Finally, Section 6.3 outlines the approach used and the results of the assessment of cavitating flow in the Warman 8/6 AH pump using ANSYS CFX and discusses the issues encountered when attempting to utilise the equivalent OpenFOAM solver, `interPhaseChangeDyMFoam` for the same study.

6.1 NACA66 Hydrofoil

The first step in understanding the ability of OpenFOAM to model cavitating flow was to perform a simple test case in the form of a NACA profile hydrofoil. The results of this study could then be compared to those obtained through the industry-standard software ANSYS CFX in order to assess suitability for more complex work.

As significant research has already been undertaken in this area [68][69], the extent of the review in this body of work was simply to validate that the software being used was capable of predicting the vapour formation rather than performing detailed comparisons between codes, cavitation models and calibrating coefficients based on experimental data.

For the work undertaken, a NACA-212 profile was modelled using SolidWorks 3D CAD software, generated using a maximum thickness to chord ratio of 12%, camber distribution, $a = 0.8$, angle of attack, $\alpha_a = 6^\circ$ and scaled such that the chord length, $c = 0.150$ m and span, $s = 0.191$ m [69]. The domain was sized such that lengths of $5c$ and $10c$ were defined upstream and downstream of the hydrofoil respectively. A representation of the computational domain and case set up can be seen in Figure 6.1. An unstructured mesh with inflation layers at the surfaces of the NACA66-212 profile was created using the ANSYS Meshing tool.

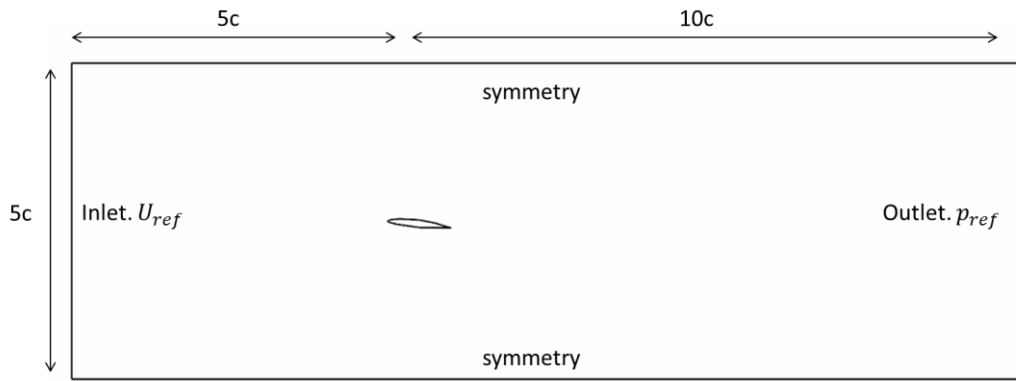


Figure 6.1 – NACA66-212 Case Set Up

For this study, the inlet velocity, U_{ref} , was fixed at 5 m/s and the outlet pressure, p_{ref} , was altered in order to vary the cavitation number, σ , as defined in (2.2). The vapour pressure in this instance was defined as $p_v = 3169$ Pa. In terms of the fluid properties, the liquid phase had a density $\rho_l = 997$ kg/m³ and kinematic viscosity $\nu_l = 8.92 \times 10^{-7}$ m²/s and the vapour phase had a density $\rho_v = 0.023$ kg/m³ and kinematic viscosity $\nu_v = 4.27 \times 10^{-4}$ m²/s, yielding a Reynolds Number, $Re = 750,000$. A turbulence intensity of 2% was used in order to calculate the inlet conditions for k and ε based on (4.21) and (6.1) respectively.

$$\varepsilon = 0.09 \frac{k^{\frac{3}{2}}}{L} \quad (6.1)$$

where L is a characteristic length, with the chord length, c , being used in this instance.

This yielded values of 0.015 m²/s² and 0.002 m²/s³ for k and ε respectively. A time step of 1×10^{-4} s was selected.

To begin the simulations a cavitation number $\sigma = 10$ was chosen in order to remove any potential issues at the initial time step with vapour formation. The outlet pressure p_{ref} was then lowered in stages in order to reduce the cavitation number to the extent that vapour formation was present.

This setup configuration and approach was the same when using both interPhaseChangeFoam and ANSYS CFX. The selected cavitation model using OpenFOAM in this particular instance was the Kunz model. Constants C_{dest} and C_{prod} from (2.43) and (2.44) were maintained at their default setting of 1000 and the mean flow time scale was set as $t_{\infty}=0.03$ s based on a length scale, L , of 0.15 m (the chord length, c). The ANSYS CFX Rayleigh-Plesset model was also left unmodified, instead using the default values for the relevant parameters in relation to the bubble dynamics [70].

The results from each solver at three different cavitation numbers of $\sigma = 2.5$, $\sigma = 1.5$ and $\sigma = 1.2$. are shown in Figure 6.2 to Figure 6.7.

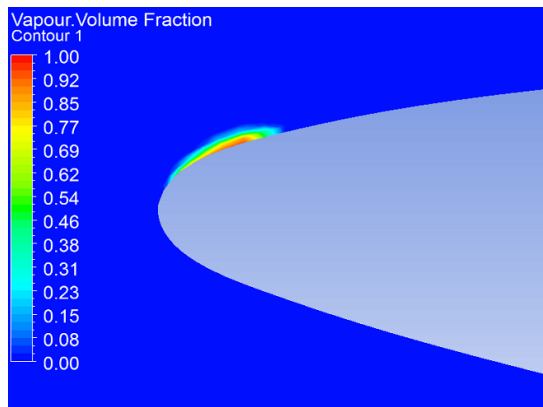


Figure 6.2 - Vapour Volume Fraction, $\sigma = 2.5$, ANSYS CFX

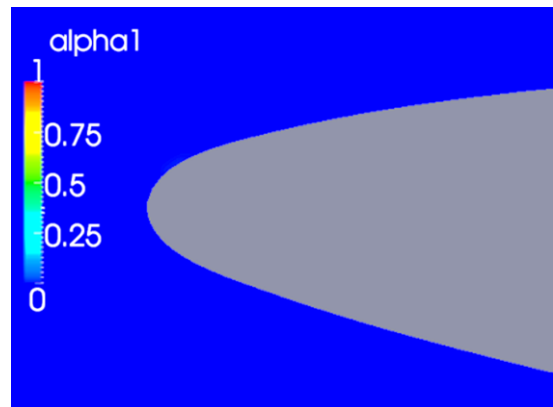


Figure 6.3 - Vapour Volume Fraction, $\sigma = 2.5$, OpenFOAM (Kunz)

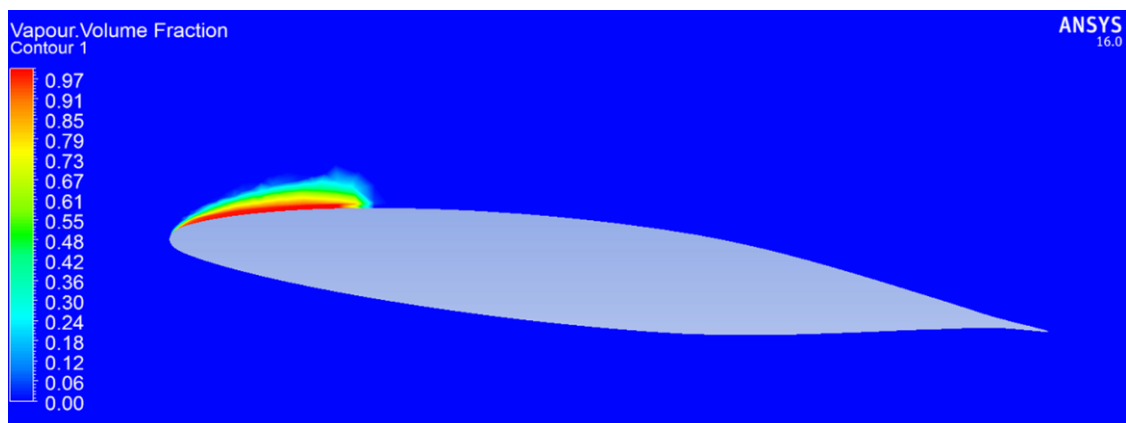


Figure 6.4 - Vapour Volume Fraction, $\sigma = 1.5$, ANSYS CFX

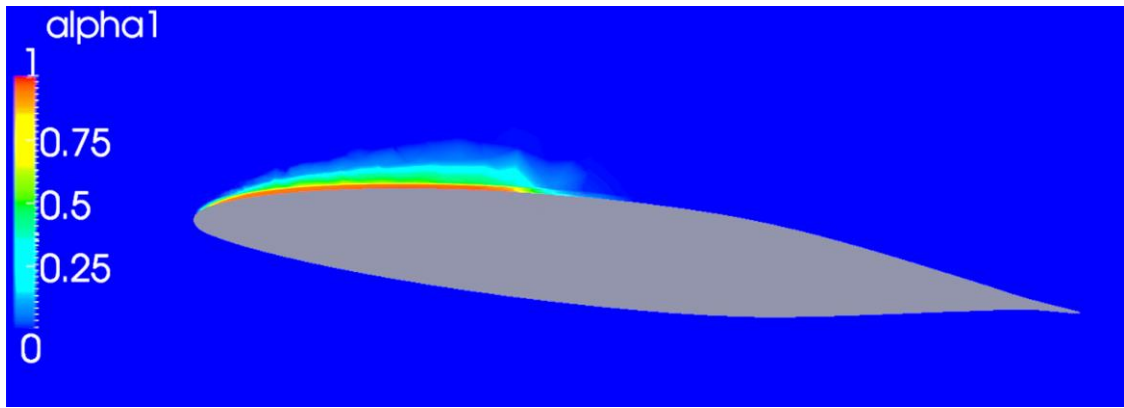


Figure 6.5 - Vapour Volume Fraction, $\sigma = 1.5$, OpenFOAM (Kunz)

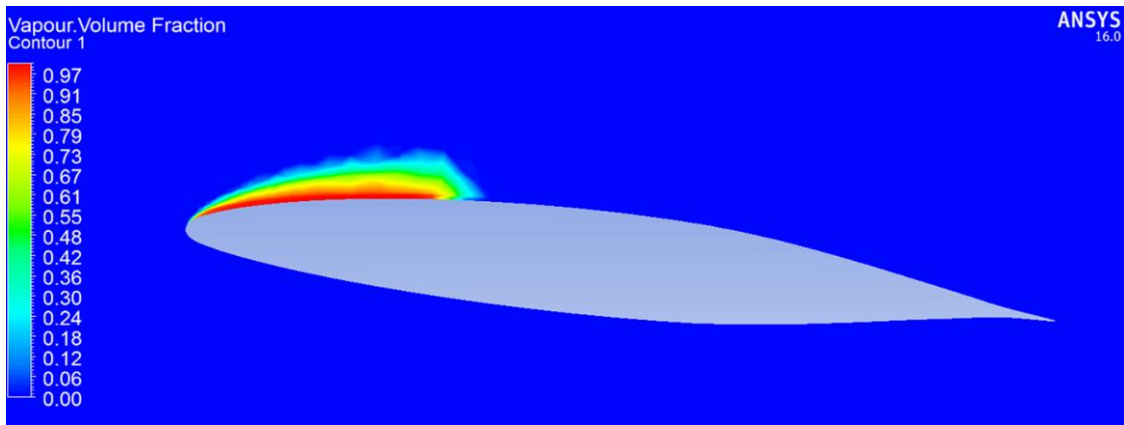


Figure 6.6 – Vapour Volume Fraction, $\sigma = 1.2$, ANSYS CFX

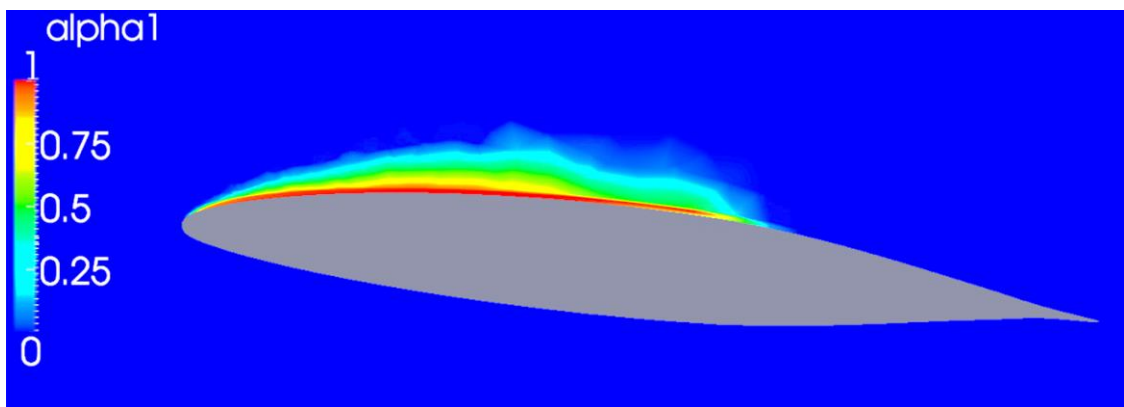


Figure 6.7 - Vapour Volume Fraction, $\sigma = 1.2$, OpenFOAM (Kunz)

Overall, it can clearly be seen that both solvers were capable of predicting the formation of vapour across the NACA66-212 profile based on the prescribed setup conditions. From Figure 6.2 and Figure 6.3 it can be seen that when $\sigma = 2.5$ the initial formation of the

vapour phase occurred using the ANSYS CFX solver, whilst the OpenFOAM (Kunz) model had not yet reached the stage to initiate phase change.

Figure 6.4 and Figure 6.5 once again show a difference in the vapour formed although it can be seen that both predicted a steady attached cavity region emanating from the leading edge of the profile. The ANSYS CFX model shows a more significant vapour region towards the leading edge whereas the OpenFOAM (Kunz) model predicted a cavity length in the region twice that of ANSYS CFX but with less area of entirely vapour at the profile surface. This trend is seen once again in Figure 6.6 and Figure 6.7 which show ANSYS CFX predicting a leading edge dominated cavity region half the length of the OpenFOAM (Kunz) prediction.

Based on this work and the literature available, there is room for significant study into the prediction of the vapour by assessing the impact of parameters, coefficients and so forth. However, the results obtained using both OpenFOAM and ANSYS CFX (at $\sigma = 1.5$) were found to be comparable to those presented in existing research [69]. This similarity was seen to be stronger for the OpenFOAM (Kunz) results which were best aligned to the existing work, particularly examples using the similar Merkle model. In terms of the shape and extent of the length of the cavity, they too exhibited a longer, shallower profile as opposed to the ANSYS CFX model which predicted the shorter, higher region.

Overall, the fact that both solvers were able to predict the vapour formation was deemed sufficient in this instance to move forward to the next phase of the investigations into the capabilities of OpenFOAM.

6.2 OpenFOAM Tutorial of Propeller

Whilst successful initial testing was seen in Section 6.1 where the capability of OpenFOAM to model the phase change from fluid to vapour using `interPhaseChangeFoam` was

demonstrated, further work was required to assess its applicability for centrifugal pumps. Therefore, the next step was to utilise a solver introduced in newer versions of OpenFOAM, `interPhaseChangeDyMFoam`. Whilst this solver is essentially the same as that used in the NACA66-121 study, the code includes the useful addition of the ability to model dynamic meshes and utilise OpenFOAM's Arbitrary Mesh Interface (AMI) functionality.

As part of the OpenFOAM-2.3.x release, a tutorial relating to a rotating propeller in a cylindrical tunnel is used to showcase the functionality of the new solver, with the basic case setup seen in Figure 6.8.

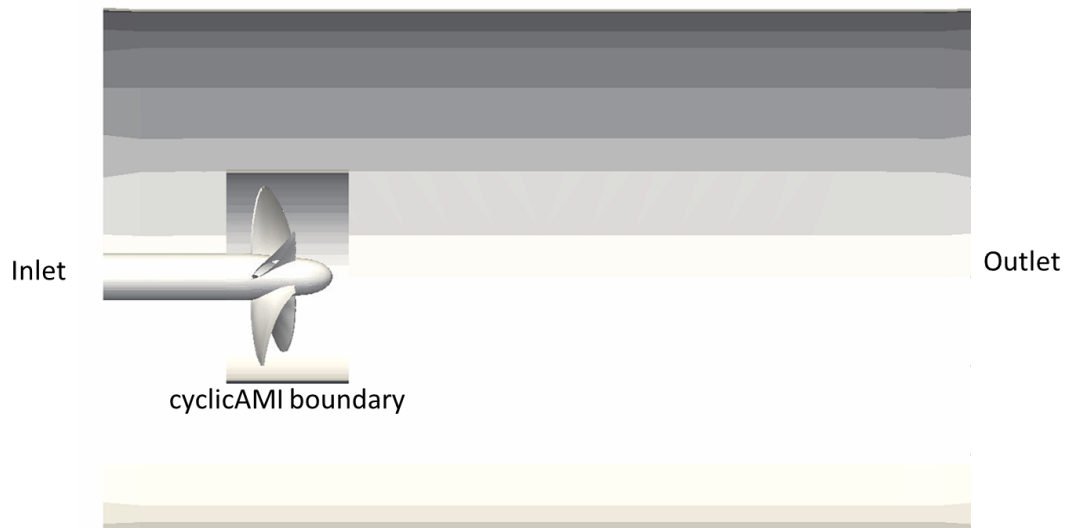


Figure 6.8 – Setup of `interPhaseChangeDyMFoam` Propeller Tutorial

The setup involved a large cylinder representing the tunnel and a smaller cylinder enclosing the propeller which was defined as the rotating domain. The cyclicAMI boundary condition was prescribed for the interface between these cylinders in order to simulate the interaction between the fluid across the boundary between the different reference frames of the propeller region and the surrounding area.

As this was a tutorial case, an 'Allrun' script was contained within the top level case directory and therefore all steps of the simulation process were completed automatically

by the software. However, a high level examination of the code could extract useful information that can be used to determine the setup of the analysis and assist with future work. For example, the mesh was constructed using the blockMesh facility for the main tunnel and snappyHexMesh for the more complex propeller blade geometry as seen in Figure 6.9.

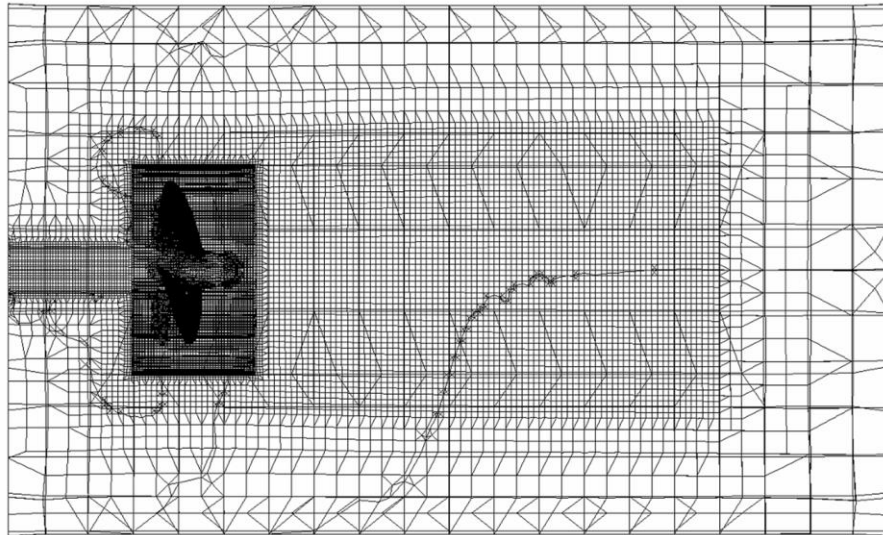


Figure 6.9 – Mesh of interPhaseChangeDyMFoam Propeller Tutorial

The main boundary conditions for the tunnel were prescribed as velocity inlet and pressure outlet. In this instance the velocity was defined in a table and therefore the initial flow into the domain, $U_{inlet} = 0$ m/s however this was then increased to 15 m/s until the final time step 0.1 (time step = 1×10^{-5} s). The rotational speed of the propeller was also defined as a table in the *dynamicMeshDict*, with an initial speed of 0 rpm before being ramped up to 6000 rpm and finally settling at 4000 rpm for the final set of time steps.

By using the 'Allrun' command, the simulation proceeded without any issues, with the propeller rotating within the tunnel. An output from the final result can be seen in Figure 6.10 which shows the level of water vapour on the surface of the blades which can therefore be used to deduce the level of vapour formation.

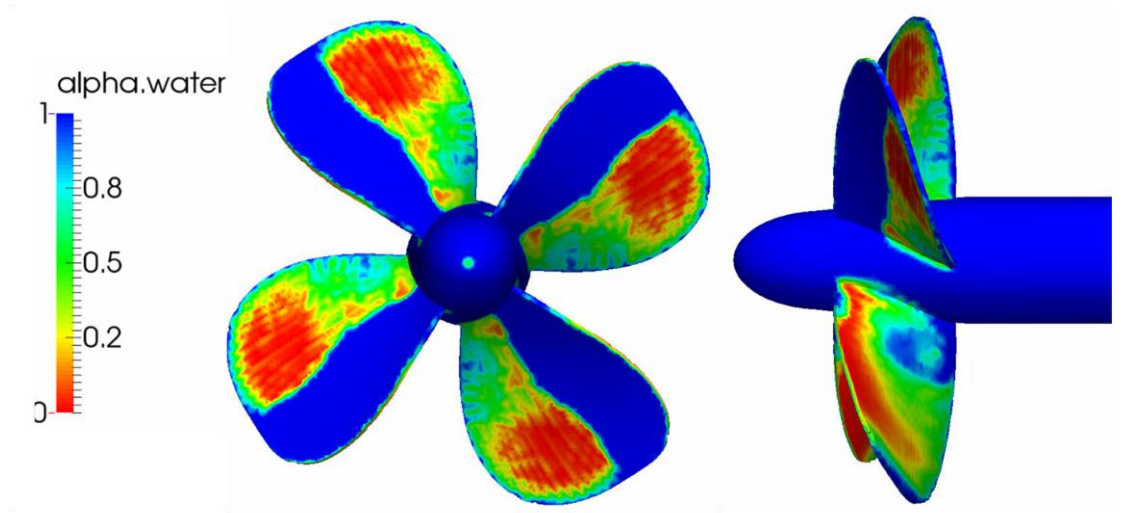


Figure 6.10 – Contours of Liquid Vapour Fraction (alpha.water) using interPhaseChangeDyMFoam

The successful run of the interPhaseChangeDyMFoam tutorial showed that the OpenFOAM code was able to model phase change whilst simultaneously accommodating a case that consisted of a moving mesh and associated interfaces between domains. This capability was important in the overall aim of attempting to model cavitation in the centrifugal pump as the cases were similar and therefore knowledge could be utilised when moving to the next step of assessing OpenFOAM for that specific purpose.

6.3 Warman 8/6 AH Pump

The next phase in the process of reviewing the cavitation modelling capabilities of the software was to attempt to model the formation of vapour in the Warman AH 8/6 centrifugal pump used in Chapter 5. The objective was to generate a head-drop curve ($NPSH_A$ vs generated head) by reducing the suction pressure at the pump inlet, as discussed in Section 2.3.2, and visually inspect the predicted regions of vapour formation at varying suction pressures and flow rates.

6.3.1 ANSYS CFX

The first task was to perform this investigation in the robust solver ANSYS CFX which is used for such applications frequently in the pump industry, utilising the results from the Q_{opt} single-phase transient simulation as previously discussed in Chapter 5 as initial values.

Before the cavitation investigation began, the single-phase simulation was re-run, this time modifying the boundary conditions from a velocity inlet and static pressure outlet to a total pressure inlet and velocity outlet, allowing the suction pressure, and therefore $NPSH_A$, to be modified directly [71]. Following successful completion of this single-phase analysis, the transient two-phase work could commence, maintaining a time step of 5.45×10^{-4} s.

As the objective was to determine the impact of varying the $NPSH_A$ on the head generated by the pump, an understanding of how this parameter is defined in relation to the simulation set up was required, with (6.2) highlighting the relationship between parameters.

$$NPSH_A = \frac{p_{ref} + p_{inlet} - p_v}{\rho g} \quad (6.2)$$

where p_{ref} was the defined atmospheric pressure of 1.013×10^5 Pa, p_{inlet} was the total pressure defined at the inlet (equivalent to a gauge pressure during physical testing), p_v was the vapour pressure of the liquid at the temperature of 298 K (3169 Pa), ρ was the density of the pumped fluid (997 kg/m^3) and g was acceleration due to gravity. All of the components in (6.2) remained fixed during the case studies apart from p_{ref} which was altered in order to vary the $NPSH_A$.

In relation to the initial study at Q_{opt} , the velocity at the outlet was defined as 12.951 m/s and the turbulence quantities remained the same as those in the single-phase work as outlined in Section 4.3.2. The default settings for the cavitation model were used as in Section 6.1.

The initial $NPSH_A$ was set at 60.0 m using a p_{ref} = The inlet pressure, p_{ref} , was then gradually lowered in stages until convergence in order to produce new performance data for each point. The summarised data obtained using this approach is shown Table 6-1 and also plotted in Figure 6.11 to Figure 6.13.

Table 6-1 – Summarised Results of $NPSH_A$ Study using ANSYS CFX at Q_{opt}

p_{inlet} (Pa)	$NPSH_A$ (m)	Pump Head (m)	Head Drop (%)	Impeller Head (m)	Absorbed Power (kW)	Hydraulic Effy. Pump (%)	Hydraulic Effy. Impeller (%)
488700	60.0	59.8	0.0	63.2	154.7	88.8	93.9
348950	45.7	59.5	0.5	62.9	154.6	88.4	93.7
249250	35.5	59.3	0.8	62.7	154.3	88.3	93.5
149550	25.3	58.5	2.2	62.0	153.3	87.7	93.0
99700	20.2	57.9	3.2	61.5	152.2	87.4	92.5
49850	15.1	57.7	3.5	61.0	151.9	87.3	92.2
24925	12.6	57.7	3.5	61.0	152.0	87.2	92.1
0	10.0	57.5	3.8	60.9	152.9	86.4	91.7
-24925	7.5	55.5	7.2	59.0	152.2	83.8	89.4
-34895	6.5	54.0	9.7	58.0	151.3	82.0	87.5
-39880	6.0	53.0	11.4	56.8	152.0	80.1	85.9
-41425	5.8	51.0	14.7	55.0	151.0	77.6	83.7
-42385	5.7	49.9	16.6	53.7	149.9	76.5	82.5

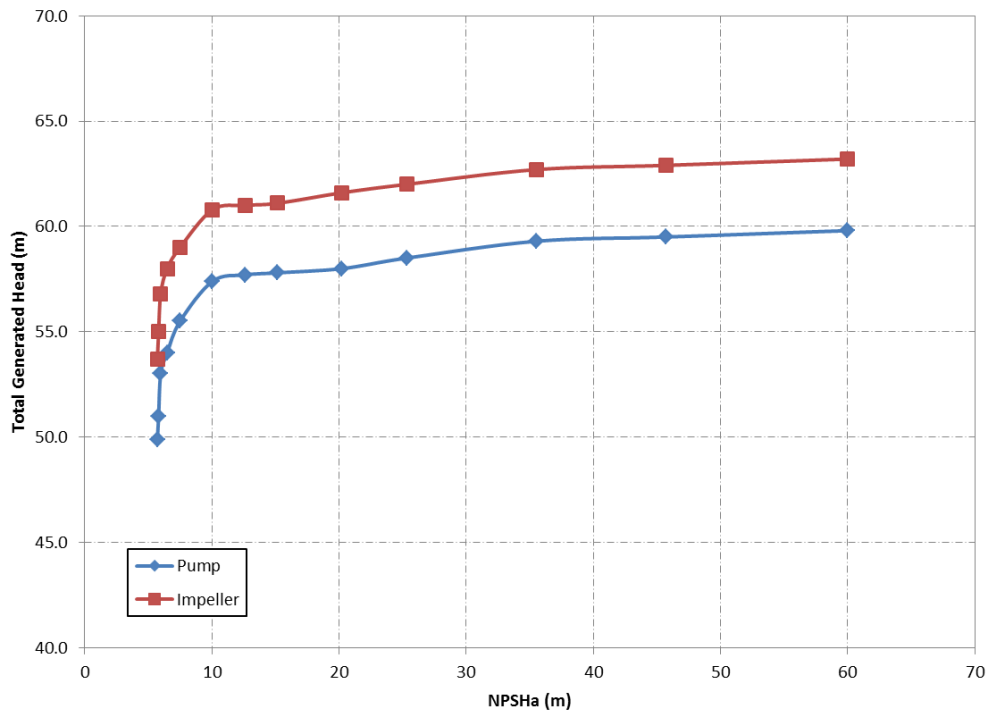


Figure 6.11 – $NPSH_A$ Study Using ANSYS CFX – Generated Head

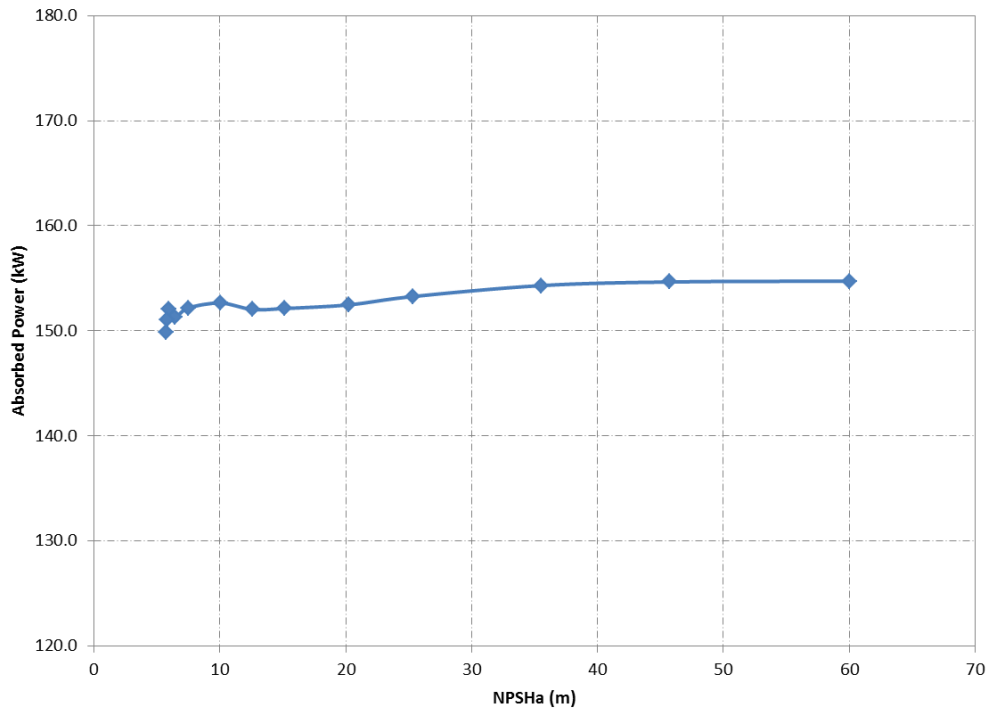


Figure 6.12 – $NPSH_A$ Study Using ANSYS CFX – Absorbed Power

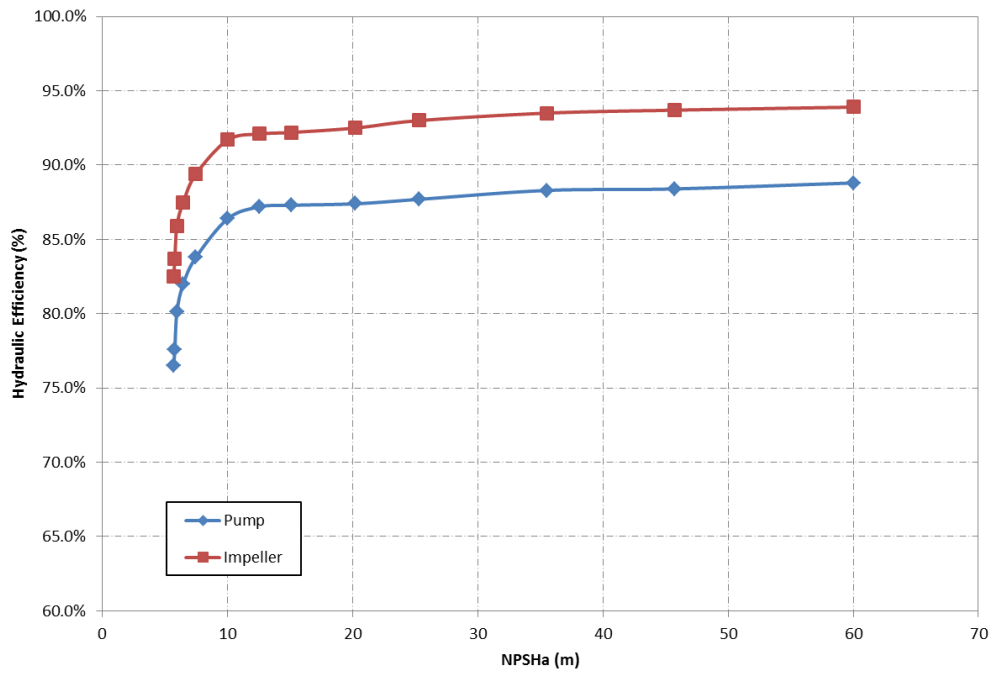


Figure 6.13 – $NPSH_A$ Study Using ANSYS CFX – Hydraulic Efficiency

It can be seen from Table 6-1 and Figure 6.11 to Figure 6.13 that the study was successful in terms of being able to predict the reduction in head generated by the pump by reducing the $NPSH_A$, with the pump and impeller-only generated head values reducing from 59.8 m to 49.9 m and 63.2 m to 53.7 m respectively based on the final values attained.

It can also be seen that the predicted hydraulic efficiency values for the pump and impeller also reduced considerably, from 88.8% to 76.5 % and 93.9 % and 82.5 % for the pump and impeller-only respectively.

By assessing the results further, it can be seen that if the industry-standard approach outlined in Section 2.3.2 is used to define the $NPSH_R$ by considering a head deterioration of 3%, a value in the region of 20 m would be suggested for this flow rate. In terms of complete cavitation breakdown, this is predicted to occur at 5.7 m which was also highlighted by the inability to reach a converged solution in ANSYS CFX after this point.

In order to highlight the considerable impact that the $NPSH_A$ has on the fluid and therefore pump performance, it was of interest to examine the regions of vapour being formed within the impeller. One method of doing so was to illustrate iso-surfaces of vapour volume fraction of 0.25 for different $NPSH_A$ conditions in the investigation as shown in Figure 6.14 to Figure 6.16.

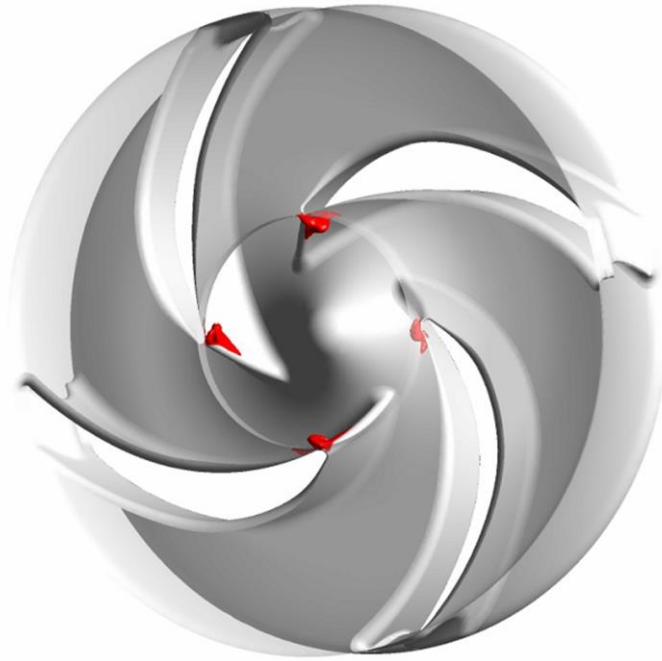


Figure 6.14 – Iso-surface of Water Vapour = 0.25 at $NPSH_A = 20.2$ m

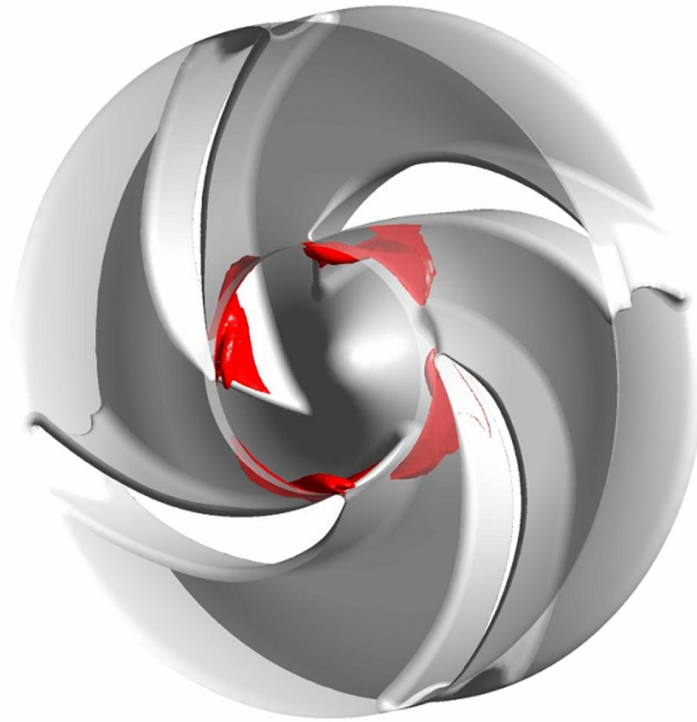


Figure 6.15 – Iso-surface of Water Vapour = 0.25 at $NPSH_A = 10.0$ m



Figure 6.16 – Iso-surface of Water Vapour = 0.25 at $NPSH_A = 5.7$ m

It is clear from Figure 6.14 to Figure 6.16 that $NPSH_A$ has a significant impact on the extent of vapour formation within the impeller. At $NPSH_A = 20.2$ m the cavitation is just beginning to form on the leading edge of the impeller blades at the shroud side which aligns with the initial performance deterioration highlighted previously. At $NPSH_A = 10.0$ m the cavitation has increased. Whilst the initiation point remained the same at the impeller blade leading edge, the region of vapour has extended further down the blade and also onto the shroud surface. At this point it can be seen that the vapour region is beginning to take over an area between the blade passages. At $NPSH_A = 5.7$ m, the cavitation has increased significantly and the region has extended even further down the blade passage and onto the shroud surface. These factors in combination have resulted in a large region of vapour forming between the impeller blades, resulting in the pumped fluid being choked. Again, this correlates well with the performance data in terms of cavitation breakdown occurring at this stage.

In order to review the impact of varying the $NPSH_A$ further, the study was repeated for the two different off-design flows previously examined in Chapter 5, 70% and 130% of Q_{opt} , with the velocity and turbulence boundary conditions modified accordingly. The summarised data from the two investigations are shown in Table 6-2 and Table 6-3 for the 70% and 130% Q_{opt} respectively where it can be seen that the generated head and hydraulic efficiency values once again decrease with decreasing $NPSH_A$.

Table 6-2 – Summarised Results of $NPSH_A$ Study using ANSYS CFX at 70% Q_{opt}

p_{inlet} (Pa)	$NPSH_A$ (m)	Pump Head (m)	Head Drop (%)	Impeller Head (m)	Absorbed Power (kW)	Hydraulic Effy. Pump (%)	Hydraulic Effy. Impeller (%)
488700	60.0	63.3	0.0	67.9	119.1	85.5%	91.6%
348950	45.7	63.3	0.0	67.8	119.2	85.4%	91.5%
249250	35.5	63.0	0.5	67.6	119.2	85.0%	91.3%
149550	25.3	62.5	1.3	66.6	118.7	84.7%	90.4%
99700	20.2	61.9	2.2	66.2	118.4	84.1%	90.0%
49850	15.1	61.9	2.2	66.2	118.5	84.0%	90.0%
24925	12.6	61.9	2.2	66.2	118.2	84.2%	90.1%
0	10.0	61.8	2.4	65.9	118.3	84.0%	90.2%
-24925	7.5	61.7	2.5	65.9	118.1	84.0%	90.1%
-34895	6.5	61.7	2.5	66.0	118.4	83.8%	89.7%
-44360	5.5	61.3	3.2	65.9	118.8	83.0%	89.3%
-49245	5.0	60.8	3.9	65.3	119.0	82.2%	88.3%
-59020	4.0	58.6	7.4	63.6	120.3	78.3%	85.2%

Table 6-3 – Summarised Results of $NPSH_A$ Study using ANSYS CFX at 130% Q_{opt}

p_{inlet} (Pa)	$NPSH_A$ (m)	Pump Head (m)	Head Drop (%)	Impeller Head (m)	Absorbed Power (kW)	Hydraulic Effy. Pump (%)	Hydraulic Effy. Impeller (%)
488700	60.0	55.1	0.0	58.5	188.3	87.4%	93.3%
348950	45.7	54.8	0.5	58.4	187.7	87.2%	93.3%
249250	35.5	54.6	0.9	58.4	187.5	87.0%	93.2%
149550	25.3	54.5	1.1	58.3	187.5	86.8%	93.1%
99700	20.2	53.5	2.9	56.9	185.2	86.3%	92.4%
49850	15.1	51.7	6.2	55.6	181.9	84.9%	91.1%
24925	12.6	50.6	8.2	54.4	181.4	83.3%	89.7%
9450	11.0	48.9	11.3	53.1	182.8	79.9%	87.0%
0	10.0	46.2	16.2	50.7	184.0	75.0%	82.5%
-5250	9.5	39.0	29.2	45.7	182.0	64.0%	75.2%

In order to review the data whilst determining the impact of the prescribed flow rate on deterioration in generated head, the pump generated head and hydraulic efficiency values for the three defined flow rates were plotted together as seen in Figure 6.17 and Figure 6.18.

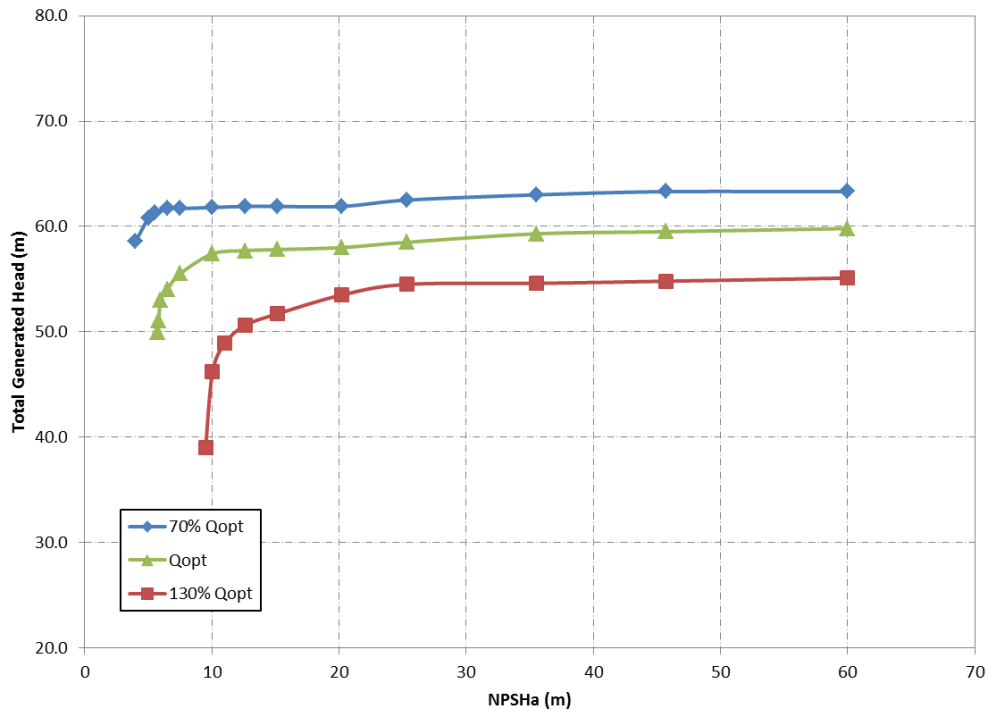


Figure 6.17 – Impact of Varying $NPSH_A$ for All Flow Conditions – Pump Generated Head

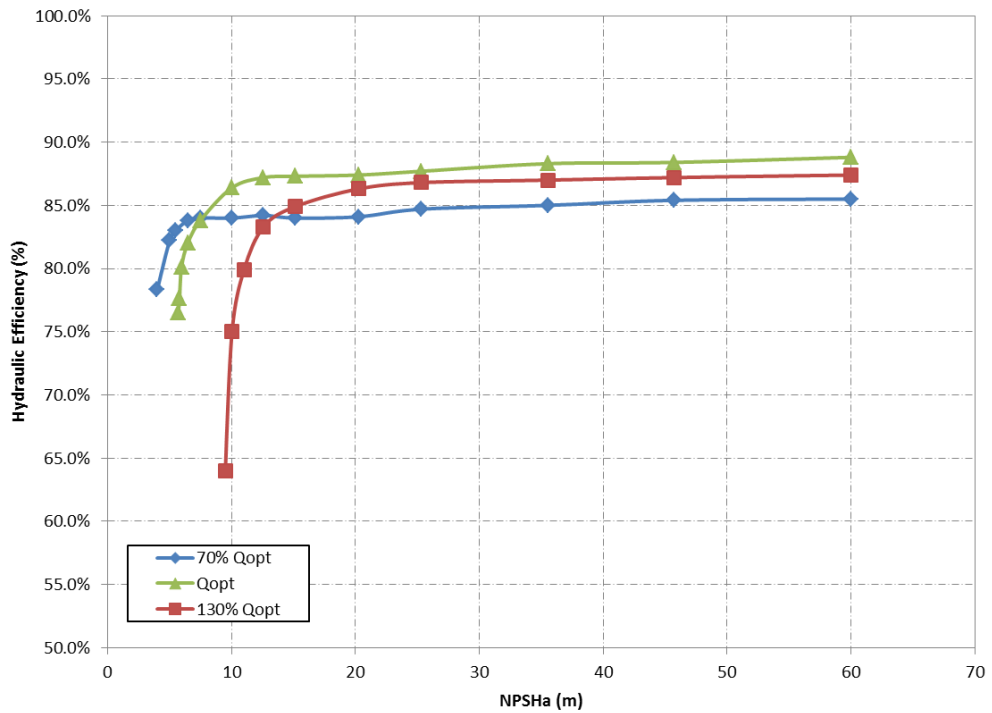


Figure 6.18 – Impact of Varying $NPSH_A$ for All Flow Conditions – Pump Hydraulic Efficiency

From the data it can be seen that pump flow has a significant impact on the $NPSH_A$ at which a substantial deterioration in performance is seen before resulting in cavitation breakdown. At the higher flow of 130% Q_{opt} cavitation is predicted to have an effect on generated head performance to the extent of a 3% head reduction at around $NPSH_A = 20$ m, similar to that of Q_{opt} . However, the larger deterioration occurs at a relatively higher value and resultant cavitation breakdown occurs at $NPSH_A = 9.5$ m. At the lower flow of 70% Q_{opt} the 3% head drop is not predicted to be an issue until a relatively low value of $NPSH_A = 6.0$ m, with cavitation breakdown occurring at $NPSH_A = 4.0$ m. The trends seen in the changes in generated head are mirrored when looking at the impact of $NPSH_A$ on hydraulic efficiency.

Overall, the methodology used in this instance in conjunction with the ANSYS CFX solver was able to predict the occurrence of cavitation in the centrifugal pump for varying $NPSH_A$ and flow conditions. A review of the performance data showed that a deterioration of generated head was seen when the $NPSH_A$ was reduced, with the critical values of $NPSH_A$

differing dependent on the pump flow. This performance data was also reinforced by reviewing the vapour formation within the impeller, with larger regions of vapour predicted to occur with reducing $NPSH_A$.

6.3.2 OpenFOAM

The next phase in the process was to attempt to use the knowledge from the previous cavitation work in OpenFOAM in Sections 6.1 and 6.2 together with the study of the centrifugal pump in ANSYS CFX in Section 6.3.1 to model cavitation in the Warman 8/6 AH pump using an OpenFOAM solver.

Despite significant attempts made in both OpenFOAM-2.1.x and OpenFOAM-2.3.x, this was not achieved during the time frame of this particular work. However, a summary of the work undertaken is provided which highlights the issues encountered to date.

6.3.3 OpenFOAM-2.1.x

The initial attempts to model cavitation in the pump were undertaken in OpenFOAM-2.1.x as this was the version used for the previous single-phase work. At that time, the `interPhaseChangeFoam` solver existed however there was no version that included the ability to model a moving mesh required for the transient analyses and the official `interPhaseChangeDyMFoam` solver had not been released at this point. The majority of work in relation to modelling cavitation in the pump using OpenFOAM was therefore based on this version of the code.

The first aspect that was required to be addressed was to create a new solver by merging several aspects of code together, using the basic framework of the `interPhaseChangeFoam` solver to do so. Based on the work conducted in Chapter 5, the decision was made that the standard `PIMPLE` loop would be replaced with the solver loop seen in

transientSimpleDyMFoam. The aspects related to time-dependency were then added accordingly to the modified solver.

Another important change was made to the references of pressure within the overall solver file. The two-phase solver uses a 'standard' pressure, p_rgh , whereas the single-phase solver uses p (where $p = p_rgh/\rho$). By modifying the code in a particular manner, the results from the single-phase work in chapter 5 could be used directly as initial conditions. Following significant work, a solver that was able to be compiled was created and the next stage was to undertake testing.

As mentioned, the modifications to the code allowed the results from Chapter 5 to be used directly in the solver, however this was not before a single-phase run had been completed in order to modify the boundary conditions from velocity inlet and static pressure outlet to total pressure inlet and velocity outlet as per the ANSYS CFX work in Section 6.3.1. Whilst the Kunz cavitation model was selected, the inlet pressure was set high enough in order that no vapour would be expected to form during the initial run, therefore allowing the code to be tested.

Following many attempts, the code did work to the extent that results were produced which were in line with expectation for the generated head and hydraulic efficiency for both the pump and impeller-only values. The inlet pressure was then dropped in order to ascertain whether the solver was working completely correctly, with the results being reviewed with every inlet pressure modification. An example of the results can be seen in Figure 6.19 for a $NPSH_A = 7.3$ m.

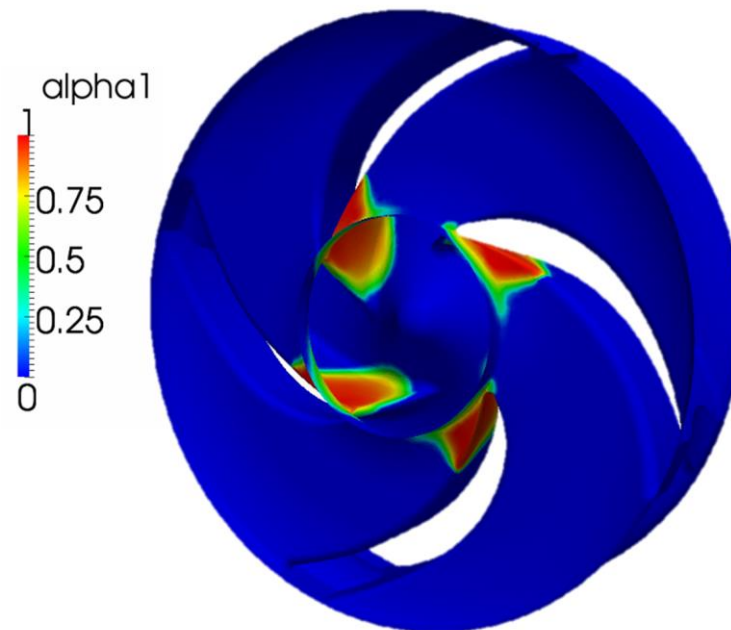


Figure 6.19 – OpenFOAM (Kunz) Vapour Volume Fraction Prediction, $NPSH_A = 7.3$ m

It can be seen from Figure 6.19 that the prediction of vapour was successful and that the cavitation was being correctly predicted along the blade and shroud surfaces of the impeller. Reviewing additional results of lower $NPSH_A$ also correctly identified larger regions of cavitation.

However, despite this success, a review of the performance data for the pump and impeller at varying $NPSH_A$ showed that no deterioration in performance was being predicted. Following a more in depth review of the results and also the approach for reporting this output data through the code, the source of the issue was identified as the solver code. When merging the solvers, the files containing the pressure and velocity equations (pEqn.H and UEqn.H) from the transientSimpleDyMFoam solver had remained the same, solving for a single-phase fluid. This made sense as the cavitation model was working correctly and was able to predict regions of vapour however the additional properties on which head and power are dependent were incorrect due to the code behind the solver.

Following the identification of the issue, significant efforts were made to update the pressure and velocity code to the correct two-phase versions. However, following this modification, further simulation attempts identified a new fundamental problem: the total generated head developed by the pump was greater than that of the impeller in isolation. Given that the pump generated head should be less than that of the impeller due losses in the volute casing, there was clearly an issue in the simulation as highlighted by Figure 6.20.

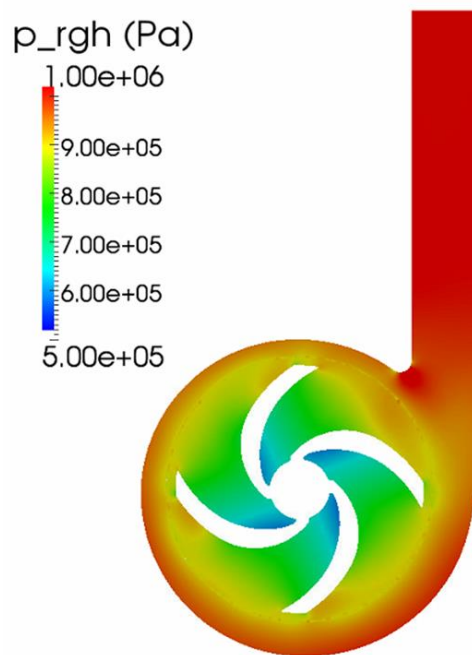


Figure 6.20 – Slice of Pump Highlighting Pressure Simulation Issues

It appeared that the problem was linked with the interface between the rotating impeller and stationary volute casing however attempts to address this through the review of the cyclicAMI boundary condition set up yielded no improvement. Considerable effort was also spent reviewing inlet/outlet boundary conditions, cavitation model selection, turbulence models, rotational speed as well as other aspects such as solver control of correction loops and numerical schemes. However, despite this, the problem remained unresolved.

6.3.4 OpenFOAM-2.3.x

Following ultimately unsuccessful attempts using a newly created solver in OpenFOAM-2.1.x, the newer version of the code, OpenFOAM-2.3.x was downloaded with the new solver `interPhaseChangeDyMFoam` that was, as seen in Section 6.2, capable of accepting moving meshes.

Following the experience obtained in creating the new two-phase solver in OpenFOAM-2.1.x and the tutorial case reviewed in Section 6.2, new attempts were made in order to successfully run the simulation.

The initial approach was once again to utilise the results from the single-phase work undertaken in Chapter 5, firstly re-running the simulation using the total pressure inlet boundary condition configuration. Following this, the file containing the pressure data for the final time step of the single-phase work was modified from p to p_rgh in this solver by using a custom made converter utility. The dimensions were also modified accordingly. Following this, the case was setup using the files from the OpenFOAM-2.1.x and propeller studies as reference.

Despite the formally released solver code and new tutorial case to use as an ongoing guide, the overall results using the OpenFOAM-2.3.x solver code were the same as those using the custom-built OpenFOAM-2.1.x code and as such were unable to predict cavitation in the centrifugal pump successfully.

6.4 Conclusions

The initial work conducted in this chapter showed that it was possible to predict the occurrence of cavitation by using the two phase solver `interPhaseChangeFoam` solver in OpenFOAM-2.1.x, with results compared against ANSYS CFX for reference.

Additional work illustrated how a phase change simulation including a moving mesh and cyclicAMI interfaces could be modelled using the tutorial of a rotating propeller in OpenFOAM-2.3.x in which the interPhaseChangeDyMFoam solver was used.

Finally, transient two-phase work was undertaken using ANSYS CFX and OpenFOAM to model cavitation prediction in a centrifugal pump with varying success. The ANSYS CFX solver was capable of predicting the formation of the vapour phase due to cavitation, which was highlighted through deterioration in performance with decreasing $NPSH_A$ and through a review of the vapour volume fraction within the impeller region. The attempt to repeat this using OpenFOAM was ultimately unsuccessful despite attempts using a custom-built solver in OpenFOAM-2.1.x and an officially released solver (interPhaseChangeDyMFoam) in OpenFOAM-2.3.x.

Chapter 7

7. Conclusions and Future Work

7.1 Summary

This thesis has investigated the use of open source CFD code OpenFOAM as an alternative to well-established software packages due to their increasingly prohibitive licensing costs. To achieve this overall objective, a centrifugal pump case study was considered and results of both steady state and transient moving mesh single-phase analyses conducted in OpenFOAM were compared against results using the commercial software package ANSYS CFX.

In order to take this a stage further, the problematic phenomena of cavitation was reviewed in greater detail with an initial assessment of cavitation and cavitation erosion modelling techniques. This was followed by an investigation into the potential use of OpenFOAM to model the occurrence of cavitation in the centrifugal pump case study with concurrent analysis undertaken using ANSYS CFX.

To achieve these aims, an introduction to the requirement of highly efficient turbomachinery systems and identification of cavitation as the most common cause of failure was given in Chapter 1. The significant limitations, associated with the licensing costs of current CFD modelling software, were also discussed and the requirement for investigating potential alternatives such as OpenFOAM was identified.

Chapter 2 introduced the theoretical concepts of cavitation, standard methods of measurement and presented an overview of the main quantitative modelling techniques currently available, discussing the particular advantages and drawbacks associated with each.

This investigation of modelling techniques was taken a stage further in Chapter 3 by examining work in the field of cavitation erosion modelling in which the complexity of the models used for quantitative prediction of material degradation was identified. Whilst the models reviewed had positive results, significant testing is required for more complex geometries in order to fully evaluate them under more realistic situations.

Chapter 4 introduced the OpenFOAM code, discussed the various solvers that were used in the single-phase, transient and cavitating flow simulations as well as the utilities that allow the desired information to be tracked during the solving period. The Weir Warman 8/6 AH centrifugal pump used as the main method of demonstrating the capabilities of each solver, was also introduced.

Chapter 5 compared the single-phase modelling capability of OpenFOAM against commercial CFD software, ANSYS CFX, using the centrifugal pump case study. Several investigations were performed to assess the sensitivity of OpenFOAM to solver the used, time step size, time discretisation scheme, number of internal corrector iteration loops and velocity convection scheme. Finally, the results obtained from the down-selected setup using OpenFOAM were validated against outputs from the commercial CFD software ANSYS CFX and in general were found to compare favourably in terms of both performance figures and flow regime in the impeller and volute casing.

A final assessment of the software packages was conducted in Chapter 6 where the basic ability of each to predict the occurrence of cavitation was demonstrated through simple

test cases. The next stage was to predict cavitation in the centrifugal pump. The ANSYS CFX solver was capable of predicting the formation of the vapour phase and deterioration in performance with decreasing $NPSH_A$. Attempts to repeat this using OpenFOAM were ultimately unsuccessful however preliminary results showed that the simulation of this process should be possible given adequate time to specifically investigate this.

7.2 Conclusions

Following a summary of the work presented in Section 7.1, distinct conclusions can be drawn in relation to the objectives first stated in Chapter 1.

The results of OpenFOAM for unsteady single-phase analyses were comparable to the commercial code ANSYS CFX for the centrifugal pump case study.

For two-phase simulations on simple geometries of pre-built test cases, the OpenFOAM solver was capable of predicting cavitation. In progression to the more complex geometry of the centrifugal pump, the code was unable to replicate the results seen in the ANSYS CFX solver within the time frame of this research. However, once again it is expected that with additional resource, the code could be set up to model this behaviour successfully.

Overall, OpenFOAM is deemed to be a viable alternative to commercial software packages when considering single-phase centrifugal pumps or two-phase flow for less complex geometries at present.

7.3 Future Work

This thesis has reviewed the capabilities of OpenFOAM for single-phase and cavitating flow in relation to a centrifugal pump. However there is scope to expand and improve the research in the following ways:

- A review of more a complex centrifugal pump case study that includes as many additional features of the pump assembly as possible such as leakage paths, wear plates and balance holes in order to obtain as accurate results as possible
- A review of a multi-stage centrifugal pump case study which would be particularly useful given that the OpenFOAM solver is not prohibited by licensing costs typically encountered with the increased computational effort required for such work
- Additional research into the cavitation prediction capabilities of OpenFOAM for more complex systems such as centrifugal pumps

8. References

- [1] Hydraulic Institute, Europump, and Office of Industrial Technologies - US Department of Energy, "Pump Life Cycle Costs: A Guide to LCC Analysis for Pumping Systems - Executive Summary," 2001.
- [2] British Pump Manufacturers Association, "Pump Industry Association Sustainable Energy Strategy - Executive Summary," 2007.
- [3] T. C. Zin, "An Experimental and Field Study of Cavitation Detection in Pump," Universiti Teknologi Malaysia, 2007.
- [4] B. Schiavello and F. C. Visser, "Pump Cavitation - Various NPSHr Criteria, NPSHa Margins, and Impeller Life Expectancy," in *Twenty Fourth International Pump Users Symposium*, 2008, no. 1985.
- [5] J.-P. Franc and J.-M. Michel, *Fundamentals of Cavitation*. 2004.
- [6] C. E. Brennan, *Cavitation and Bubble Dynamics*, vol. 4, no. 2. Oxford University Press, 1997.
- [7] C. Bramanti, "Experimental Study of Cavitation and Flow Instabilities in Space Rocket Turbopumps and Hydrofoils," Università degli Studi di Pisa, 2008.
- [8] E. Grist, *Cavitation And The Centrifugal Pump: A Guide For Pump Users*. CRC Press, 1998.
- [9] A. A. M. Al-chalaby, "Cavitation in Process Pumps." .
- [10] S. Yedidiah, "The Meaning and Application-limits of Thoma's Cavitation Number," *Cavitation Polyph. Flow Forum*, pp. 45–6, 1981.
- [11] J.-P. Franc, "The Rayleigh-Plesset Equation : A Simple and Powerful Tool to Understand Various Aspects of Cavitation," in *Fluid Dynamics of Cavitation and Cavitating Turbopumps*, Springer Vienna, 2007, pp. 1–41.
- [12] E. Goncalves and R. F. Patella, "Numerical simulation of cavitating flows with homogeneous models," *Comput. Fluids*, vol. 38, no. 9, pp. 1682–1696, 2009.
- [13] Y.-J. Wei, C.-C. Tseng, and G.-Y. Wang, "Turbulence and cavitation models for time-dependent turbulent cavitating flows," *Acta Mech. Sin.*, vol. 27, no. 4, pp. 473–487, Jul. 2011.
- [14] C. E. Brennan, "A Numerical Solution of Axisymmetric cavity flows," *J. Fluid Mech.*, vol. 37, pp. 671–688, 1969.

- [15] R. A. Furness and S. P. Hutton, "Experimental and Theoretical Studies of Two-dimensional Fixed-type Cavities," *J. Fluids Eng.*, pp. 515–522, 1975.
- [16] Y. Chen and S. D. Heister, "A Numerical Treatment for Attached Cavitation," *J. Fluids Eng.*, vol. 116, pp. 613–618, 1994.
- [17] M. Deshpande, J. Feng, and C. L. Merkle, "Numerical Modeling of the Thermodynamic Effects of Cavitation," *J. Fluids Eng.*, vol. 119, p. 153, 1997.
- [18] J. L. Kueny, F. Schultz, and J. Desclaud, "Numerical prediction of partial cavitation in pumps and inducers.," in *IAHR Symposium*, 1998.
- [19] J. . Peallat and C. Pellone, "Experimental validation of two and three dimensional numerical analysis of partially cavitating hydrofoils," *J. Sh. Res.*, vol. 40, no. 3, pp. 211–223, 1996.
- [20] S. A. Kinnas and N. E. Fine, "A numerical non-linear analysis of the flow around two and three dimensional partially cavitating hydrofoils.," *J. Fluid Mech.*, vol. 254, pp. 151–181, 1993.
- [21] Y. Chen and S. D. Heister, "Modeling Hydrodynamic No- equilibrium in Cavitating Flows," *J. Fluids Eng.*, vol. 118, no. 172, 1996.
- [22] Q. Qin, C. C. S. Song, and R. E. A. Arndt, "A Virtual Single-Phase Natural Cavitation Model And its Application to Cav2003 Hydrofoil," pp. 1–8, 2003.
- [23] J. L. Reboud and Y. Delannoy, "Numerical simulation of the unsteady behaviour of cavitating flows," *Int. J. Numer. Methods Fluids*, vol. 548, no. September 2001, pp. 527–548, 2003.
- [24] Y. Delannoy and J. L. Kueny, "Two phase flow approach in unsteady cavitation modelling," in *Cavitation and Multiphase Flow Forum, ASME-FED*, 1990, pp. 153–158.
- [25] H. W. M. Hoeijmakers, M. E. Janssens, and W. Kwon, "Numerical Simulation of Sheet Cavitation," in *3rd International Symposium on Cavitation*, 1998.
- [26] C. C. S. Song and G. He, "Numerical Simulation of Cavitating flow by Single-Phase Flow Approach," in *3rd International Symposium on Cavitation*, 1998.
- [27] M. Dular, R. Bachert, B. Stoffel, and B. Širok, "Experimental evaluation of numerical simulation of cavitating flow around hydrofoil," *Eur. J. Mech. - B/Fluids*, vol. 24, no. 4, pp. 522–538, Jul. 2005.
- [28] M. Hofmann, "Ein Beitrag zur vermindering des erosiven Potentials kavitierender Stoemungen –," *Technische Univesitaet Darmstadt*, Darmstadt, San Francisco, CA, 2001.

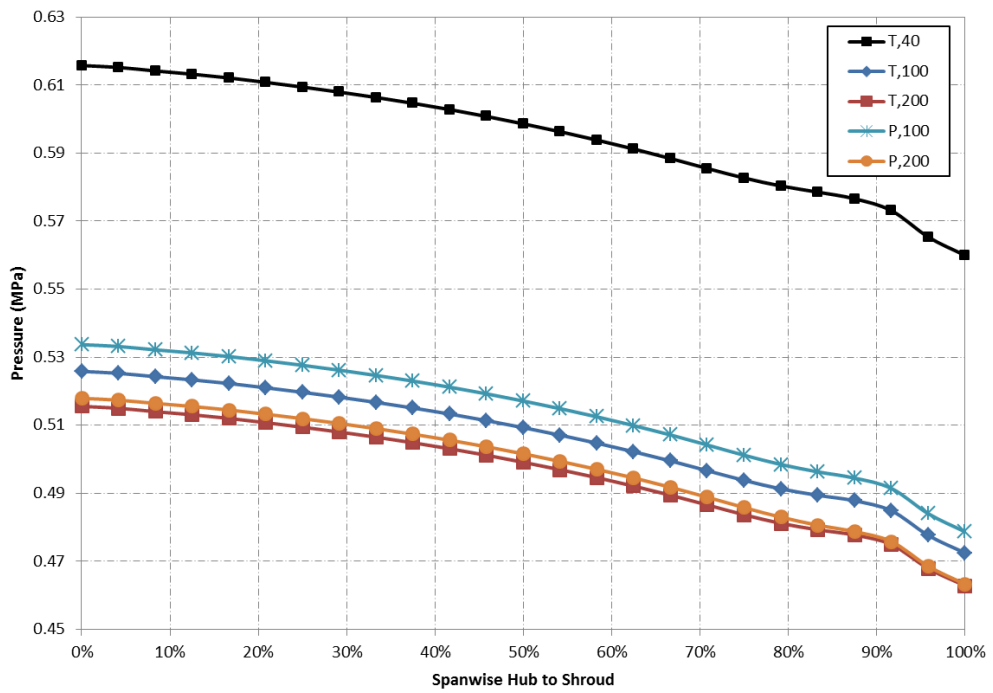
- [29] K. Habr, "Gekoppelte Simulation hydraulischer Gesamtsysteme unter Einbeziehung von CFD," Technische Universität Darmstadt, Darmstadt, 2001.
- [30] A. Kubota, H. Kato, and H. Yamaguchi, "A new modelling of cavitating flows : a numerical study of unsteady cavitation on a hydrofoil section," vol. 240, 1992.
- [31] G. H. Schnerr and J. Sauer, "Physical and numerical modelling of unsteady cavitation dynamics," in *4th International Conference on Multiphase Flow*, 2001.
- [32] M. Frobenius, R. Schilling, R. Bachert, and B. Stoffel, "Three-dimensional, unsteady cavitation effects on a single hydrofoil and in a radial pump – measurements and numerical simulations," in *Proceedings of the Fifth International Symposium on Cavitation*, 2003.
- [33] P. J. Roache, *Computational Fluid Dynamics*. Albuquerque: Hermosa, 1976.
- [34] C. L. Merkle, J. Feng, and Buelow, "Computational Modeling of the Dynamics of Sheet Cavitation," in *3rd International Symposium on Cavitation*, 1998.
- [35] R. F. Kunz, T. S. Chyczewski, D. A. Boger, D. R. Stinebring, and H. J. Gibeling, "Multi-phase CFD Analysis of Natural and Ventilated Cavitation About Submerged Bodies," in *3rd ASME/JSME Joint Fluids Engineering Conference*, 1999.
- [36] A. K. Singhal, M. M. Athavale, H. Li, and Y. Jiang, "Mathematical Basis and Validation of the Full Cavitation Model," *J. Fluids Eng.*, vol. 124, pp. 617–624, 2002.
- [37] I. Senocak and W. Shyy, "Interfacial dynamics-based modelling of turbulent cavitating flows , Part-1 : Model development and steady-state computations," *Int. J. Numer. Methods Fluids*, pp. 975–995, 2004.
- [38] N. A. Owis and A. H. Nayfeh, "Numerical simulation of 3-D incompressible, multi-phase flows over cavitating projectiles," *Eur. J. Mech. B Fluids*, pp. 339–351, 2004.
- [39] R. B. Medvitz, R. F. Kunz, D. a. Boger, J. W. Lindau, A. M. Yocum, and L. L. Pauley, "Performance Analysis of Cavitating Flow in Centrifugal Pumps Using Multiphase CFD," *J. Fluids Eng.*, vol. 124, no. 2, p. 377, 2002.
- [40] B. Huang and G. Wang, "A modified density based cavitation model for time dependent turbulent cavitating flow computations," *Chinese Sci. Bull.*, vol. 56, no. 19, pp. 1986–1992, Jun. 2011.
- [41] A. Kubota, "Unsteady Structure Measurement of Cloud Cavitation on a Foil Section Using Conditional Sampling Technique," *J. Fluids Eng.*, vol. 111, pp. 204–210, 1989.
- [42] D. A. Drew, "Mathematical modelling of two-phase flow," *Ann. Rev. Fluid Mech.*, vol. 15, pp. 261–291, 1983.
- [43] O. Simonin, "Eulerian prediction of a turbulent bubbly flow of a sudden pipe expansion," in *Proceeding of the 6th Workshop on Two-Phase Flow Prediction*, 1992.

- [44] S. Mimouni, M. Boucker, J. Laviéville, A. Guelfi, and D. Bestion, "Modelling and Computation of Cavitation and Boiling Bubbly Flows with the NEPTUNE_CFD Code," *Nucl. Eng. Des.*, vol. 238, no. 3, pp. 680–692, Mar. 2008.
- [45] Jones et al., "A active cavity model for flashin," *Nucl. Eng. Des.*, pp. 185–196, 1986.
- [46] R. W. Erney, "Verification and Validation of Single Phase and Cavitating Flows Using an Open Source CFD Tool," The Pennsylvania State University, 2008.
- [47] F. . Hamitt, "Observations on Cavitation Damage in a Flowing System," *J. Basic Eng.*, p. 3, 1963.
- [48] R. Fortes-Patella, J. L. Reboud, and L. Brianconn-Marjollet, "A phenomenological and numerical model for scaling the flow aggressiveness in cavitation erosion," in *EROCAV Workshop*, 2004.
- [49] G. Bark, J. Freisch, G. Kuiper, and J. Ligtelijn, "Cavitation Erosion on Ship Propellers and Rudders," in *9th Symposium on Practical Design of Ships and Other Floating Structures*, 2004.
- [50] M. Dular and O. Coutier-Delgosha, "Numerical modelling of cavitation erosion," *Int. J. Numer. Methods Fluids*, no. 61, pp. 1388–1410, 2009.
- [51] B. M. S. Plesset and R. B. Chapman, "Collapse of an initially spherical vapour cavity in the neighbourhood of a solid boundary," *J. Fluid Mech.*, vol. 47, pp. 283–290, 1971.
- [52] W. Lauterborn and H. Bolle, "Experimental investigations of cavitation-bubble collapse in the neighbourhood of a solid boundary," *J. Fluid Mech.*, vol. 72, pp. 391–401, 1975.
- [53] M. Dular, B. Stoffel, and B. Širok, "Development of a Cavitation Erosion Model," *Wear*, vol. 261, no. 5–6, pp. 642–655, Sep. 2006.
- [54] H. Kato, A. Konno, M. Maeda, and H. Yamaguchi, "Possibility of quantitative prediction of cavitation erosion without a model test," *J. Fluid Eng.*, vol. 118, 1996.
- [55] G. Bark, N. Berchiche, and M. Grekula, *Application of principles for observation and analysis of eroding cavitation - the EROCAV observation handbook*. 2004.
- [56] M. Shimada, T. Kobayashi, and Y. Matsymuto, "Dynamics of cloud cavitation and cavitation erosion," in *Proceedings of the ASME/JSME Fluids Engineering Dvission Summer Meeting*, 1999.
- [57] L. L. Beranek, "Acoustics, Acoustical Society of America." New York, 1996.
- [58] P. A. Lush, "Impact of a liquid mass on a perfectly plastic solid," *J. Fluid Mech.*, vol. 135, pp. 373–387, 1983.

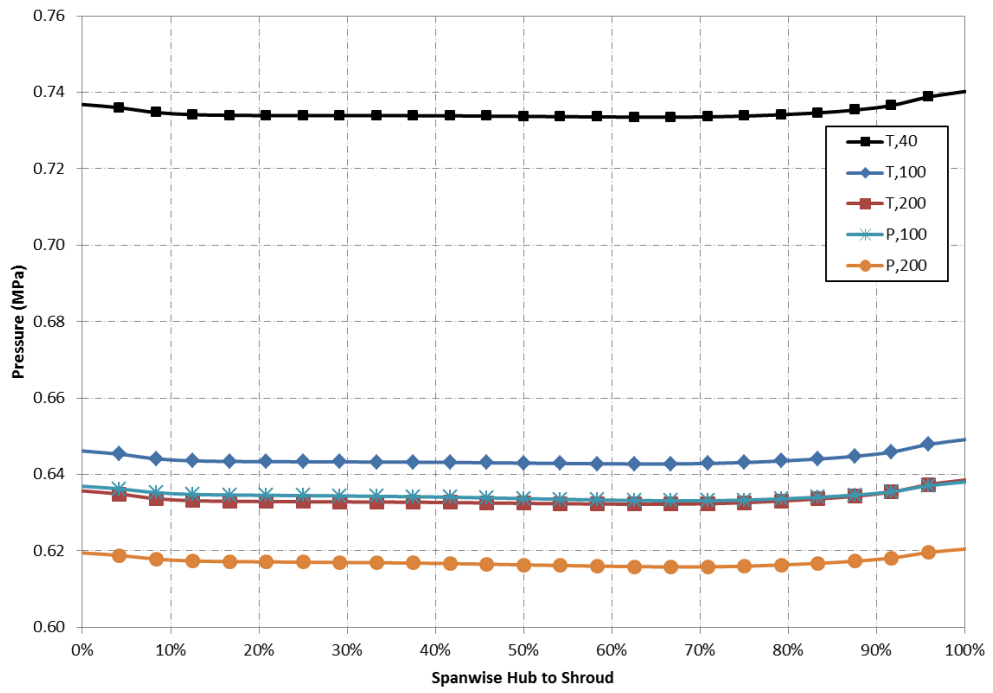
- [59] M. Dular, B. Bachert, B. Stoffel, and B. Širok, "Relationship between cavitation structures and cavitation damage," *Wear*, vol. 257, no. 11, pp. 1176–1184, Dec. 2004.
- [60] B. Bachert, G. Ludwig, B. Stoffel, B. Sirok, and M. Novak, "Experimental Investigations Concerning Erosive Aggressiveness of Cavitation in a Radial Test Pump With the Aid of Adhesive Copper Films," in *Proceedings of the 5th International Symposium on Cavitation*, 2003.
- [61] F. R. Menter, "Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications," *AIAA J.*, vol. 32, no. 8, pp. 1598–1605, 1994.
- [62] M. Auvénin, H. Nilsson, D. A. Boger, and B. Lewis, "Sig Turbomachinery Library turboPerformance," 2010. [Online]. Available: https://openfoamwiki.net/index.php/Sig_Turbomachinery_Library_turboPerformance.
- [63] Weir Minerals Australia (PTC), "Typical Pump Performance Curve WPA86A03/2." 2011.
- [64] W. Nicholls and W. Dempster, "CFD Modelling of Single Phase Flow in Centrifugal Pumps," *University of Strathclyde Internal Report*, 2012.
- [65] J. Tu, G. H. Yeoh, and C. Liu, *Computational Fluid Dynamics - A Practical Approach*, First Edit. Elsevier Inc, 2008.
- [66] ANSYS Inc, *ANSYS CFD-Post User's Guide (R15.0)*. 2013, p. 297.
- [67] J. F. Gulich, *Centrifugal Pumps*. Springer, 2010.
- [68] M. Morgut and E. Nobile, "Numerical Predictions of Cavitating Flow around Model Scale Propellers by CFD and Advanced Model Calibration," *Int. J. Rotating Mach.*, pp. 1–11, 2012.
- [69] A. Ducoin, B. Huang, and Y. L. Young, "Numerical Modeling of Unsteady Cavitating Flows around a Stationary Hydrofoil," *Int. J. Rotating Mach.*, pp. 1–17, 2012.
- [70] ANSYS Inc, *ANSYS CFX-Solver Theory Guide (R15.0)*. 2013, p. 202.
- [71] ANSYS Inc, *ANSYS CFX Tutorials*. 2013, pp. 559–582.

APPENDIX A - Supplementary Data

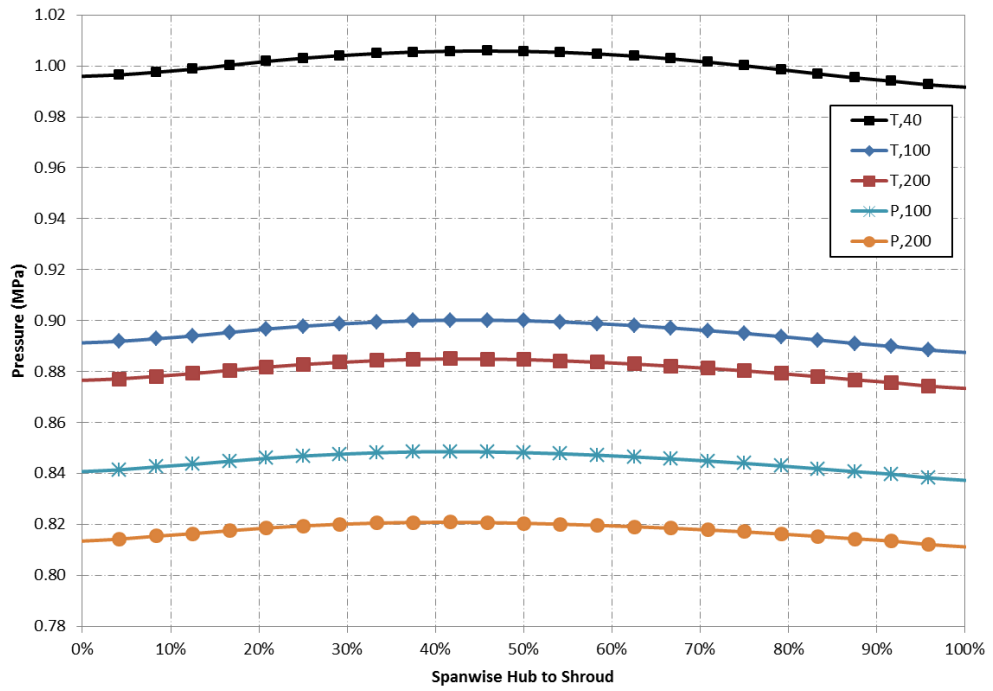
A1. Solver Capability (Study 1)



(a)

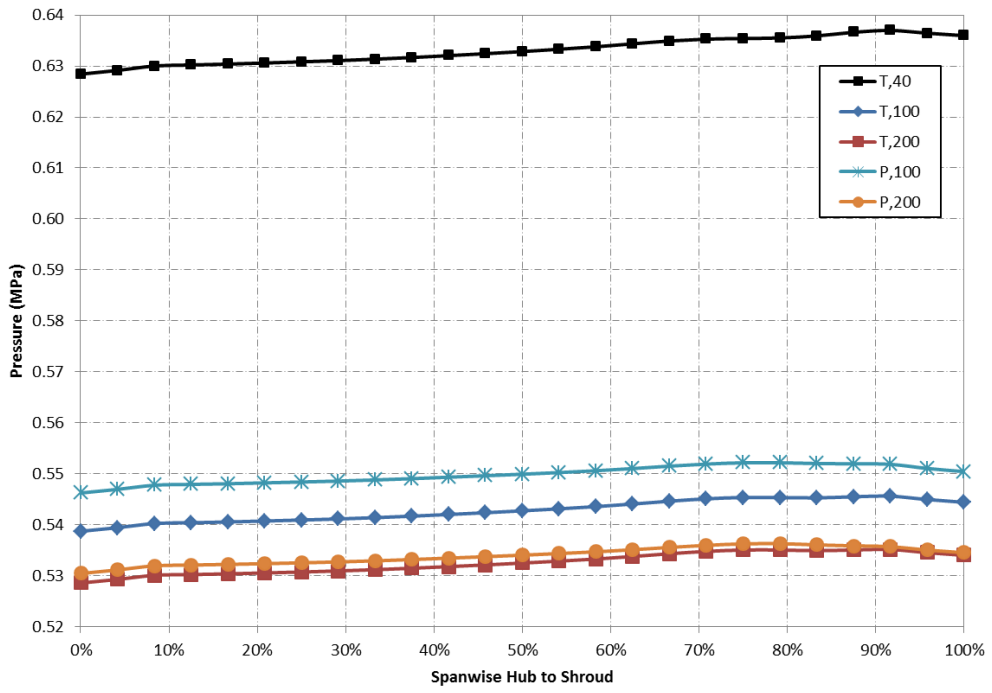


(b)

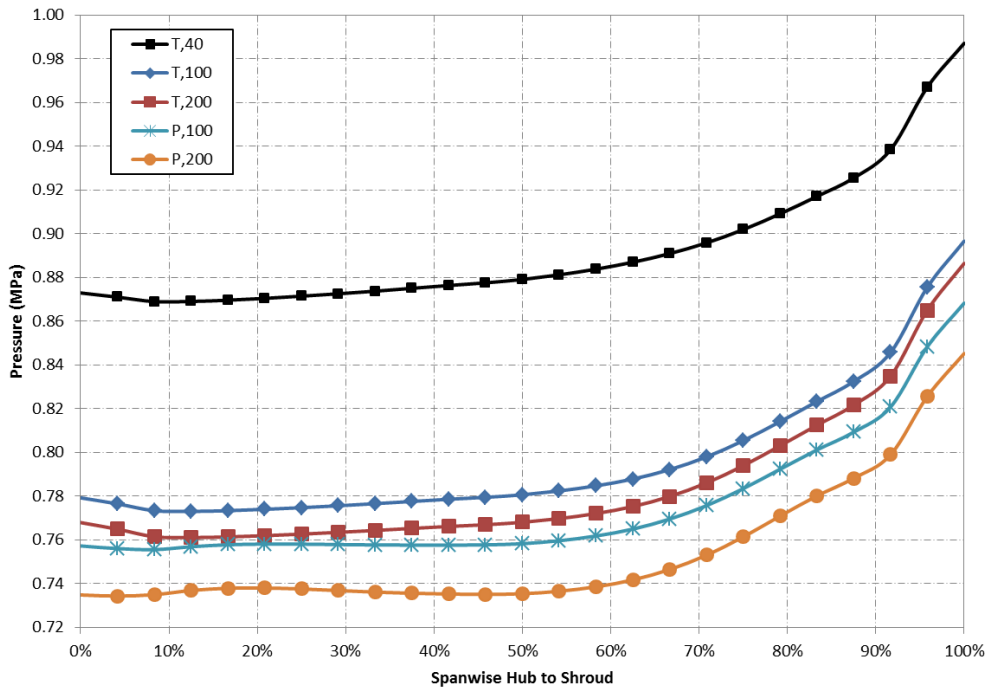


(c)

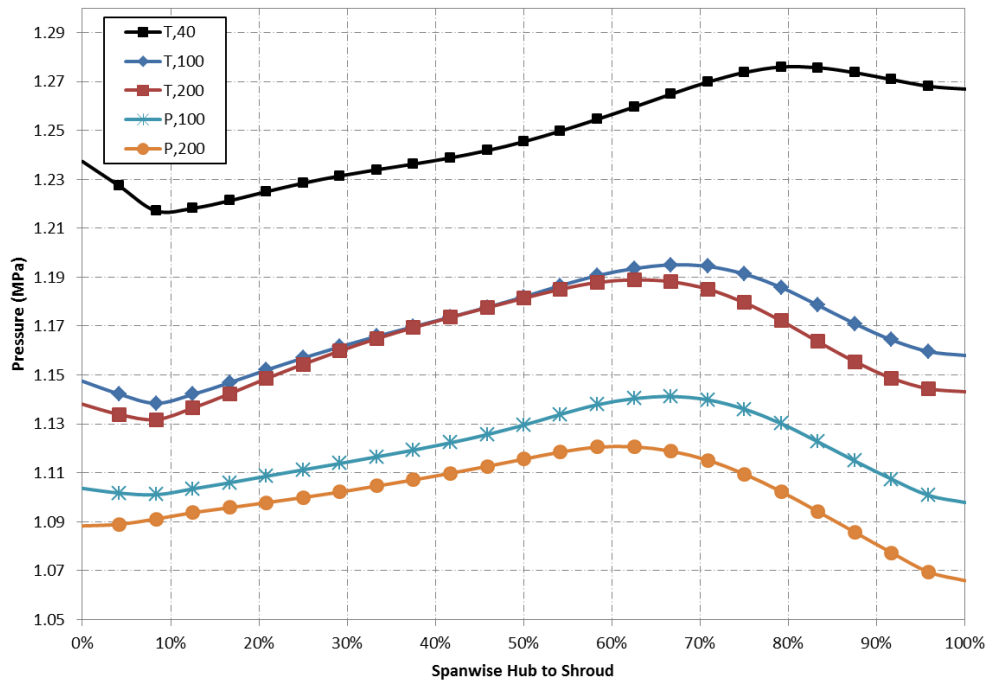
Figure A1 - Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 1)



(a)

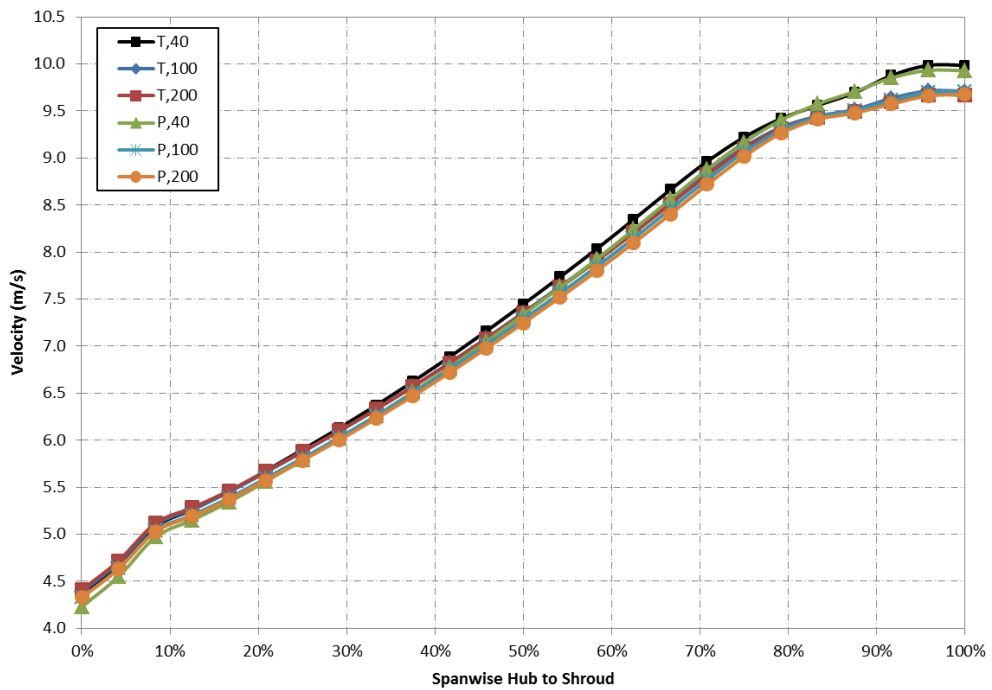


(b)

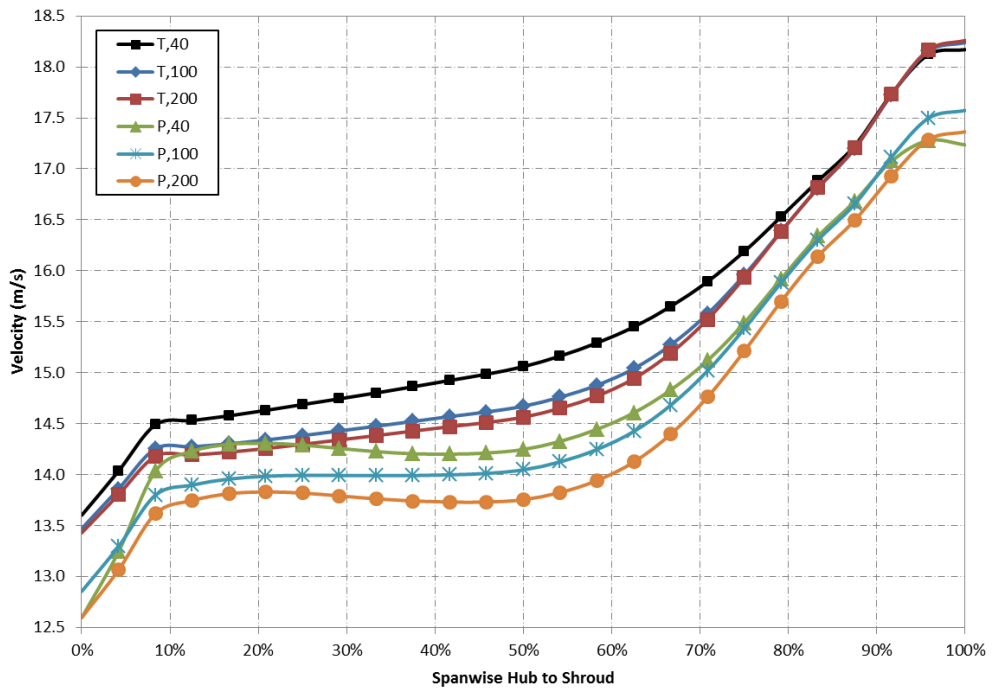


(c)

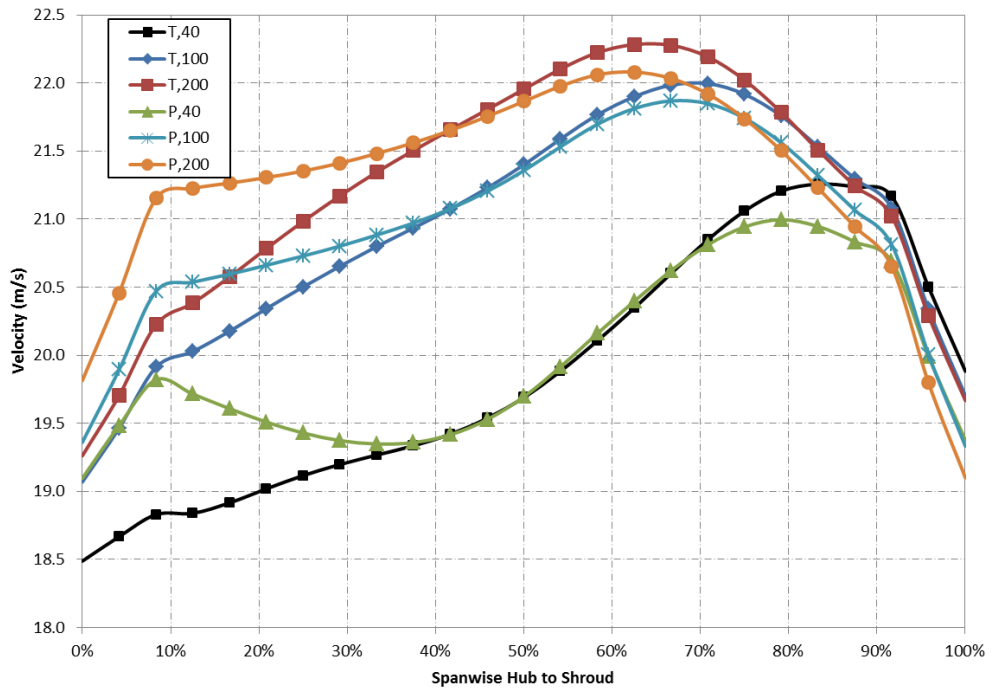
Figure A2 - Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 1)



(a)

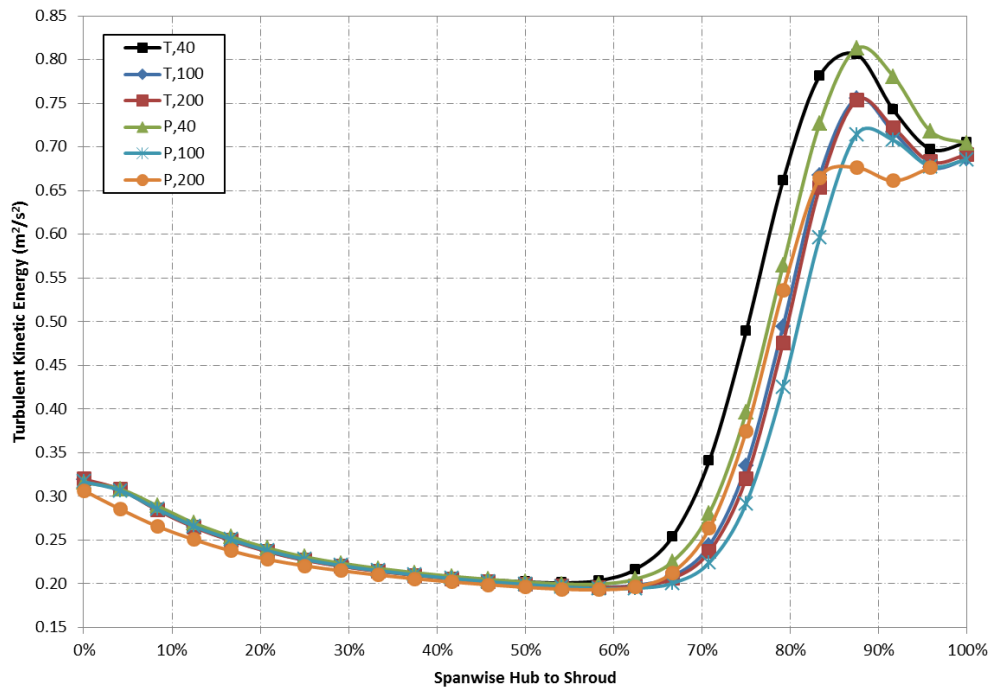


(b)

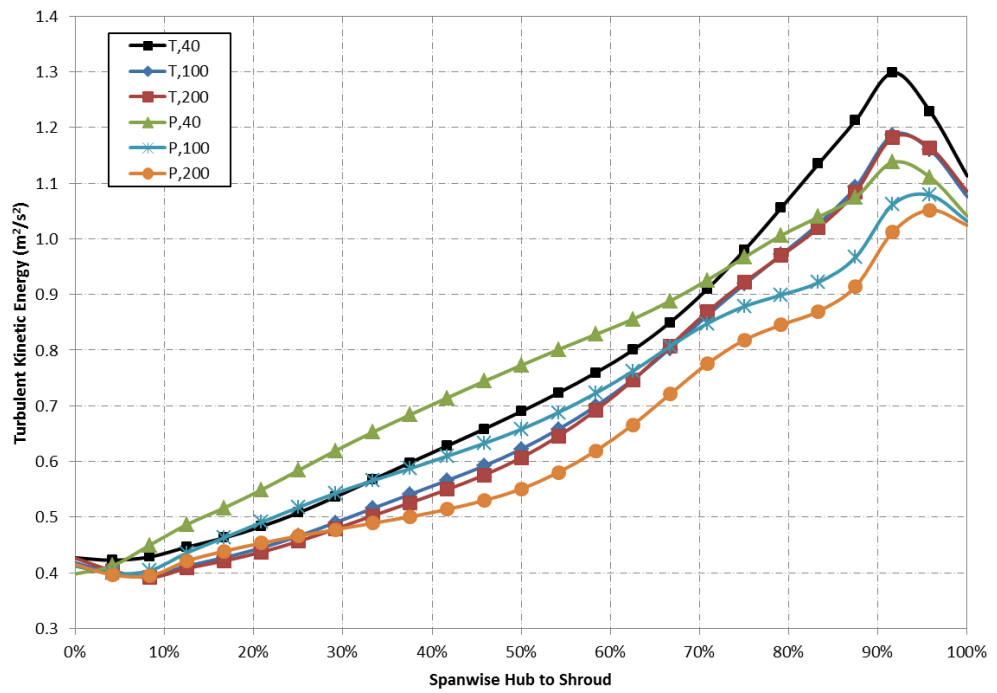


(c)

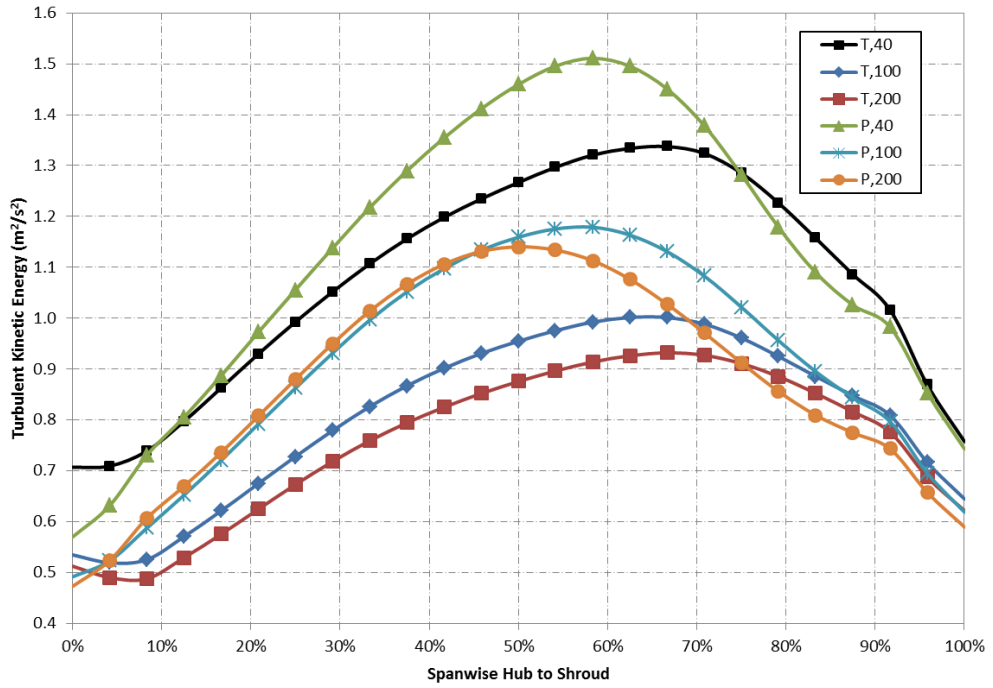
Figure A3 - Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 1)



(a)



(b)



(c)

Figure A4 - Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 1)

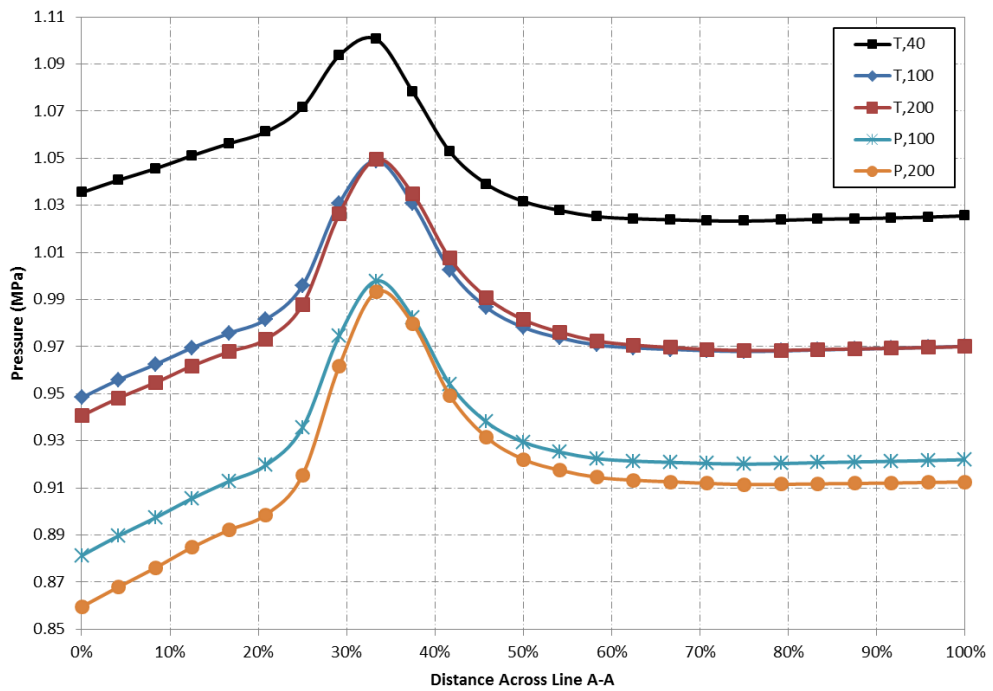


Figure A5 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 1)

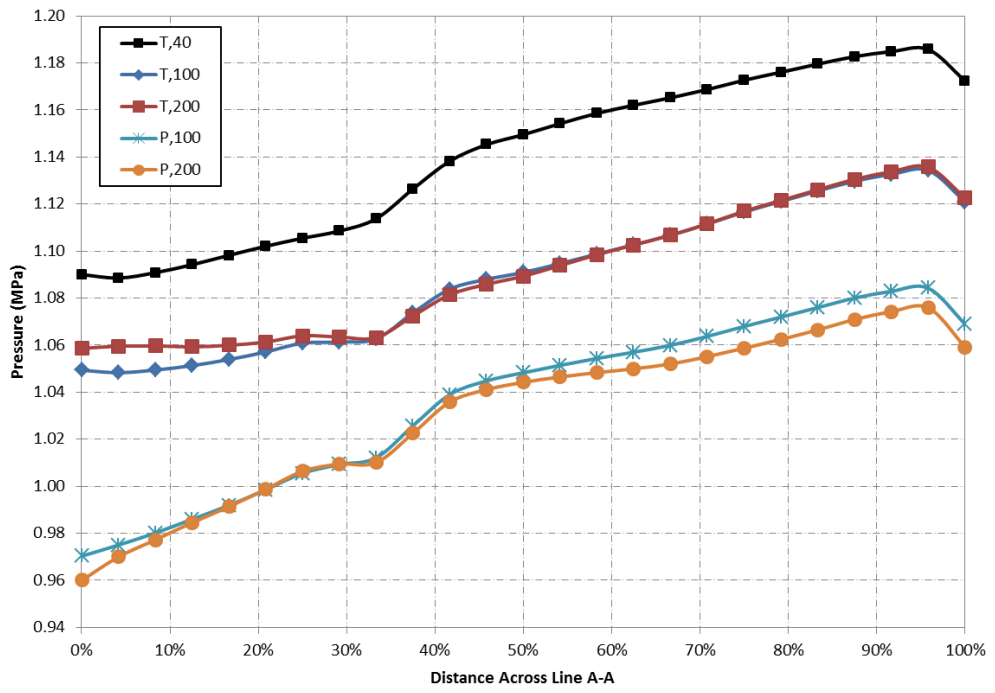


Figure A6 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 1)

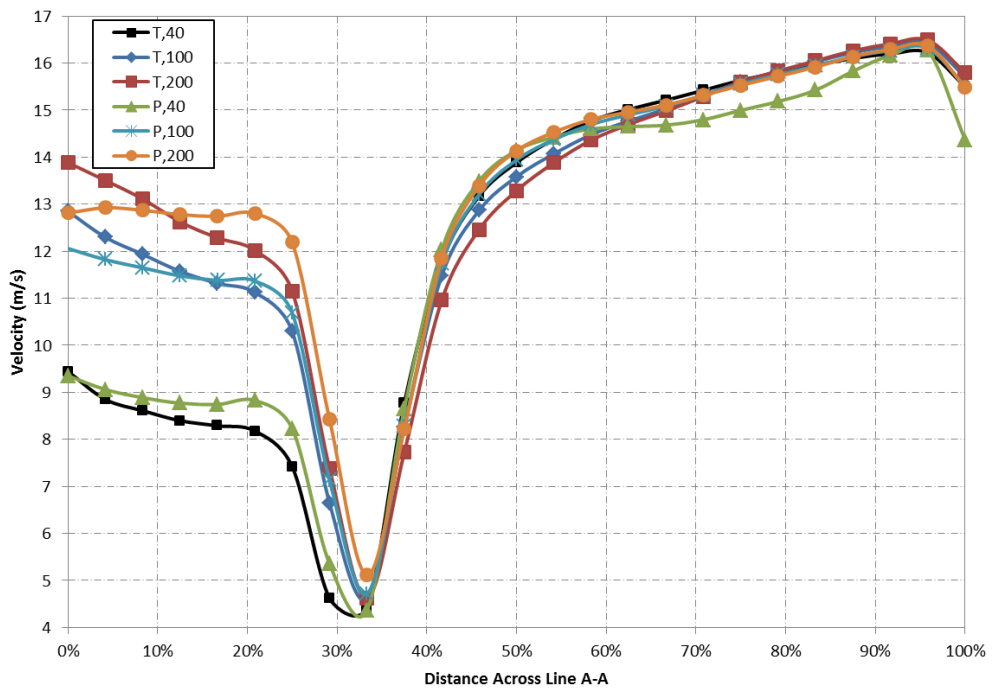


Figure A7 - Velocity across Line A-A (Figure 4.11) in Volute Casing (Study 1)

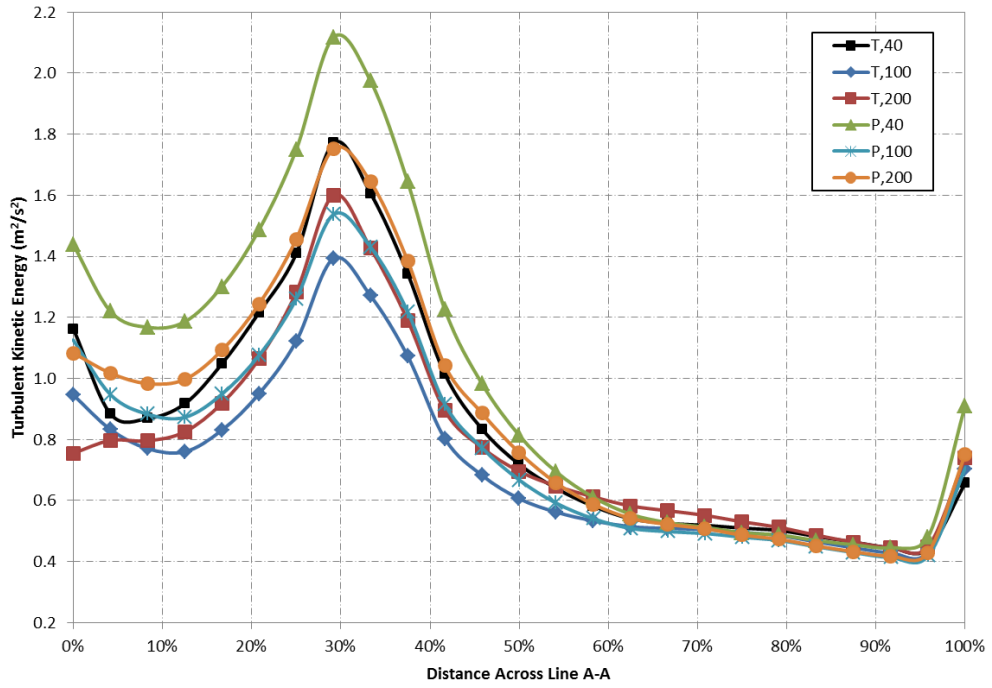


Figure A8 - Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Study 1)

A2. Number of Pressure Correcting Loops (Study 2)

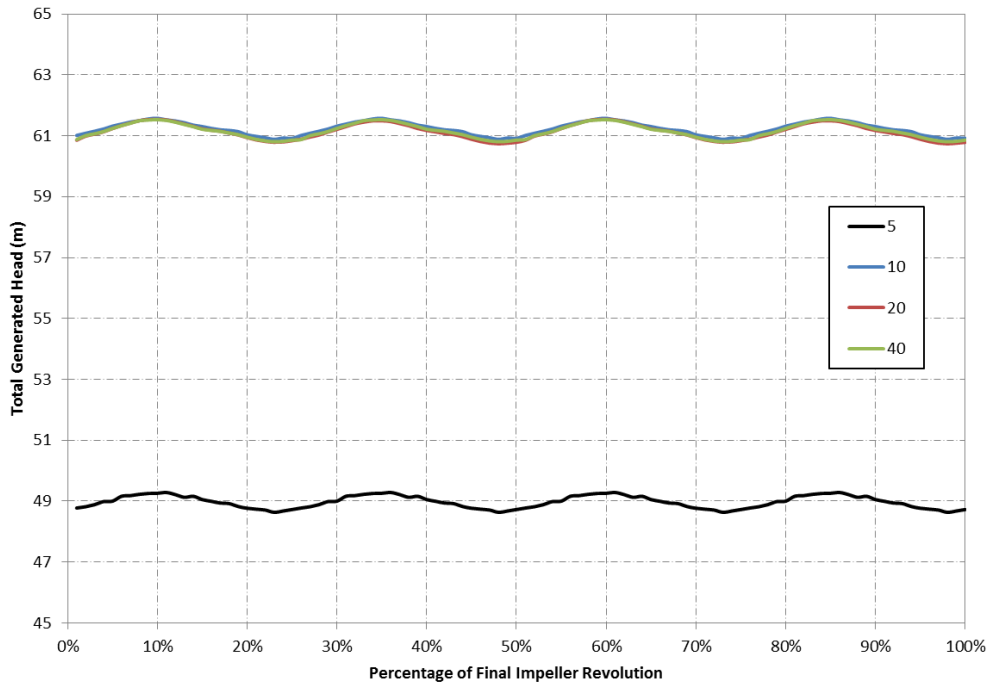


Figure A9 - Head Generated by Impeller during Final Revolution of Impeller (Study 2)

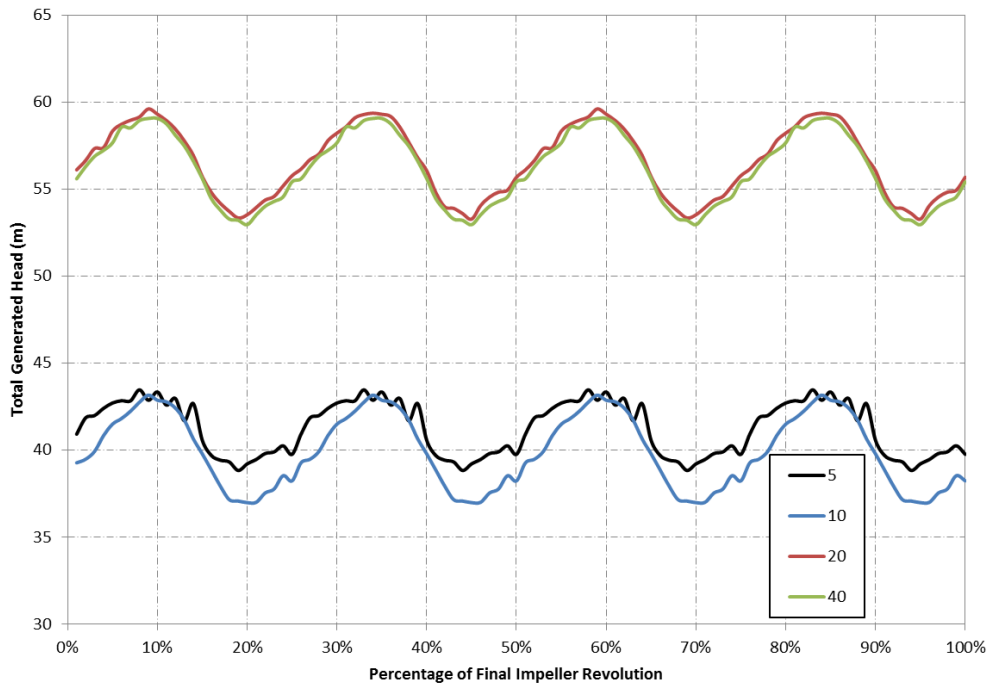


Figure A10 - Head Generated by Pump during Final Revolution of Impeller (Study 2)

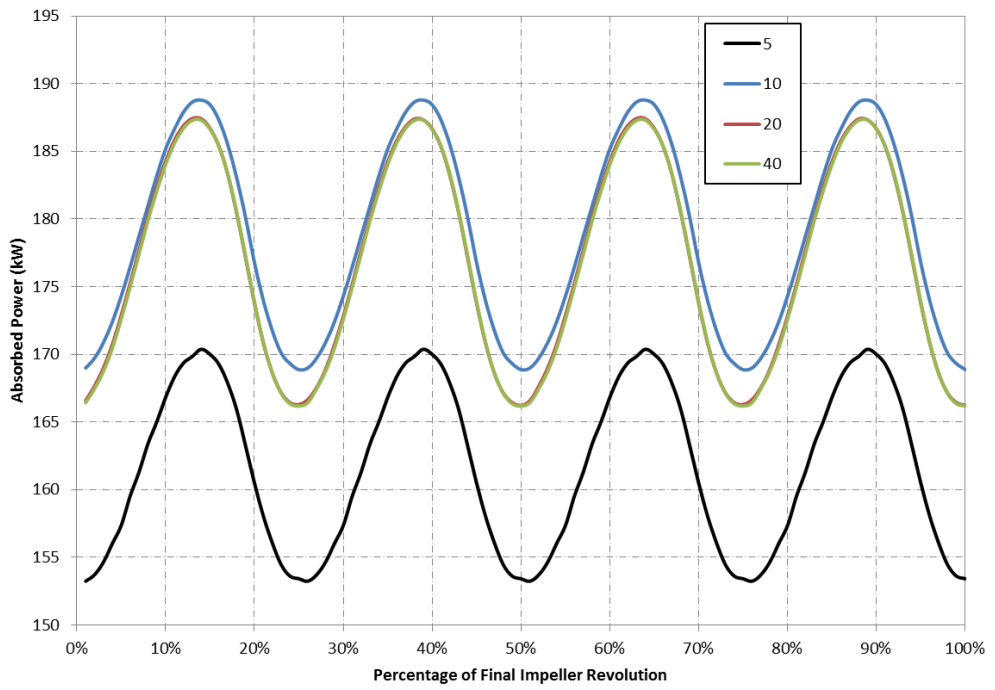


Figure A11 - Absorbed Power during Final Revolution of Impeller (Study 2)

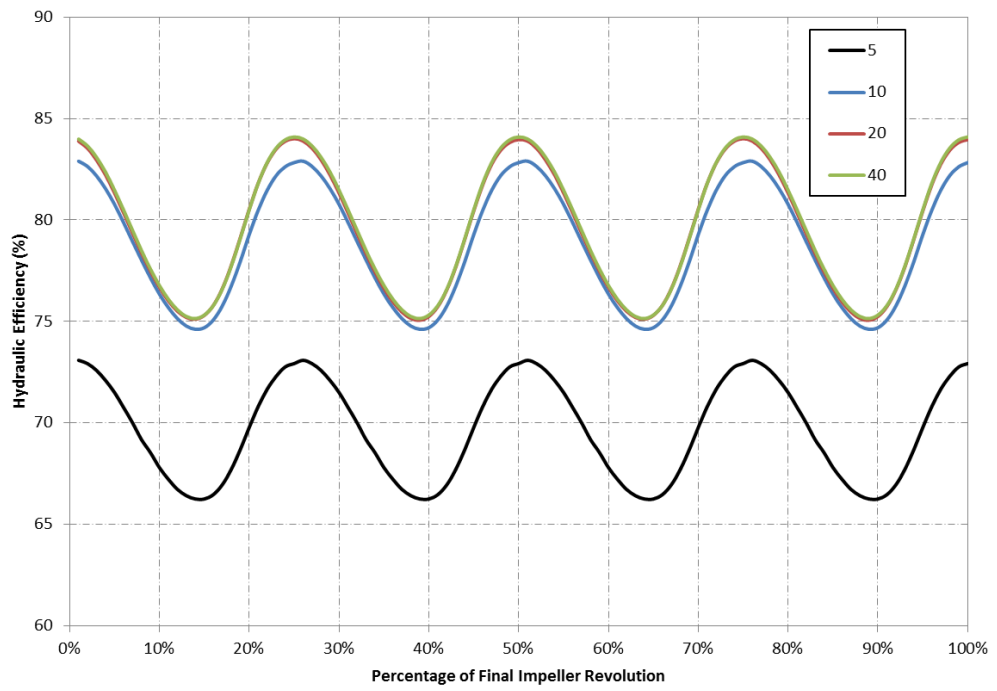


Figure A12 - Impeller Hydraulic Efficiency during Final Revolution of Impeller (Study 2)

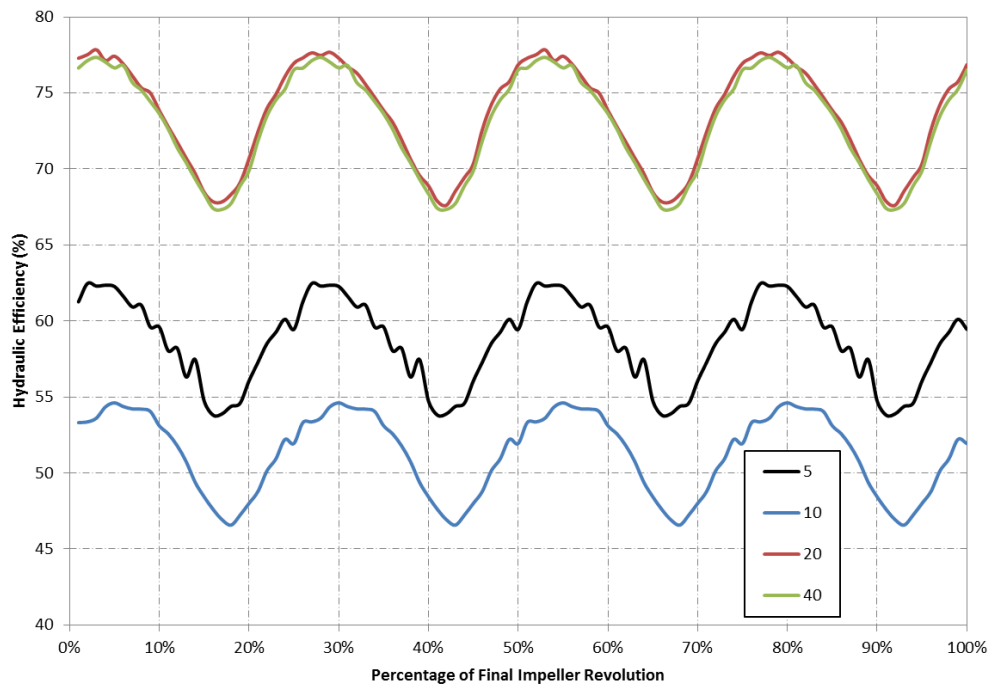
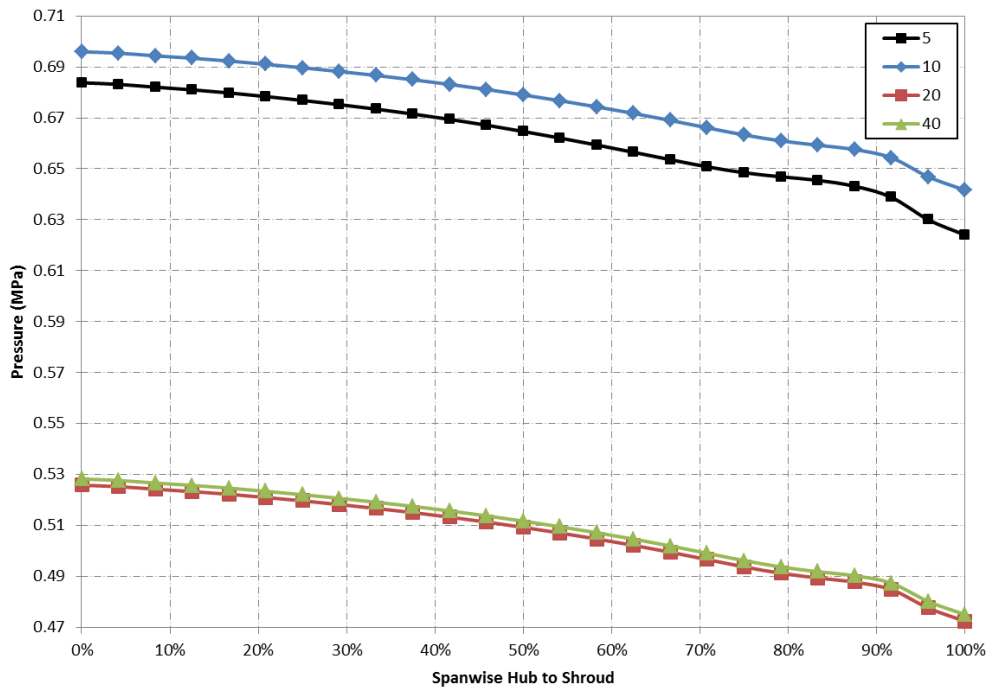
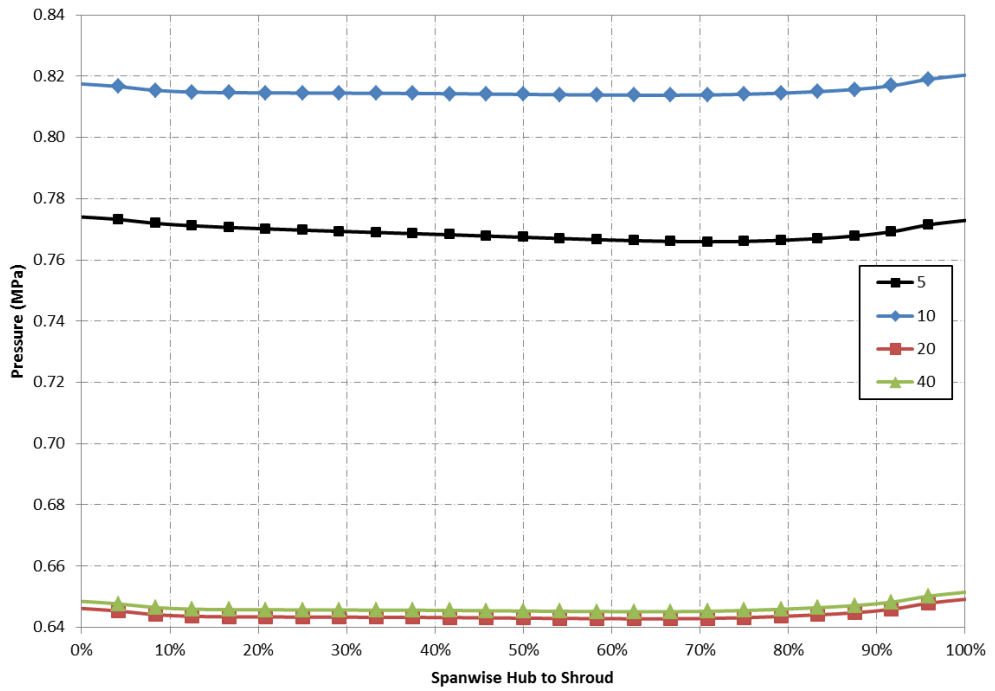


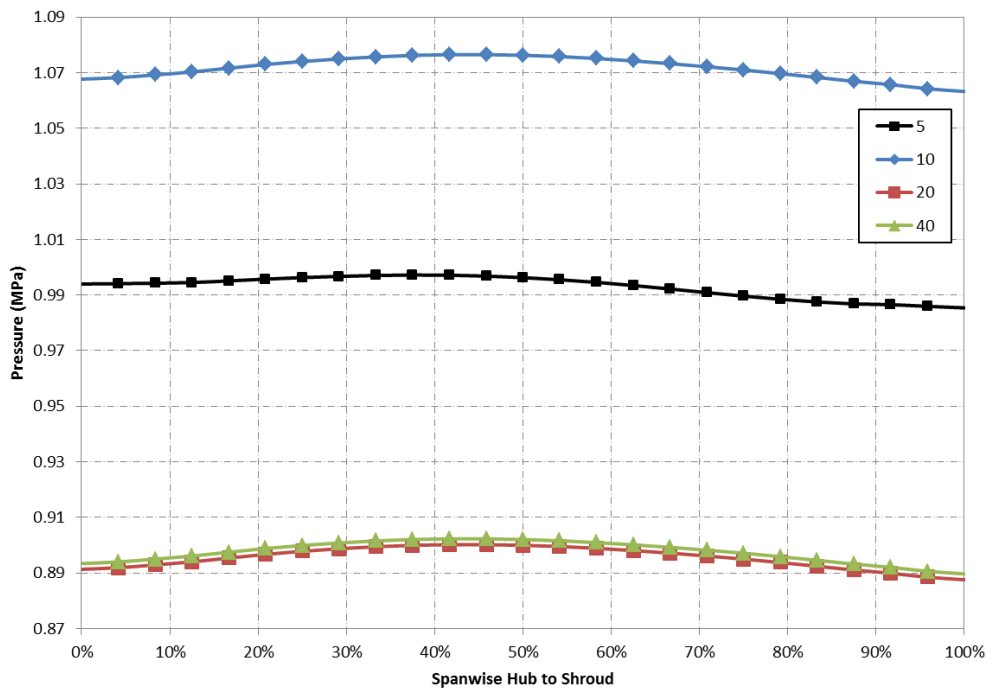
Figure A13 - Pump Hydraulic Efficiency during Final Revolution of Impeller (Study 2)



(a)

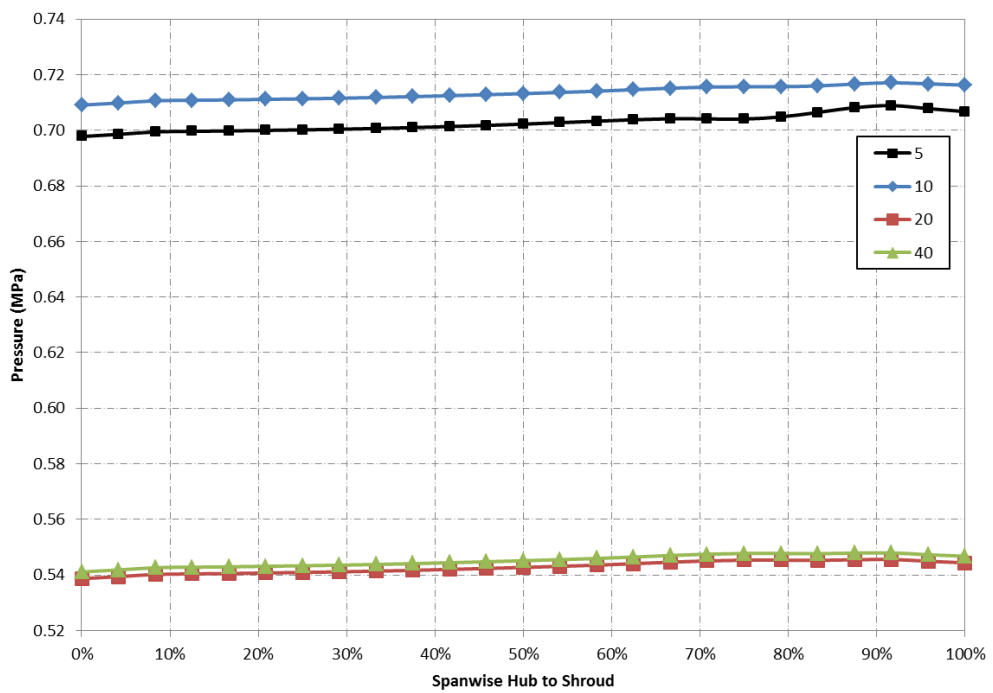


(b)

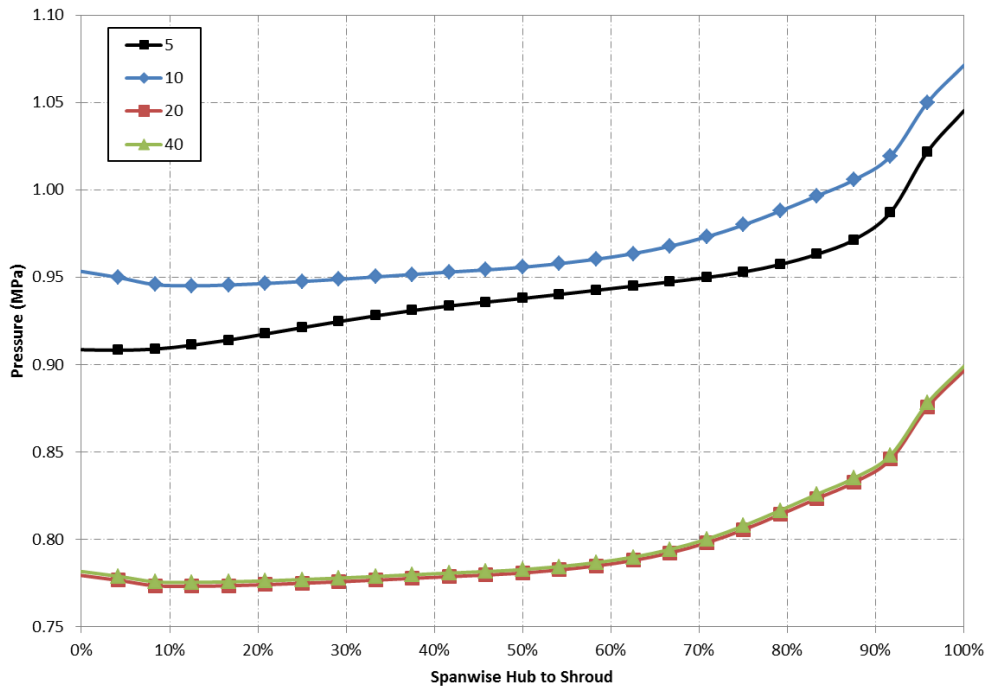


(c)

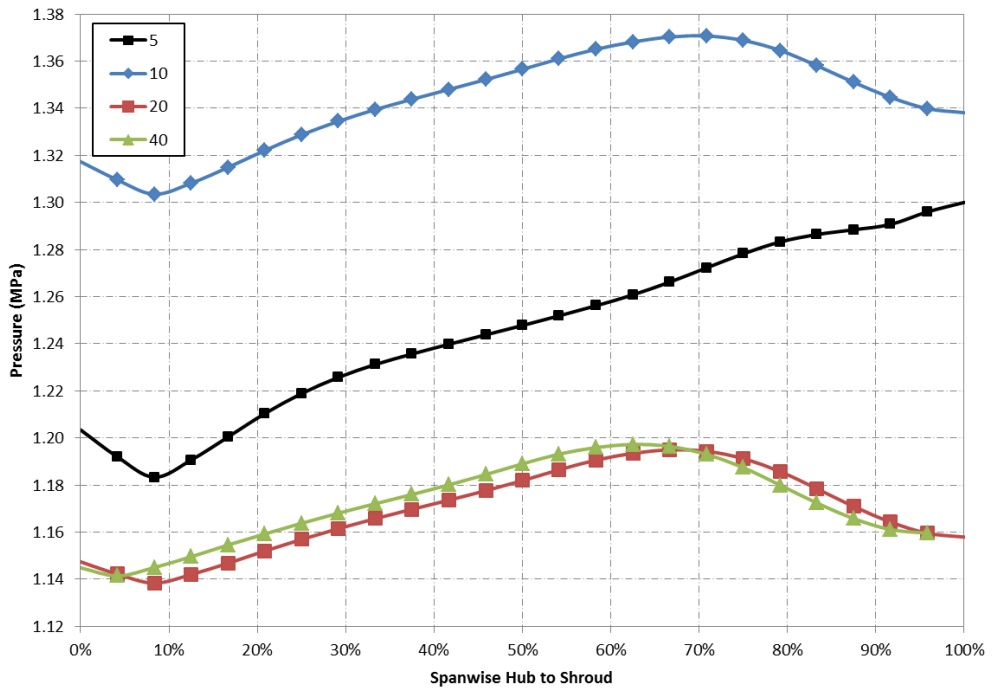
Figure A14 - Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 2)



(a)

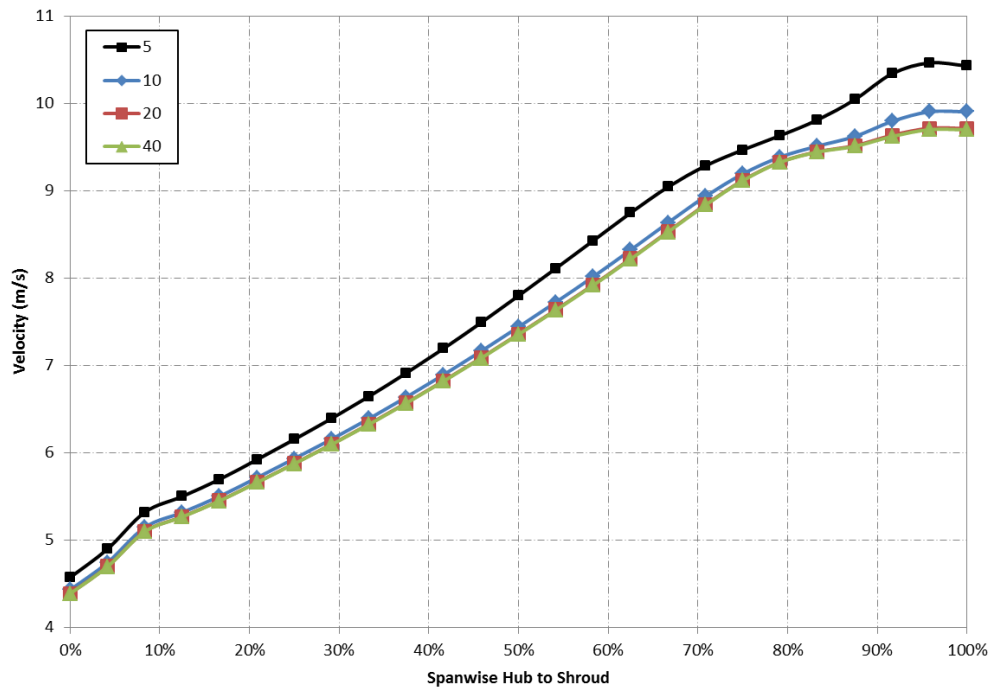


(b)

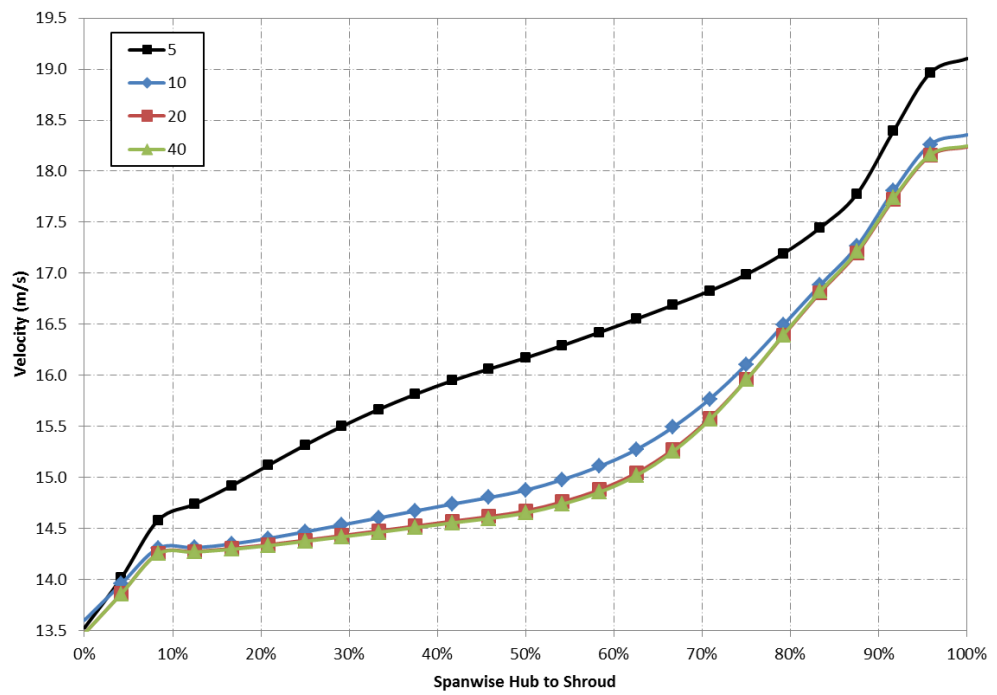


(c)

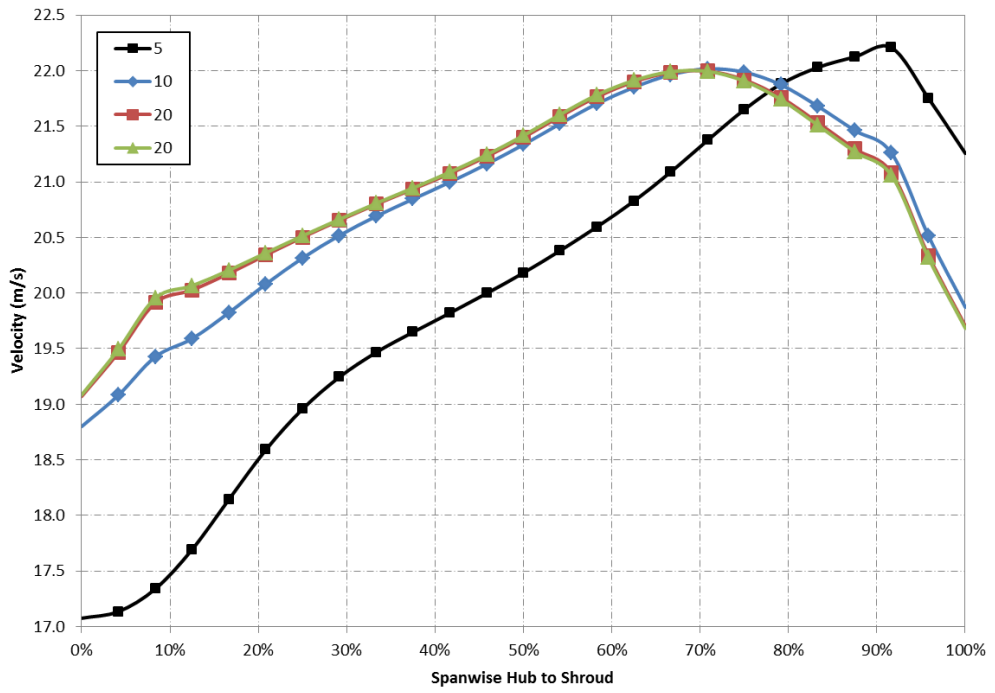
Figure A15 - Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 2)



(a)

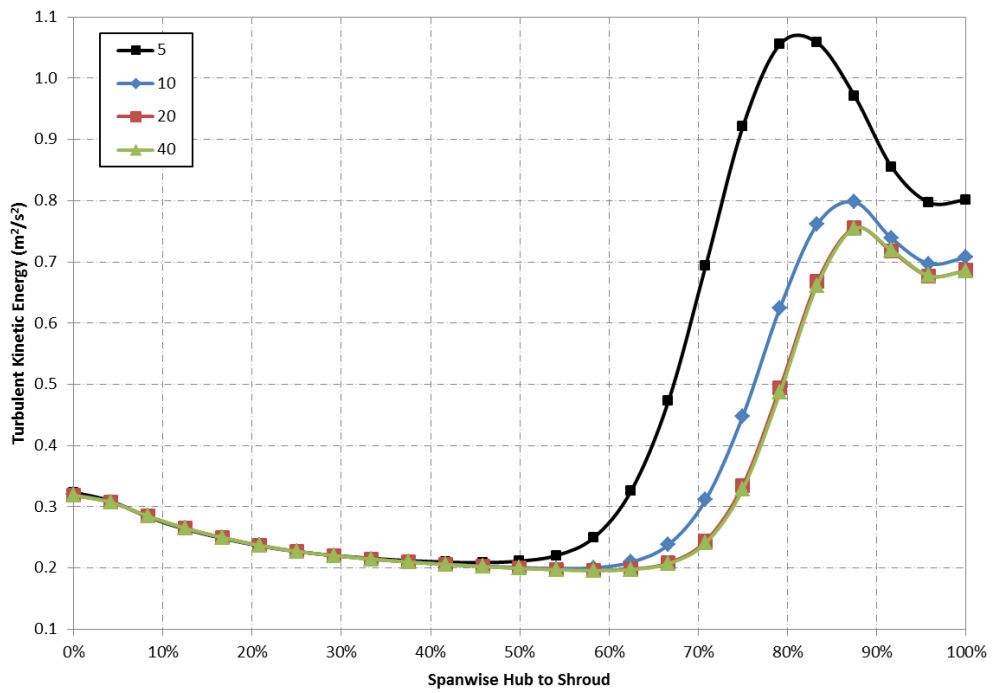


(b)

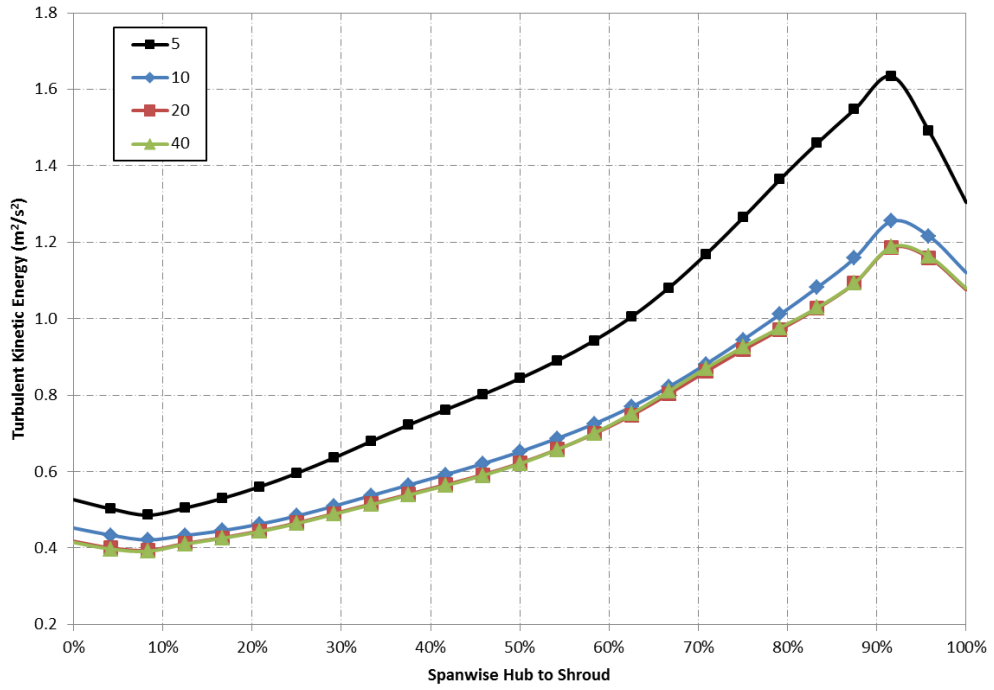


(c)

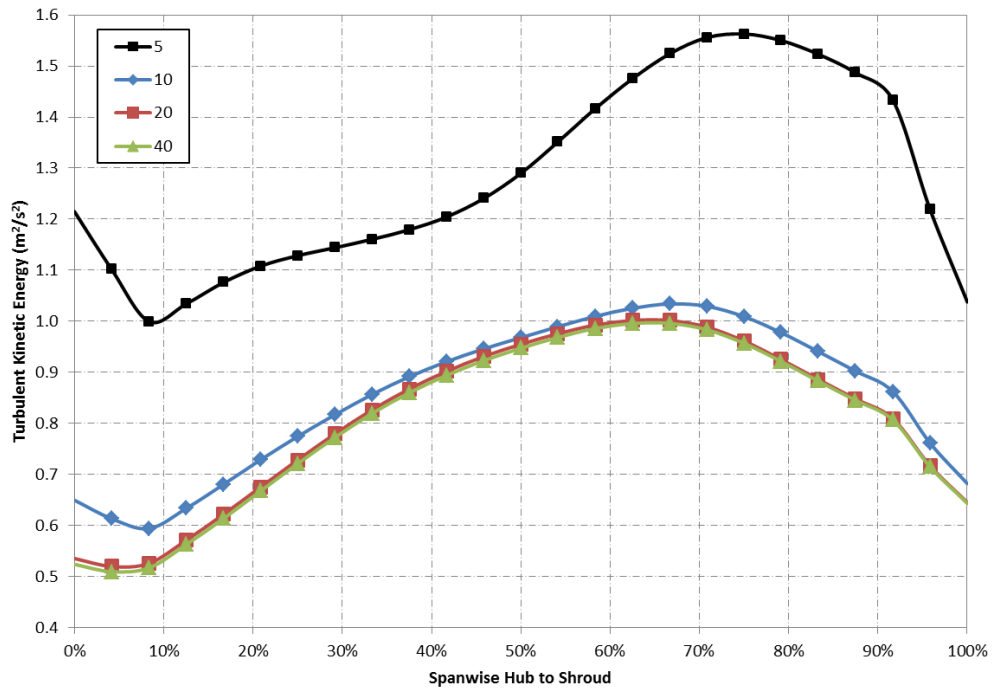
Figure A16 - Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 2)



(a)



(b)



(c)

Figure A17 - Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 2)

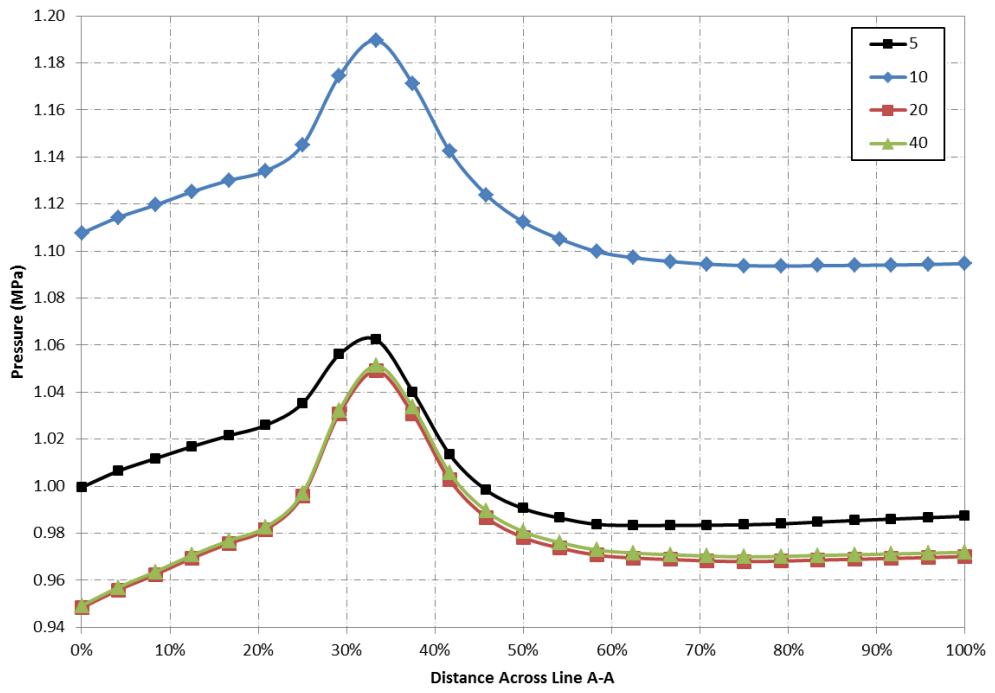


Figure A18 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 2)

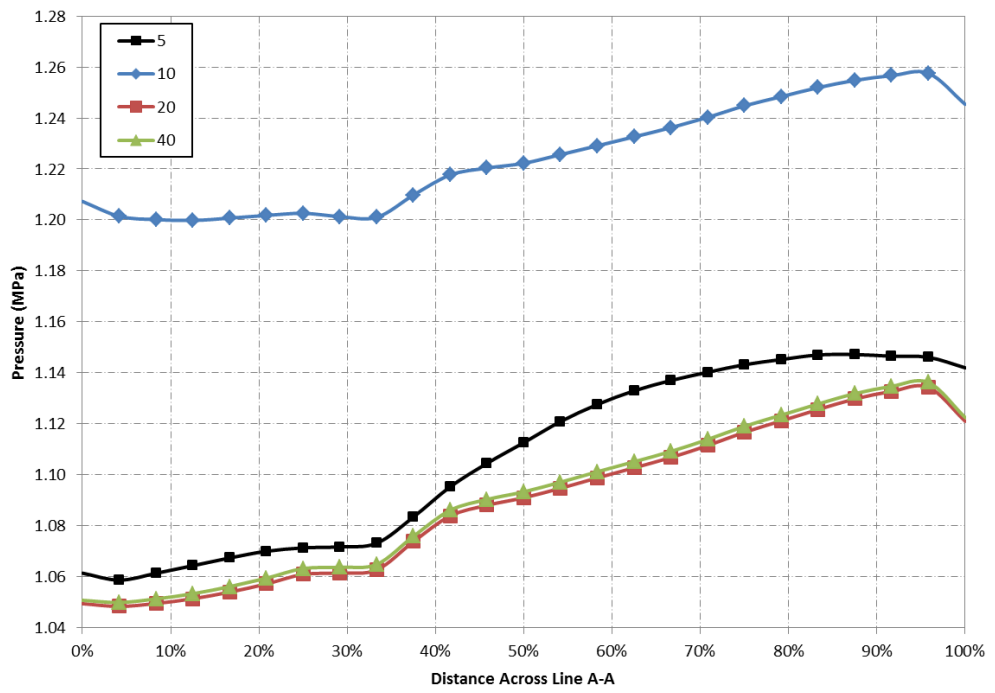


Figure A19 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 2)

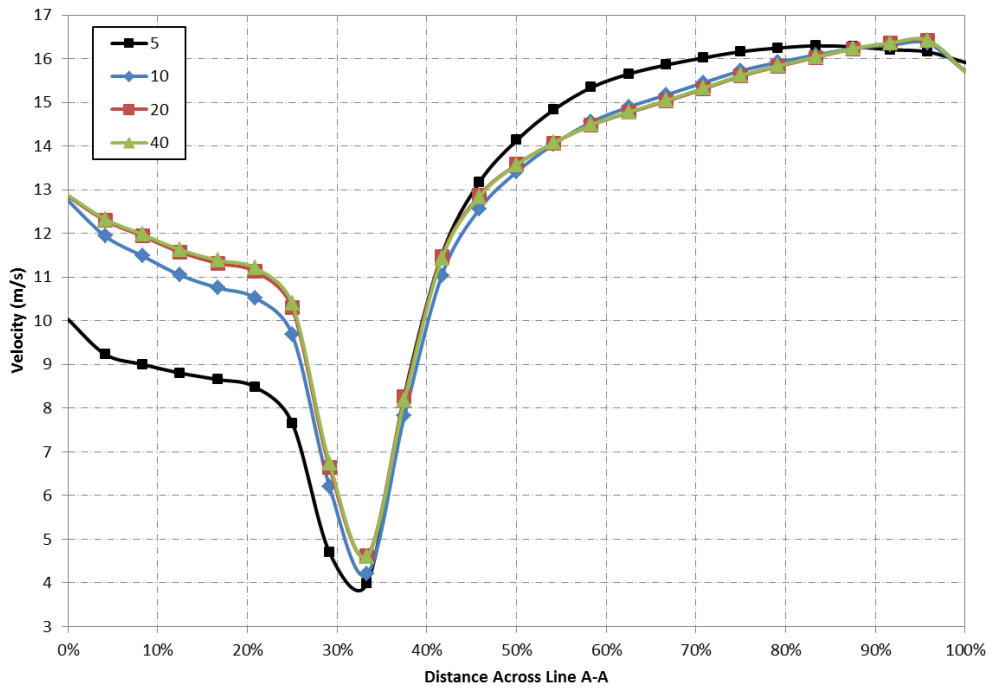


Figure A20 - Velocity across Line A-A (Figure 4.11) in Volute Casing (Study 2)

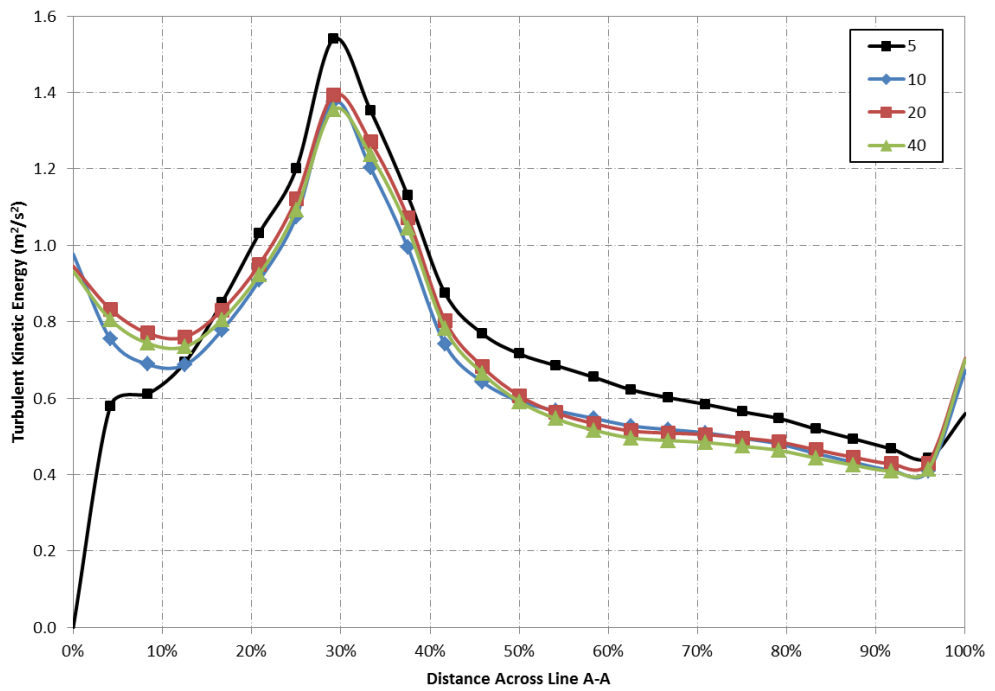


Figure A21 - Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Study 2)

A3. Time Discretisation Scheme (Study 3)

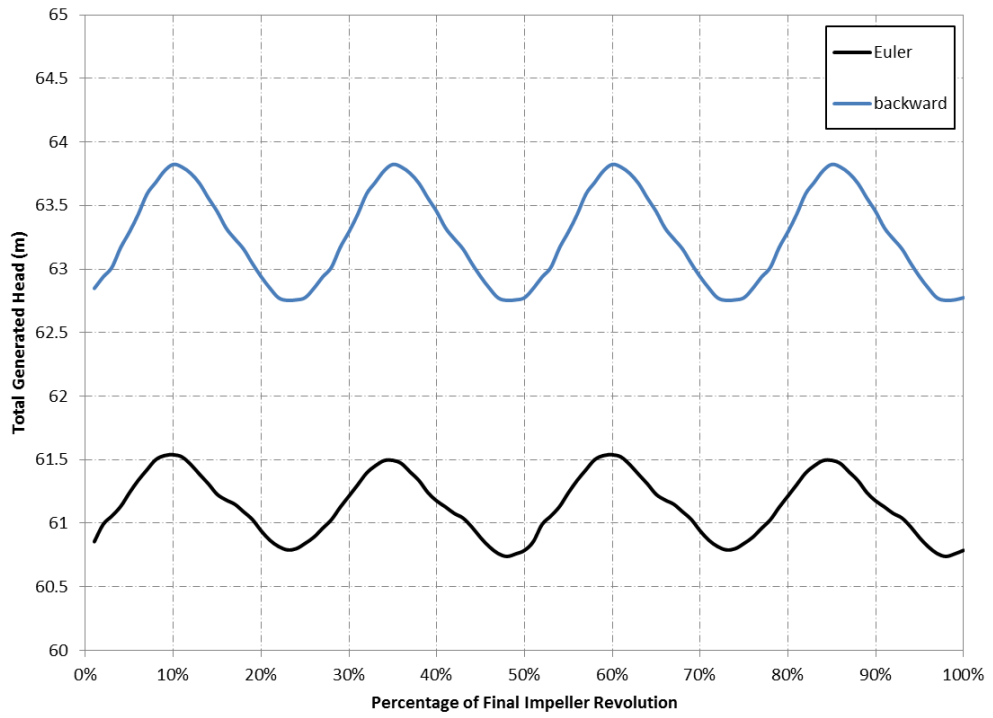


Figure A22 - Head Generated by Impeller during Final Revolution of Impeller (Study 3)

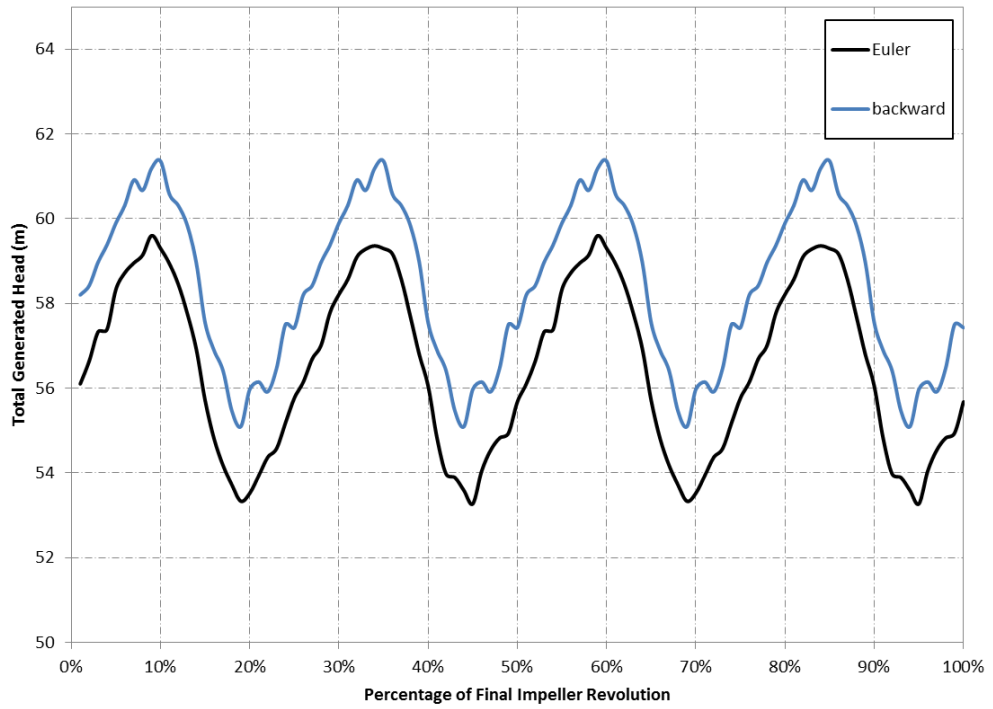


Figure A23 - Head Generated by Pump during Final Revolution of Impeller (Study 3)

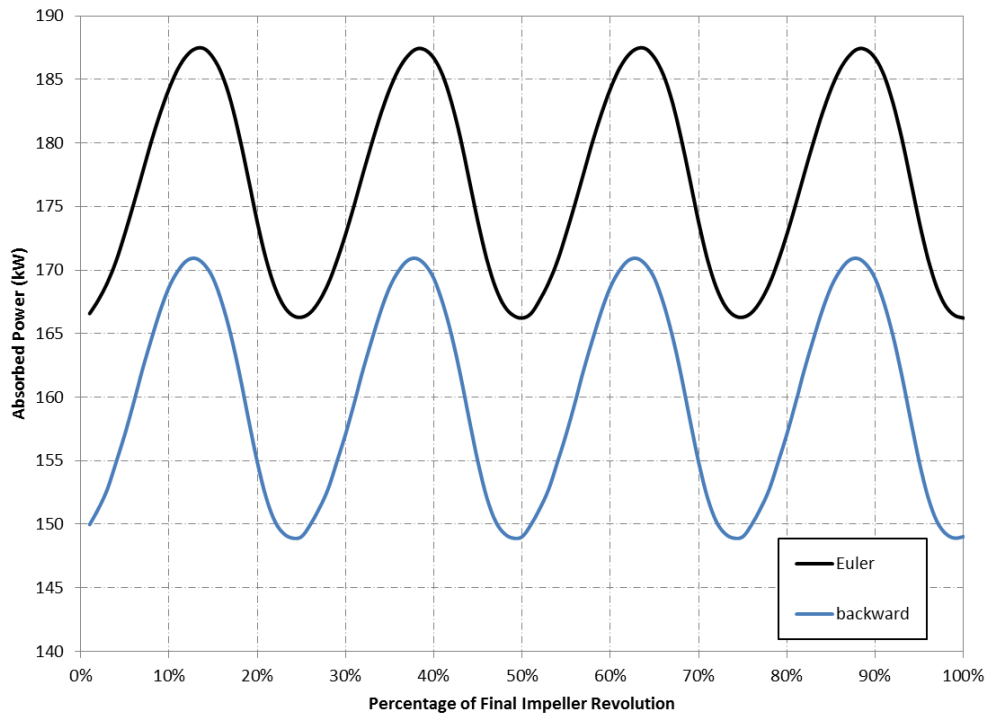


Figure A24 - Absorbed Power during Final Revolution of Impeller (Study 3)

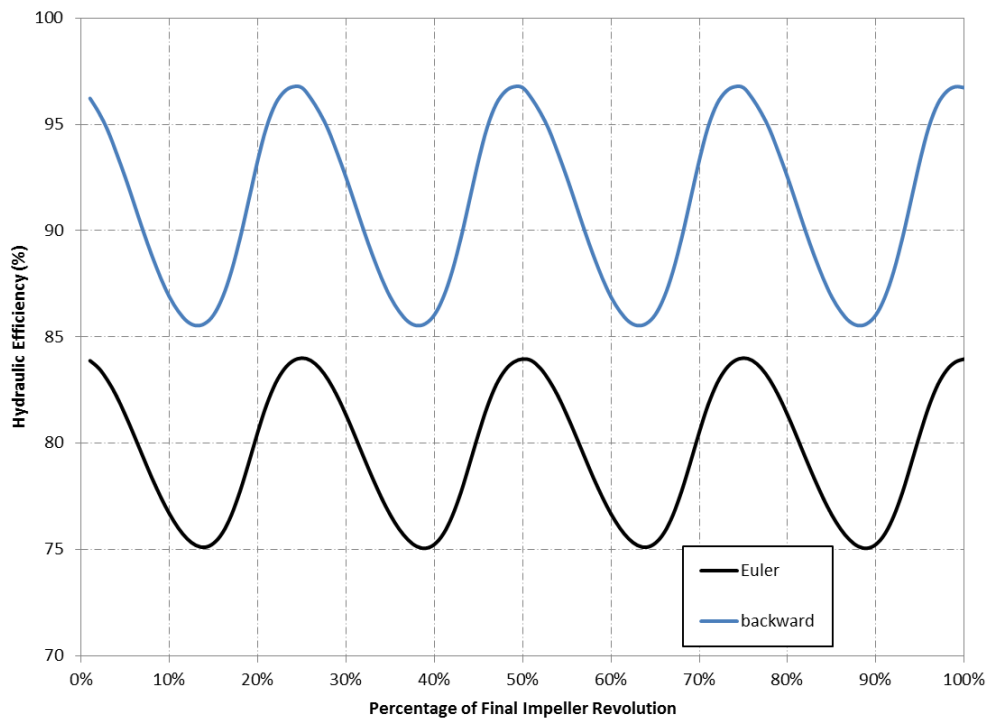


Figure A25 - Impeller Hydraulic Efficiency during Final Revolution of Impeller (Study 3)

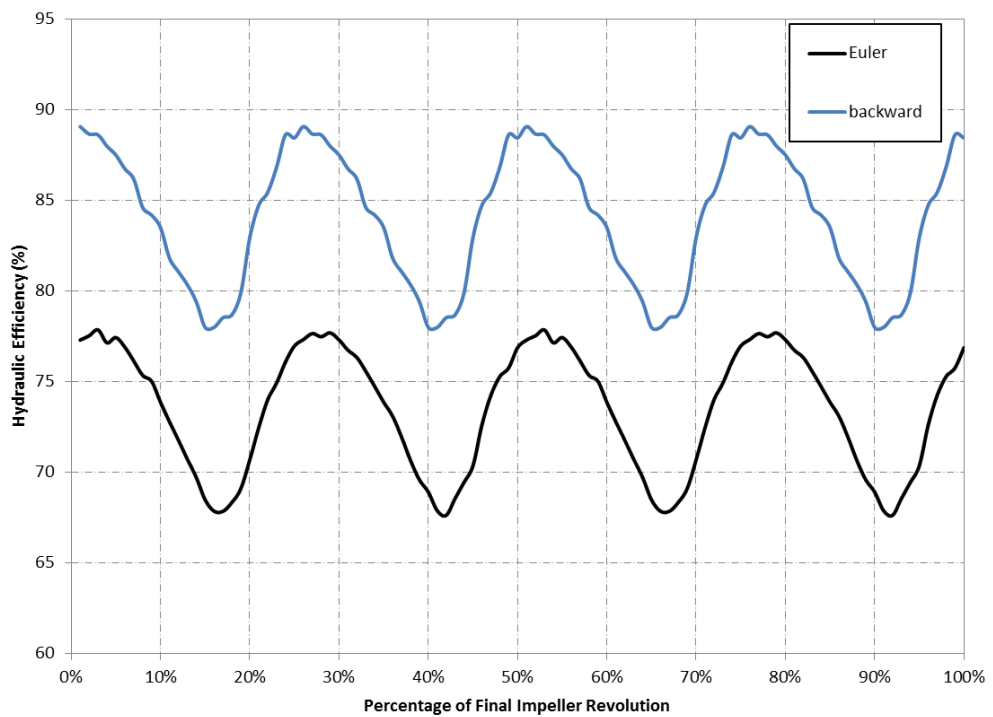
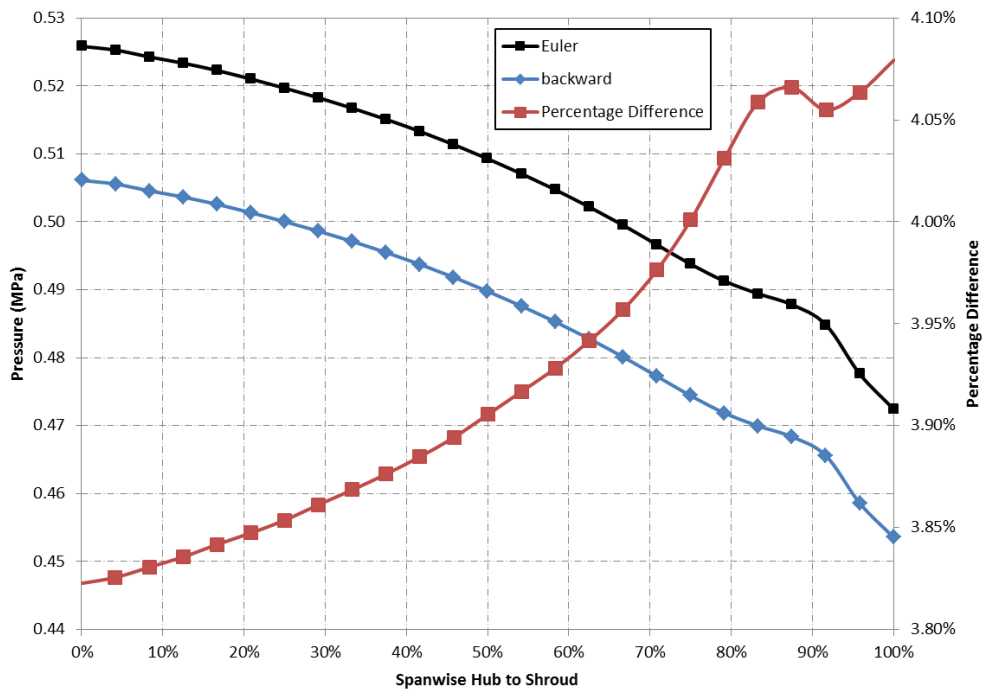
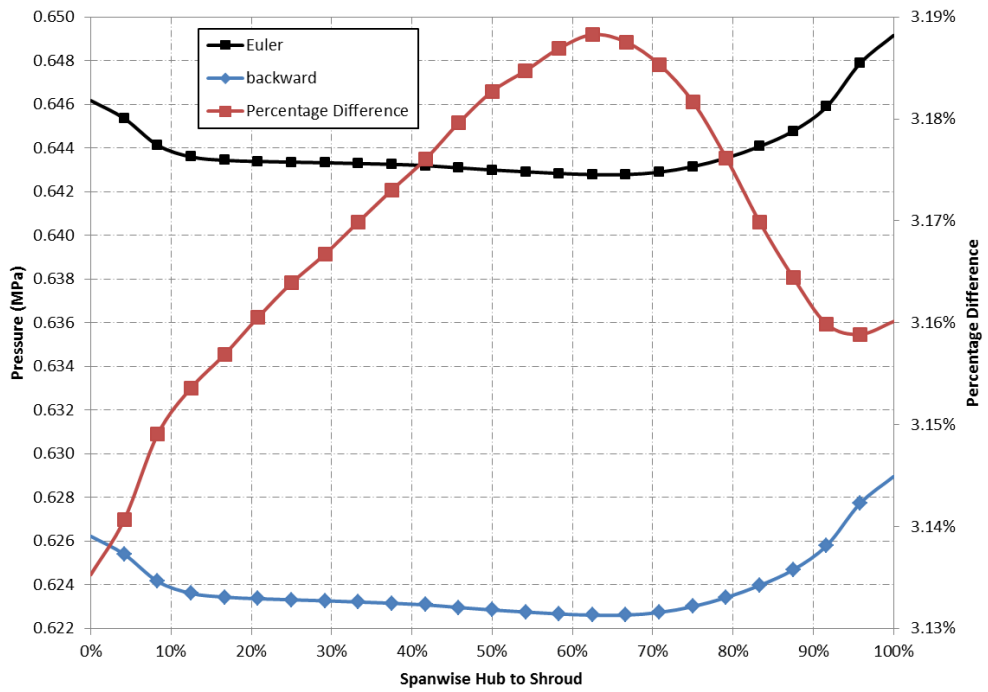


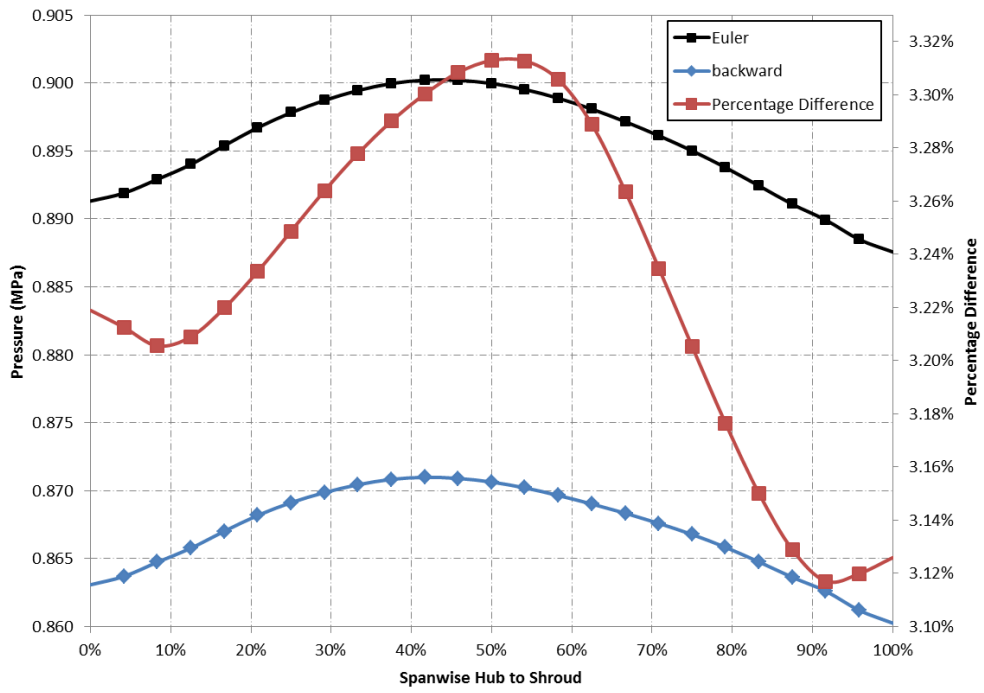
Figure A26 - Pump Hydraulic Efficiency during Final Revolution of Impeller (Study 3)



(a)

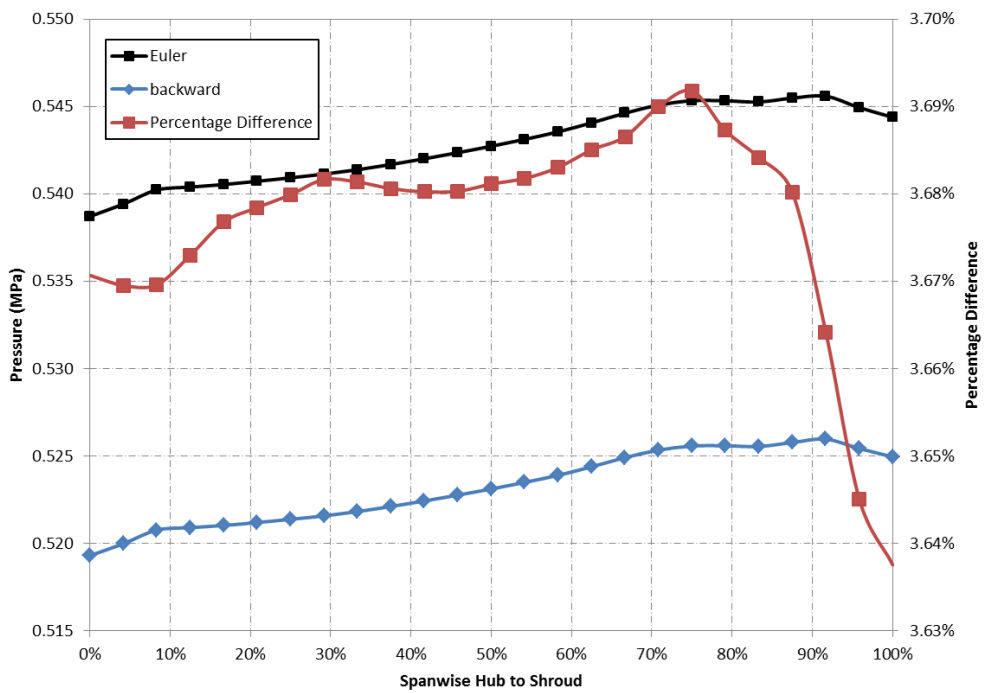


(b)

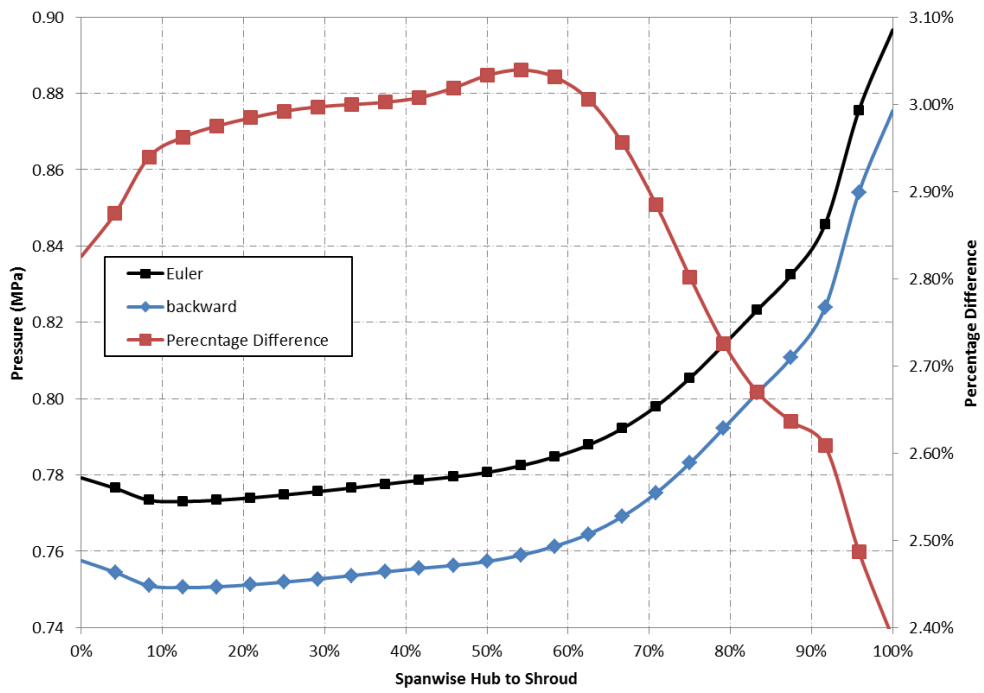


(c)

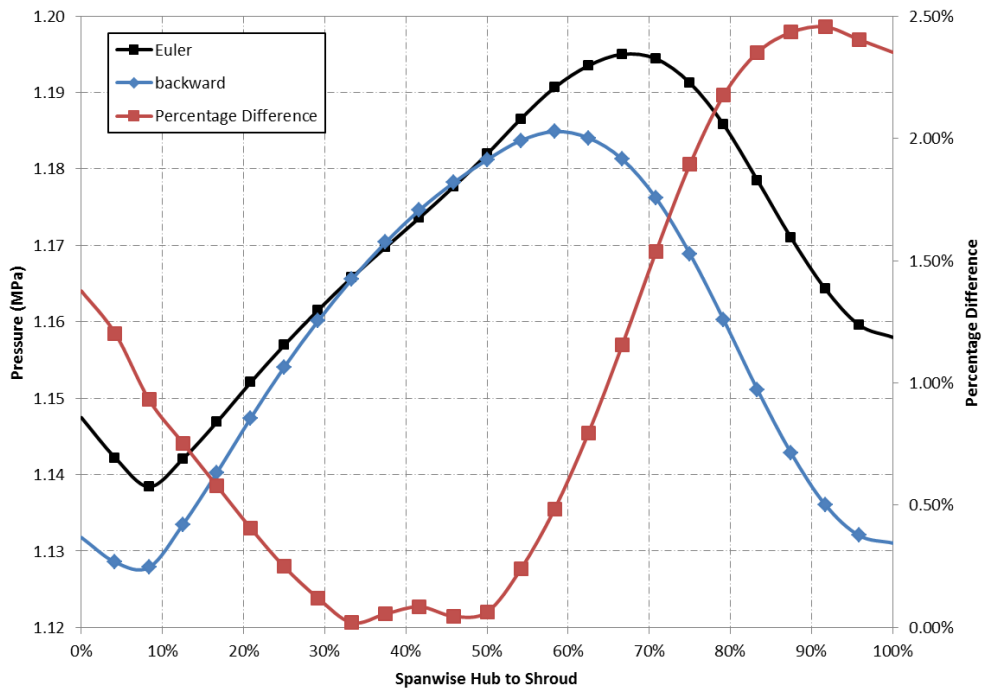
Figure A27 - Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 3)



(a)

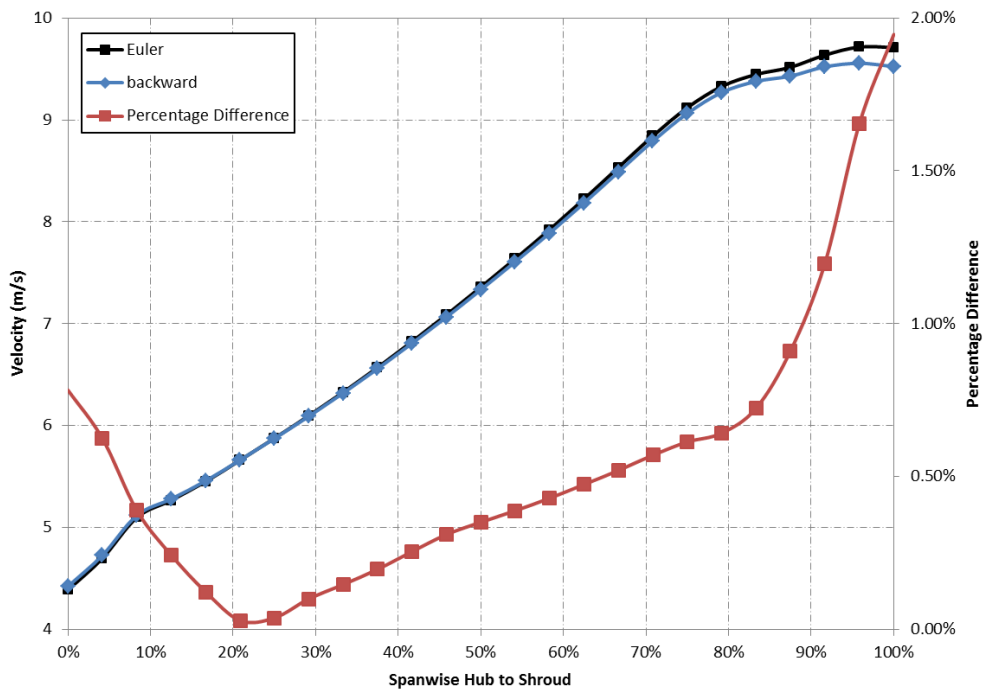


(b)

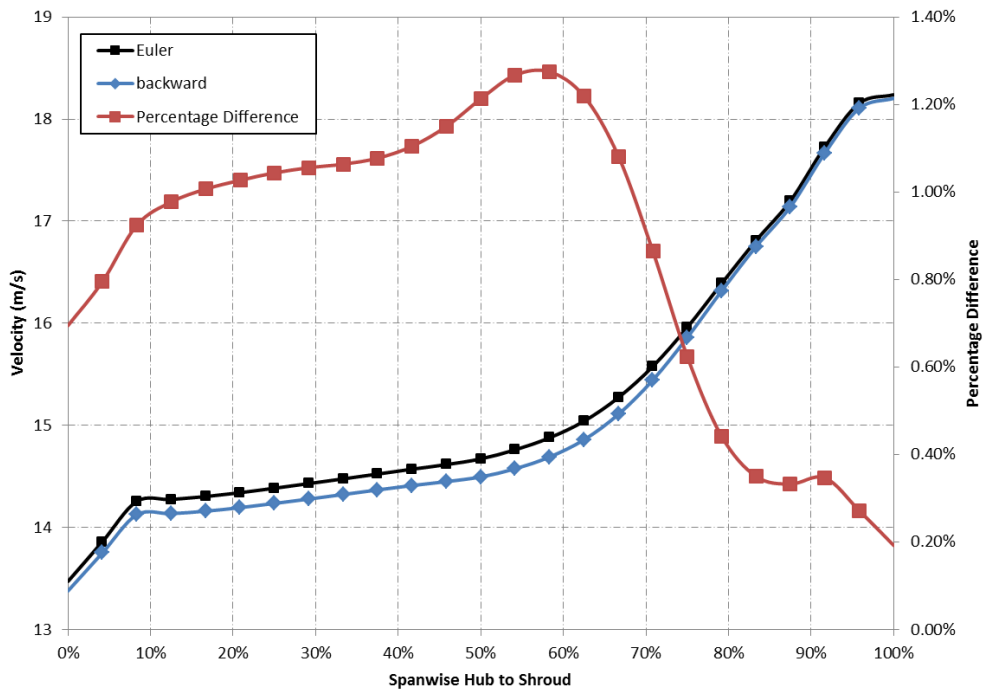


(c)

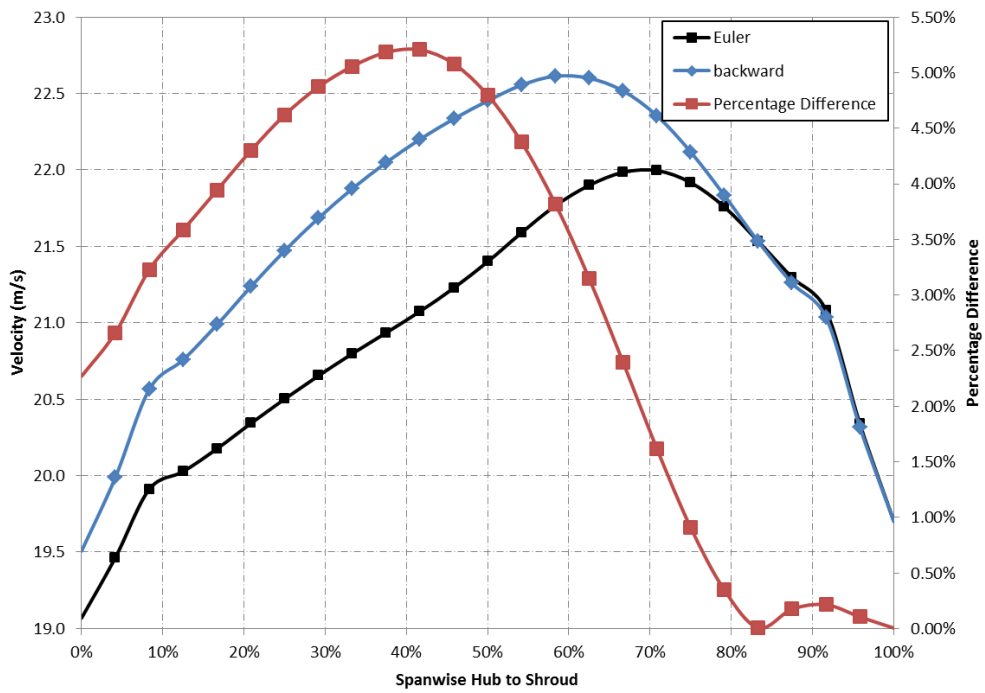
Figure A28 - Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 3)



(a)



(b)



(c)

Figure A29 - Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 3)

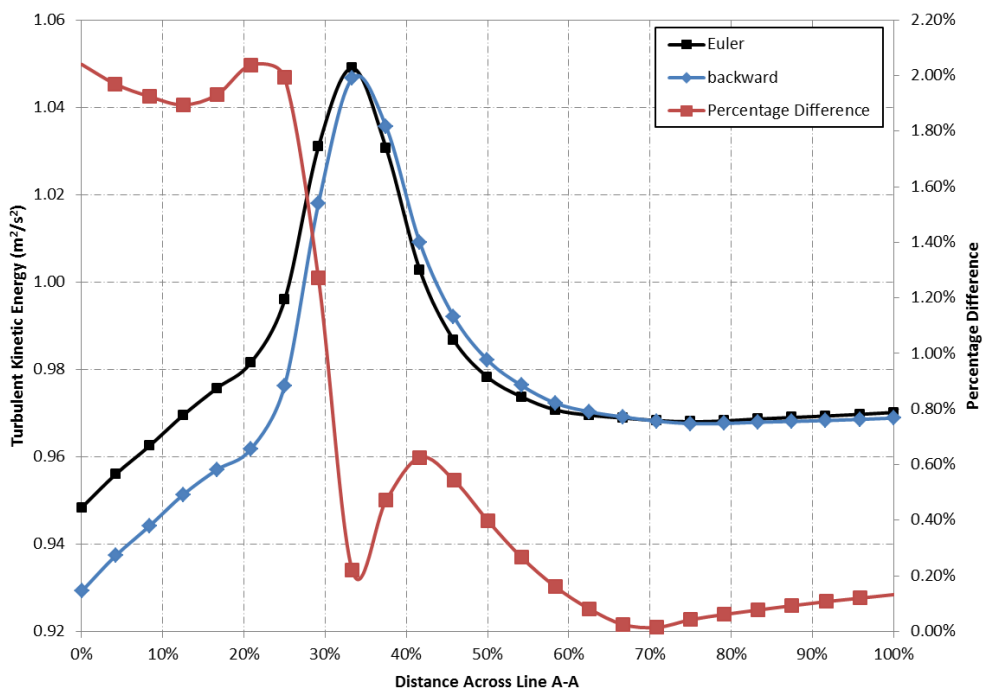


Figure A30 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 3)

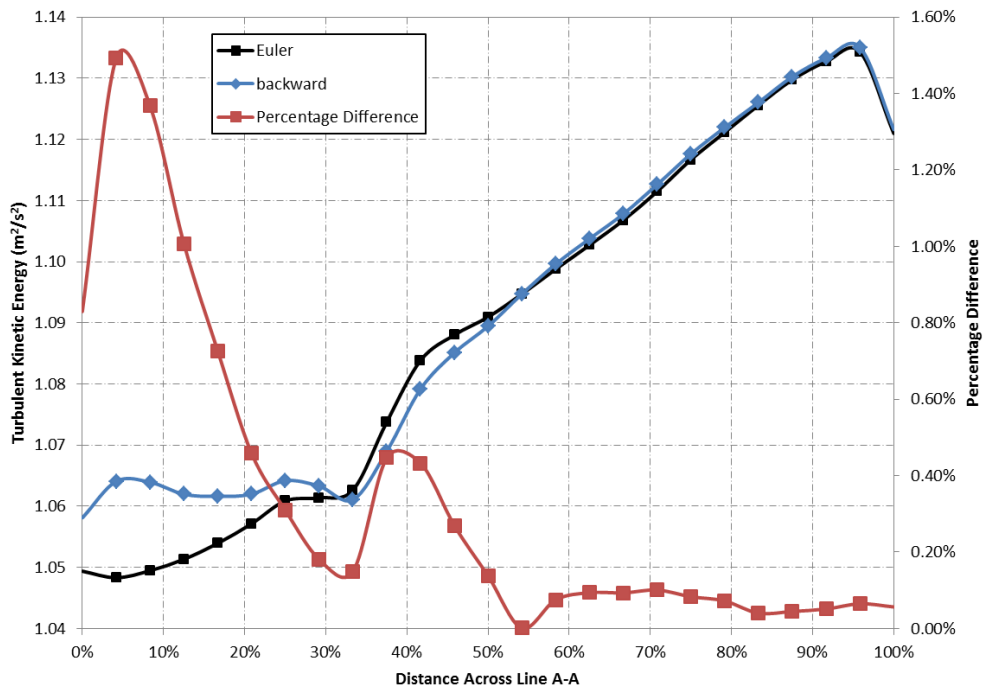


Figure A31 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 3)

A4. Velocity Convection Scheme (Study 4)

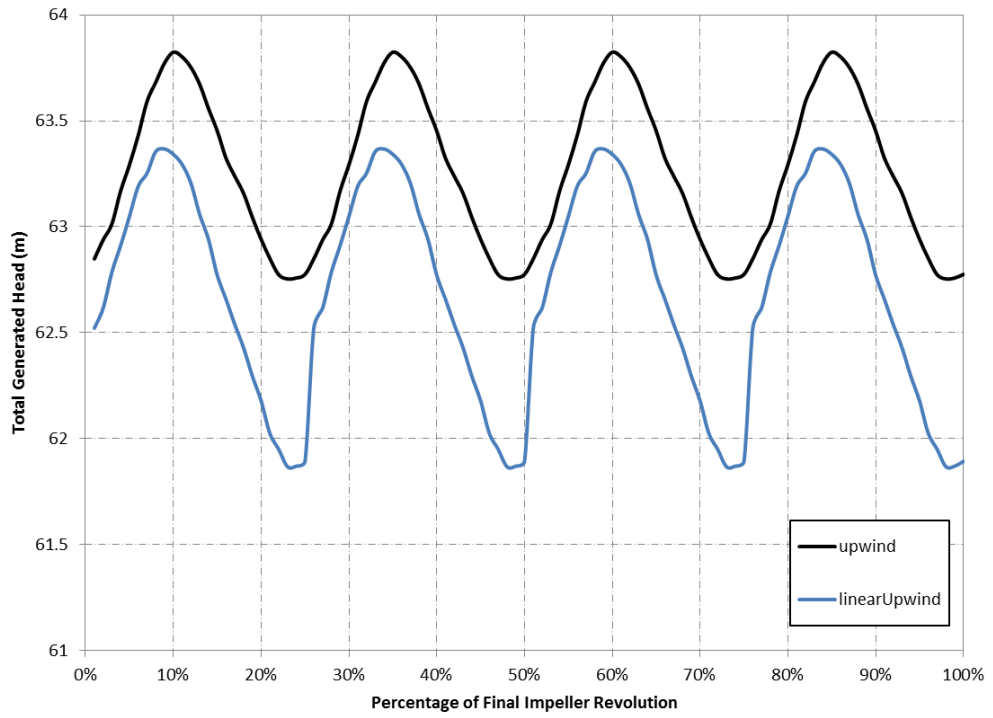


Figure A32 - Head Generated by Impeller during Final Revolution of Impeller (Study 4)

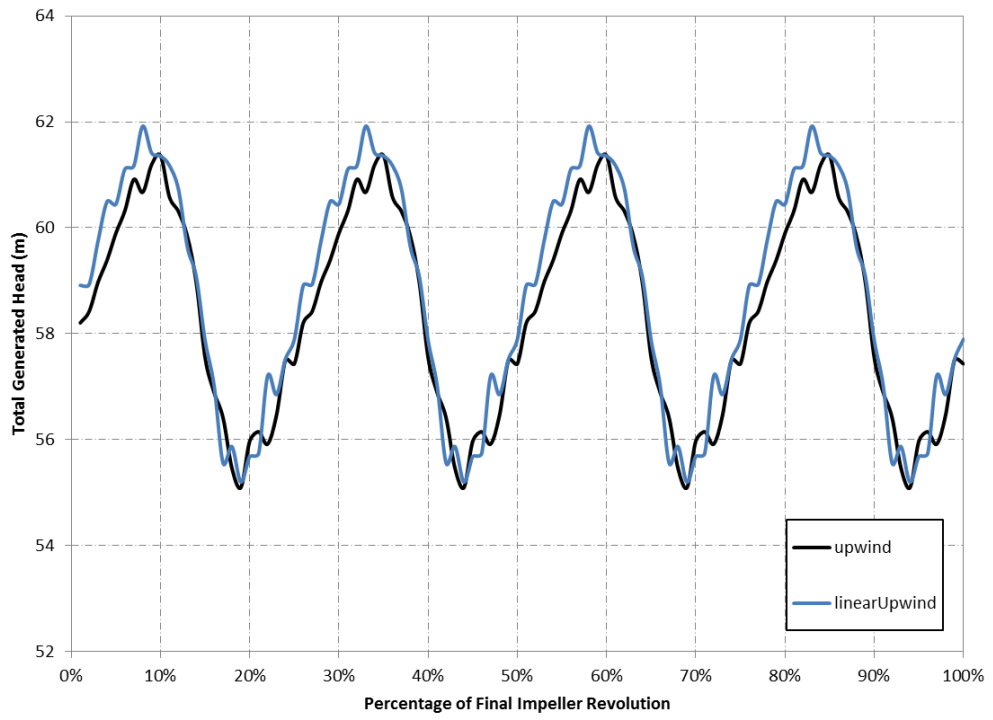


Figure A33 - Head Generated by Pump during Final Revolution of Impeller (Study 4)

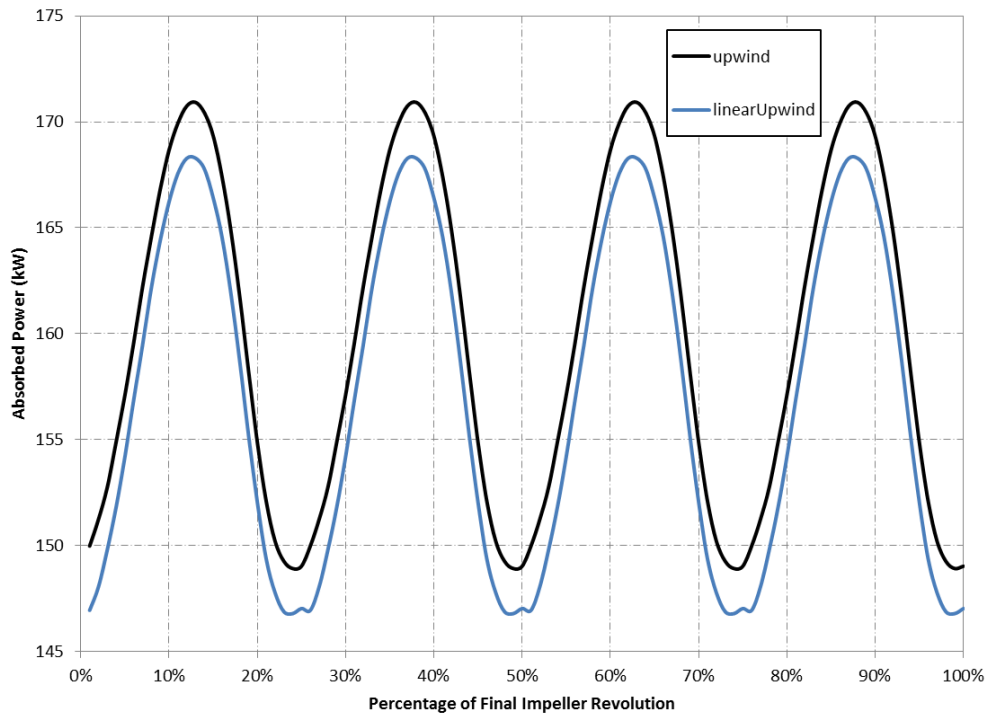


Figure A34 - Absorbed Power during Final Revolution of Impeller (Study 4)

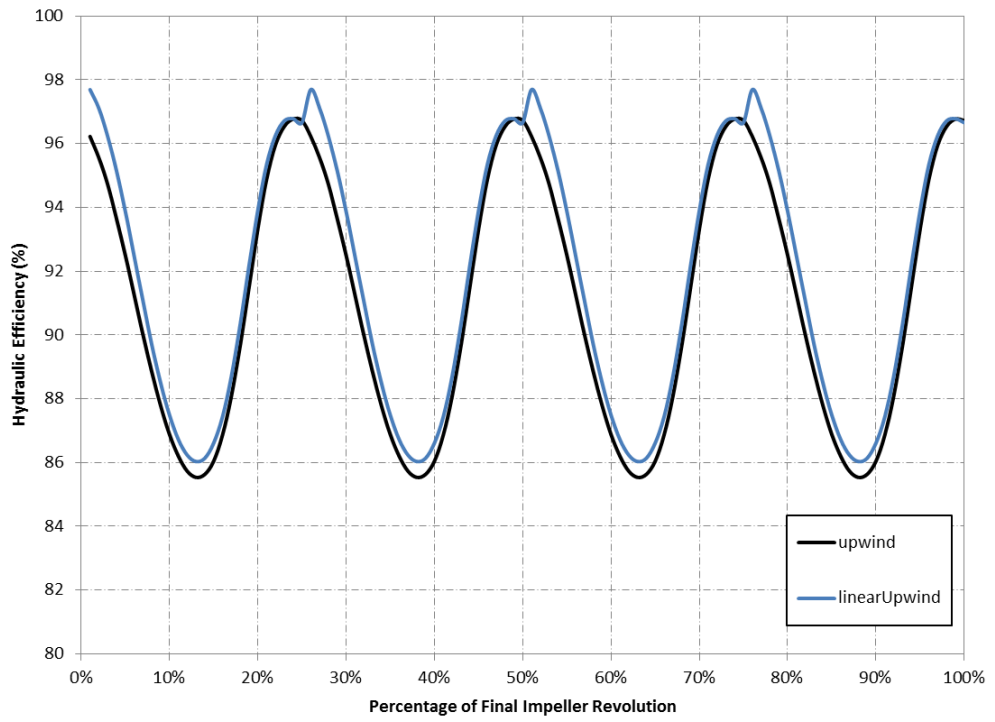


Figure A35 - Impeller Hydraulic Efficiency during Final Revolution of Impeller (Study 4)

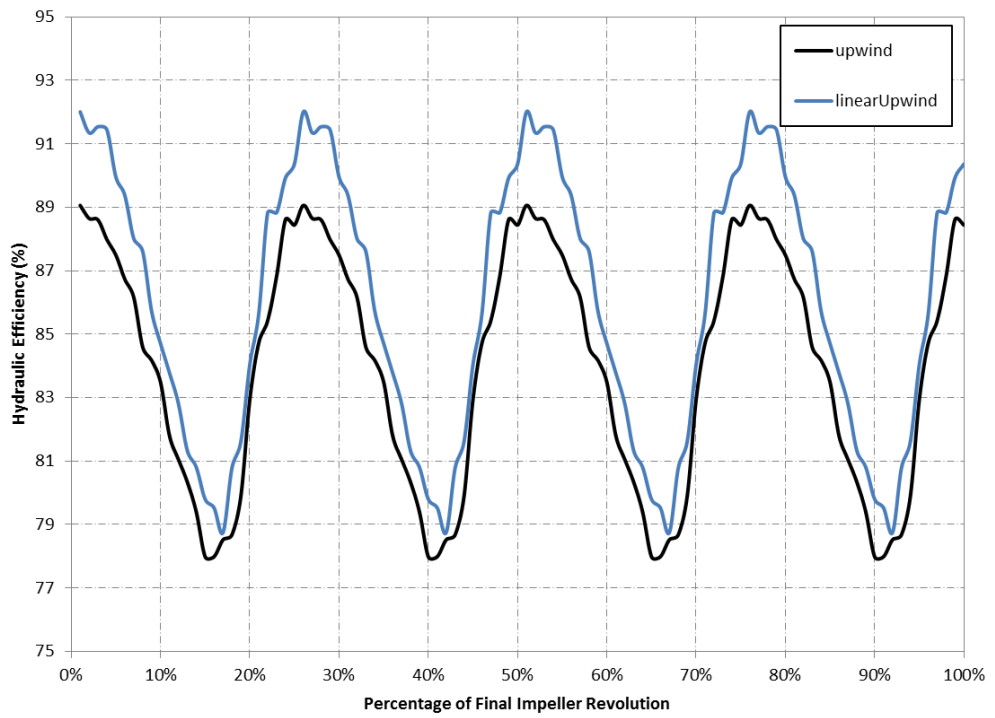
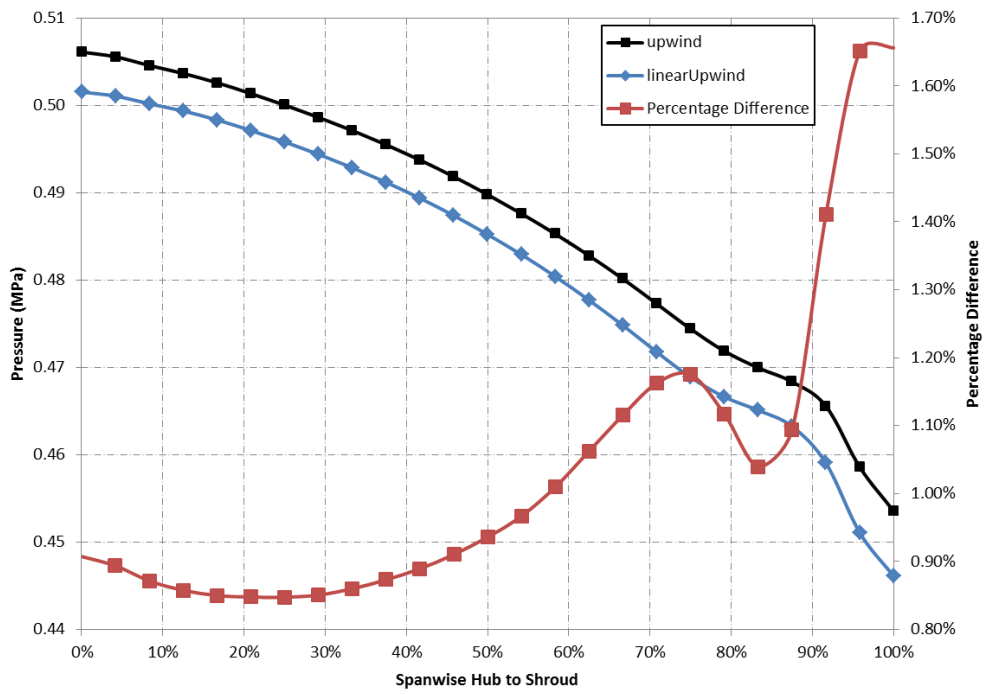
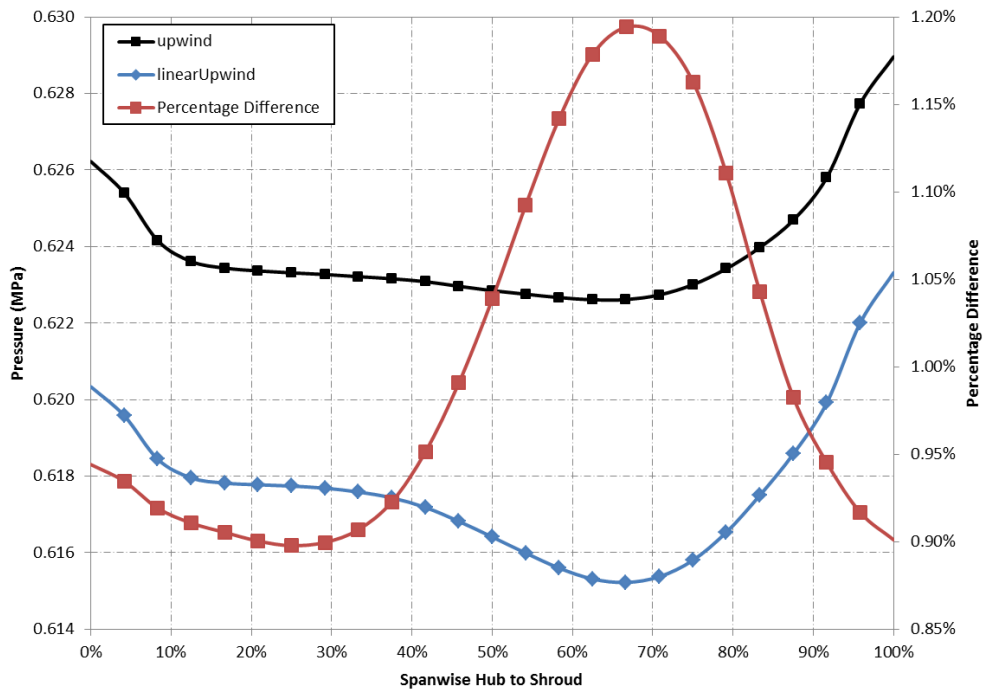


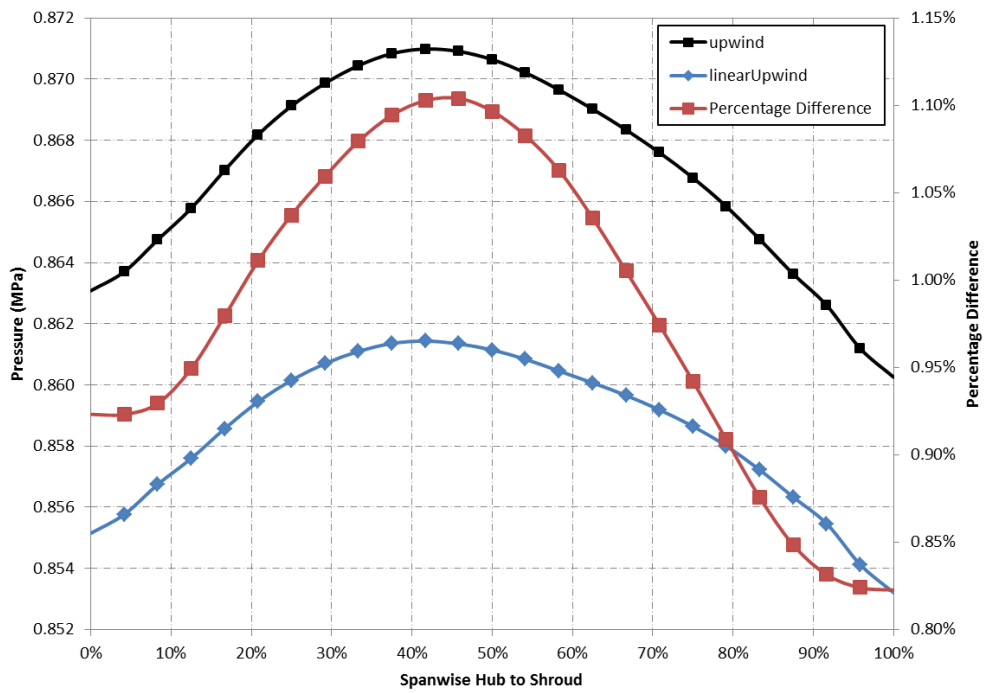
Figure A36 - Pump Hydraulic Efficiency during Final Revolution of Impeller (Study 4)



(a)

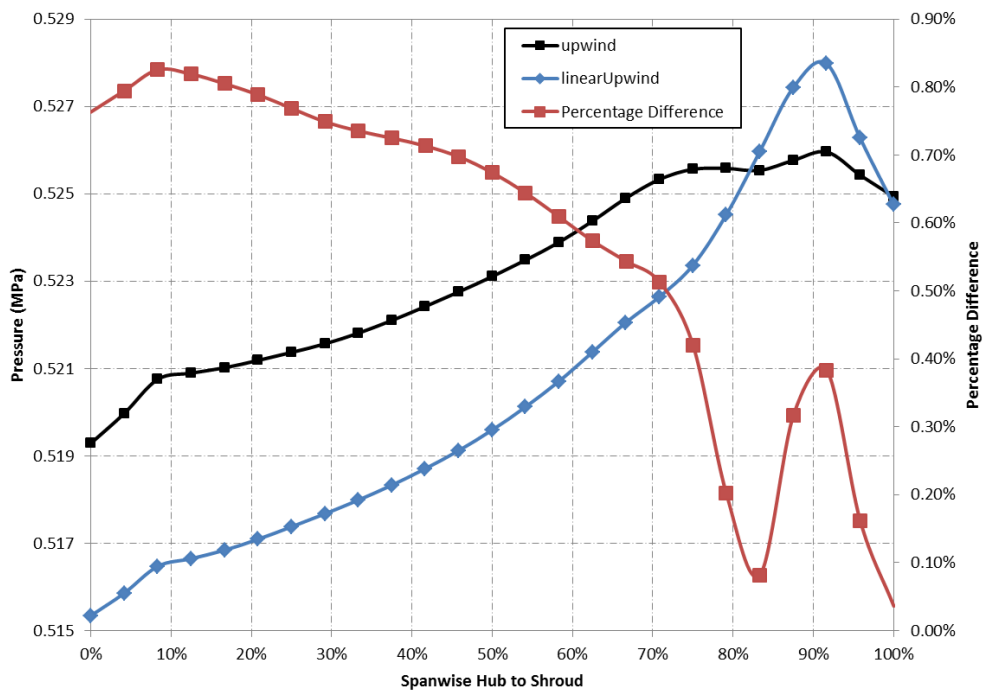


(b)

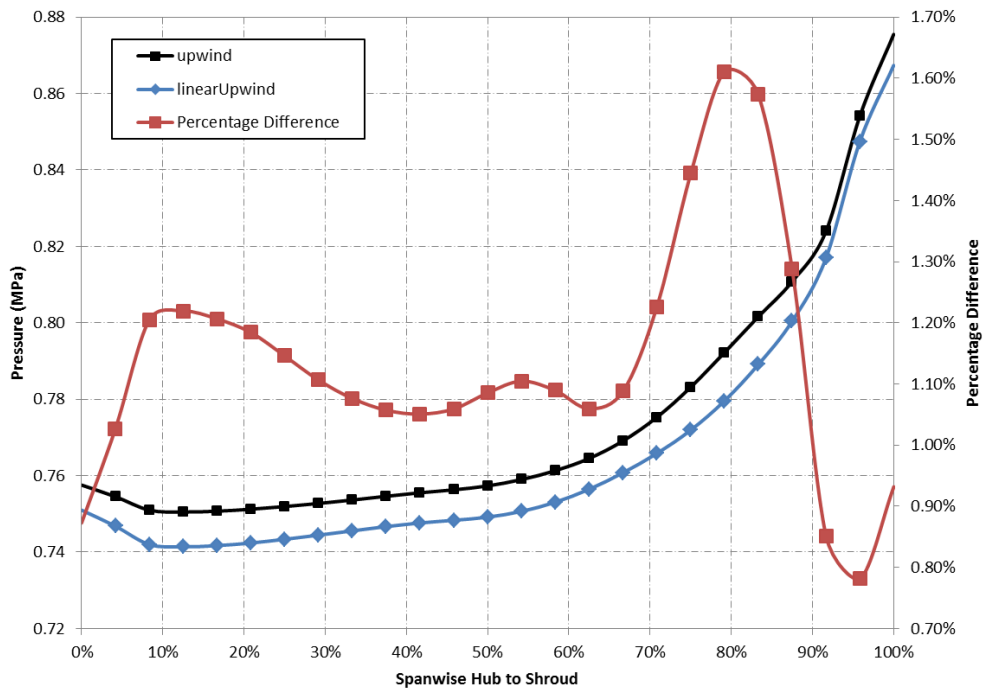


(c)

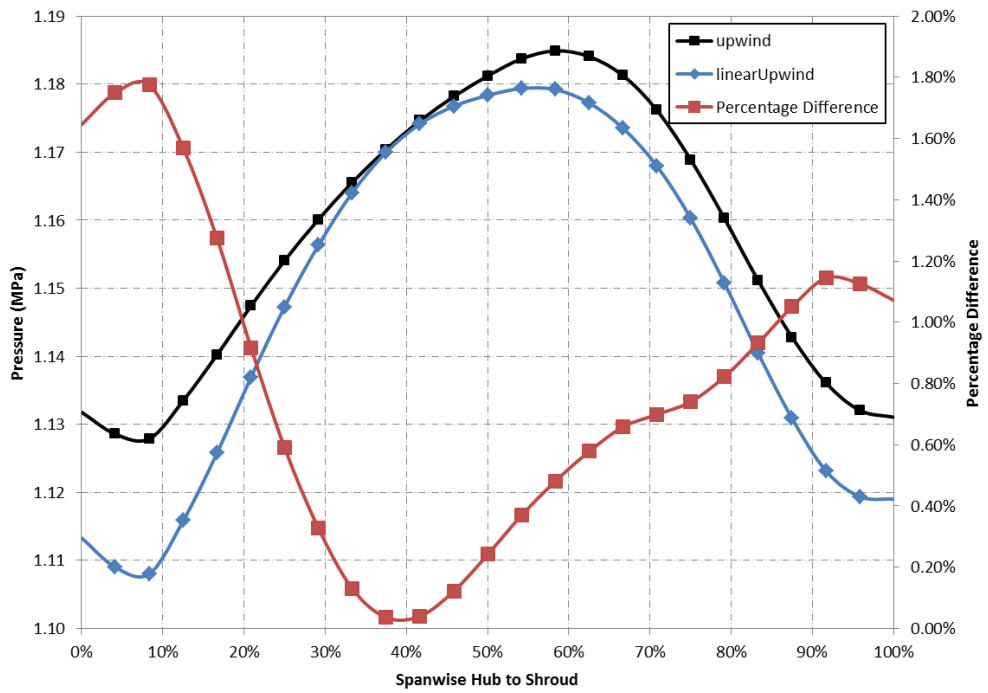
Figure A37 - Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 4)



(a)

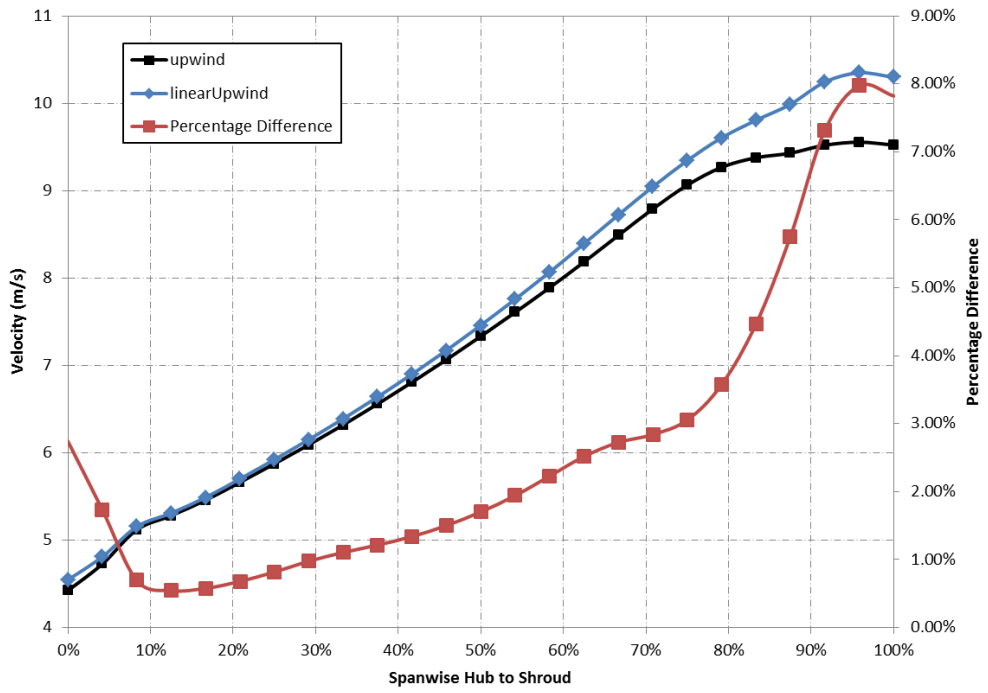


(b)

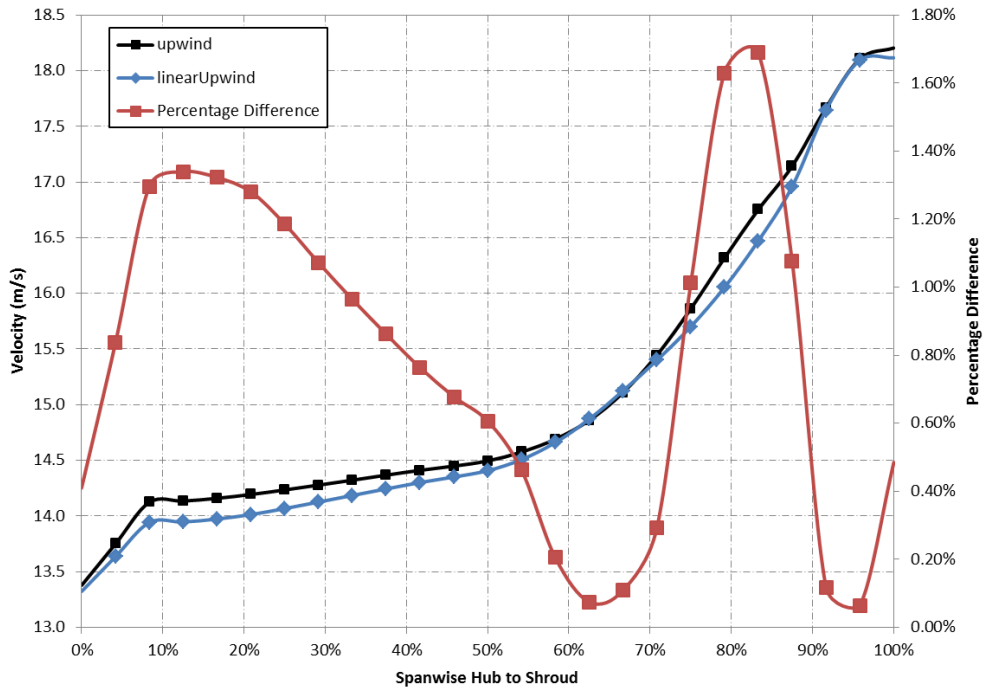


(c)

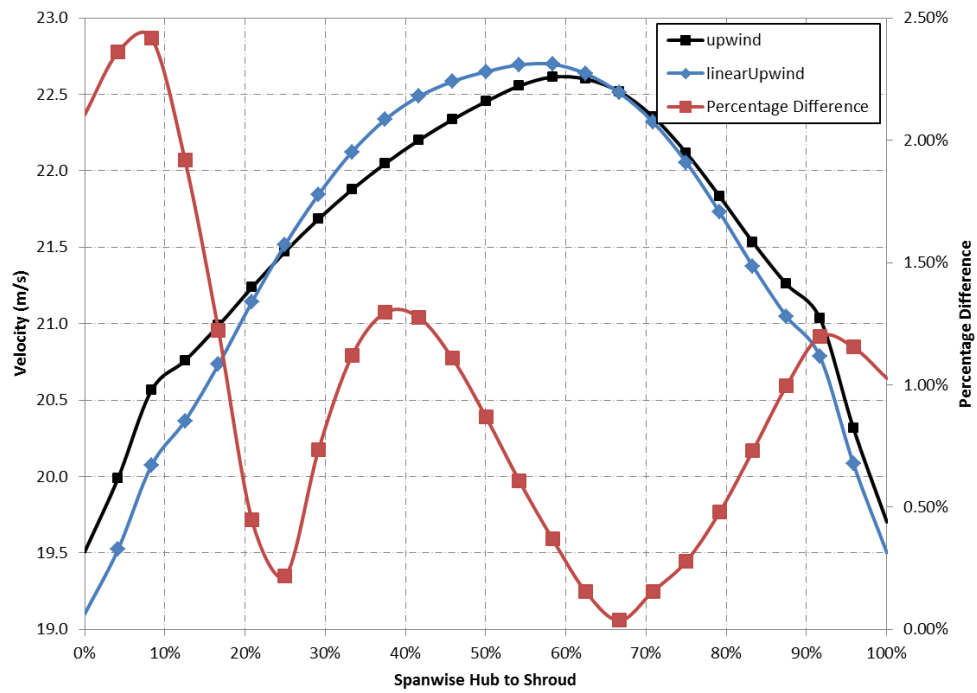
Figure A38 - Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 4)



(a)



(b)



(c)

Figure A39 - Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Study 4)

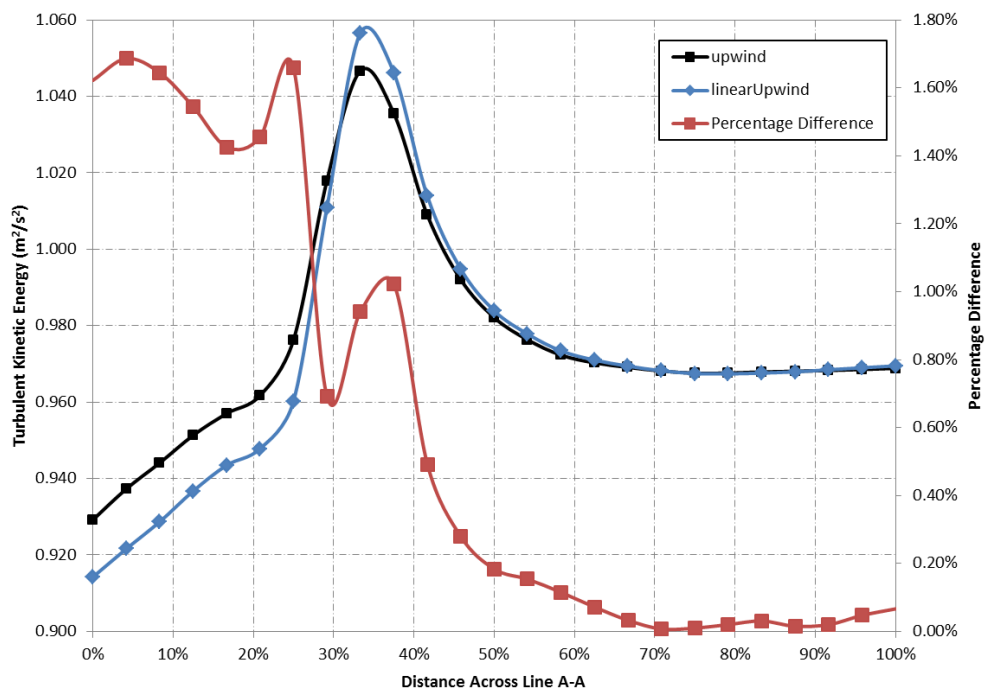


Figure A40 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 4)

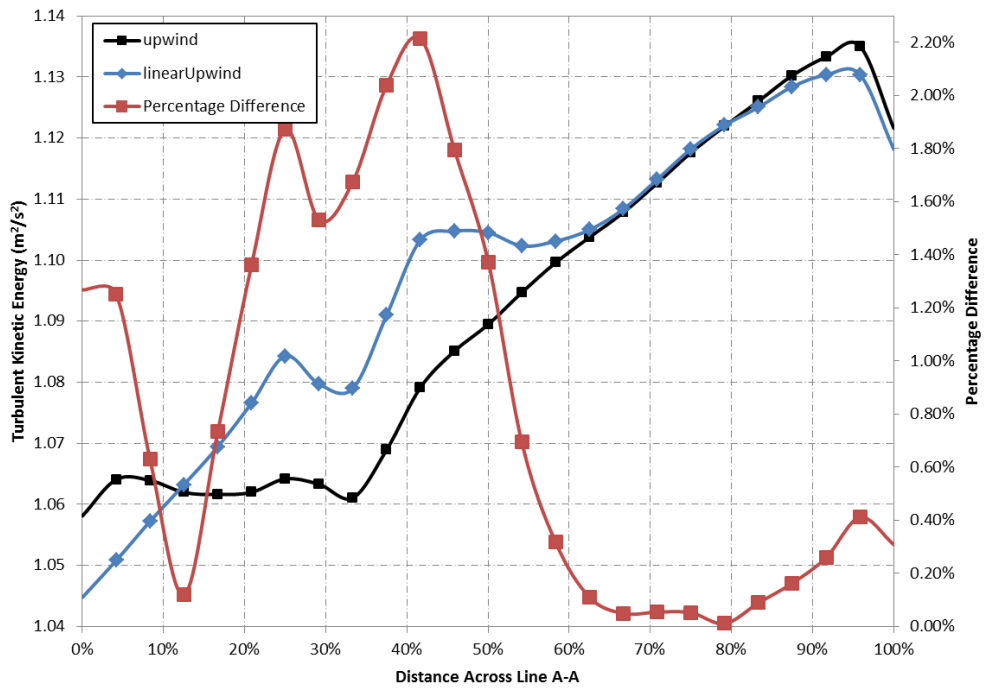


Figure A41 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Study 4)

A5. Comparison between OpenFOAM and ANSYS CFX

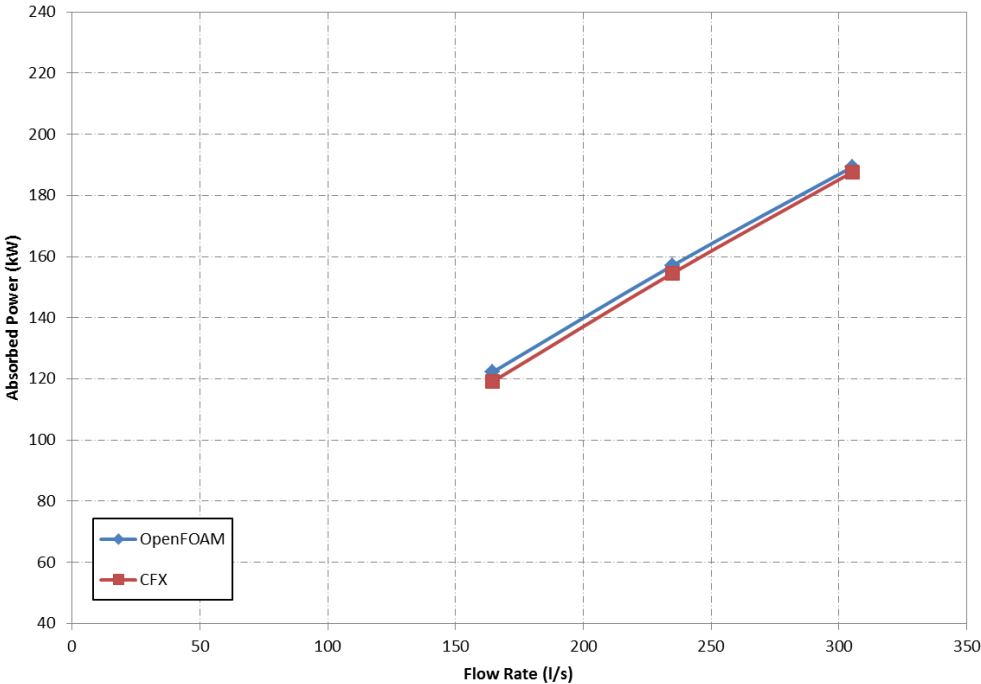


Figure A42 - Absorbed Power Comparison between OpenFOAM and ANSYS CFX

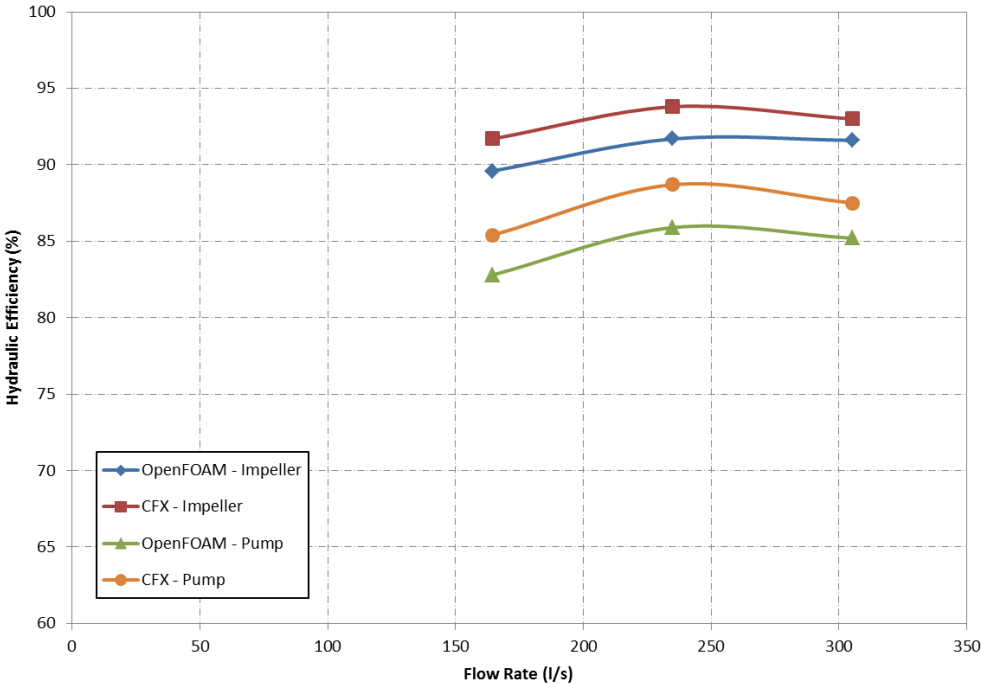
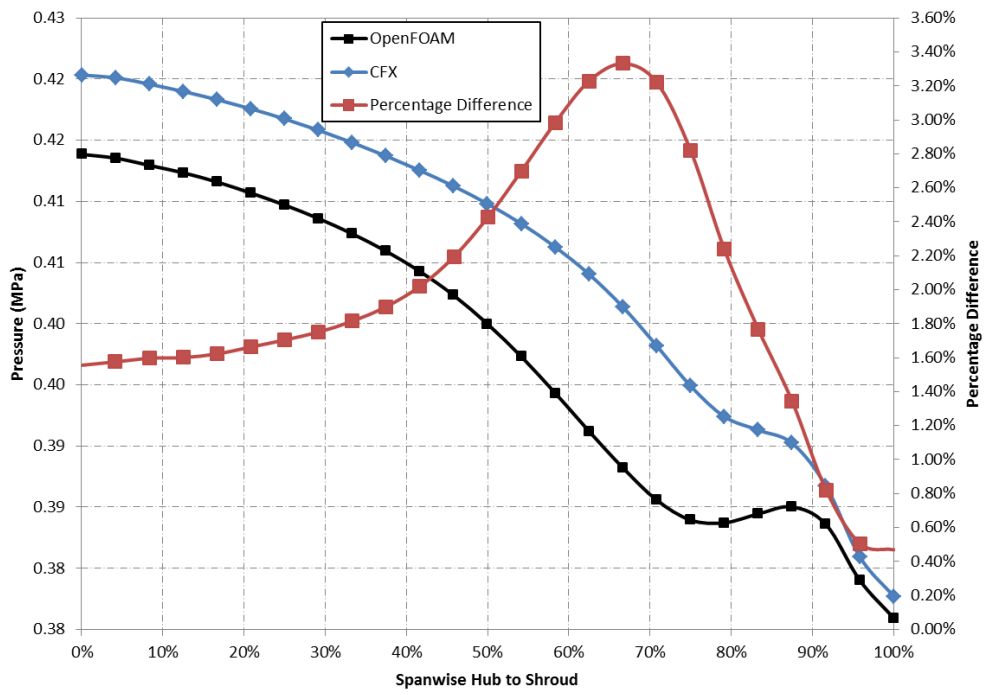
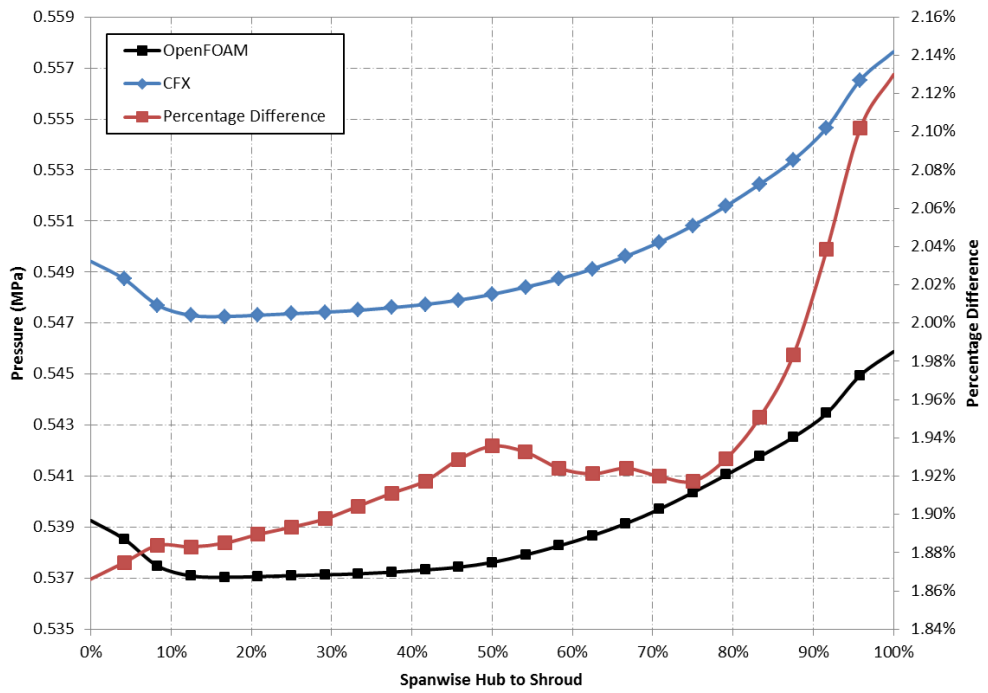


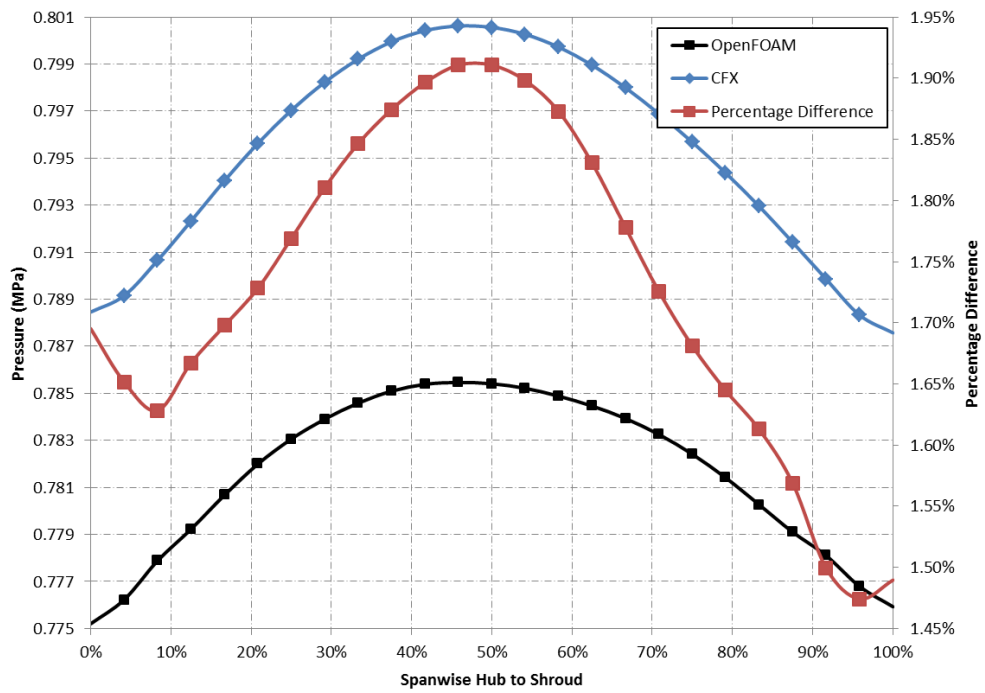
Figure A43 - Hydraulic Efficiency Comparison between OpenFOAM and ANSYS CFX



(a)

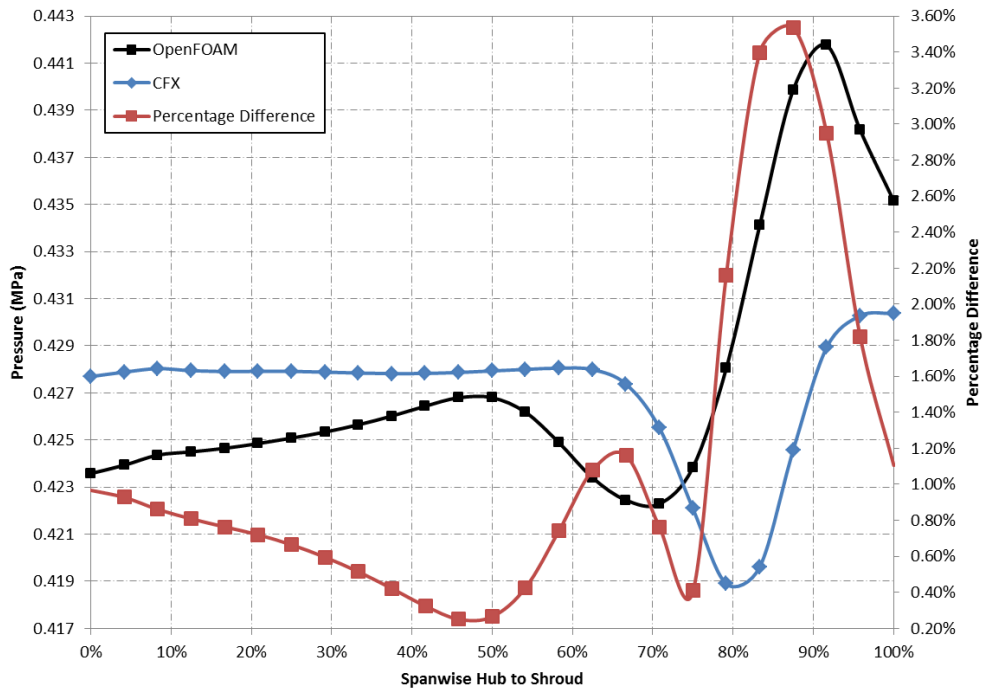


(b)

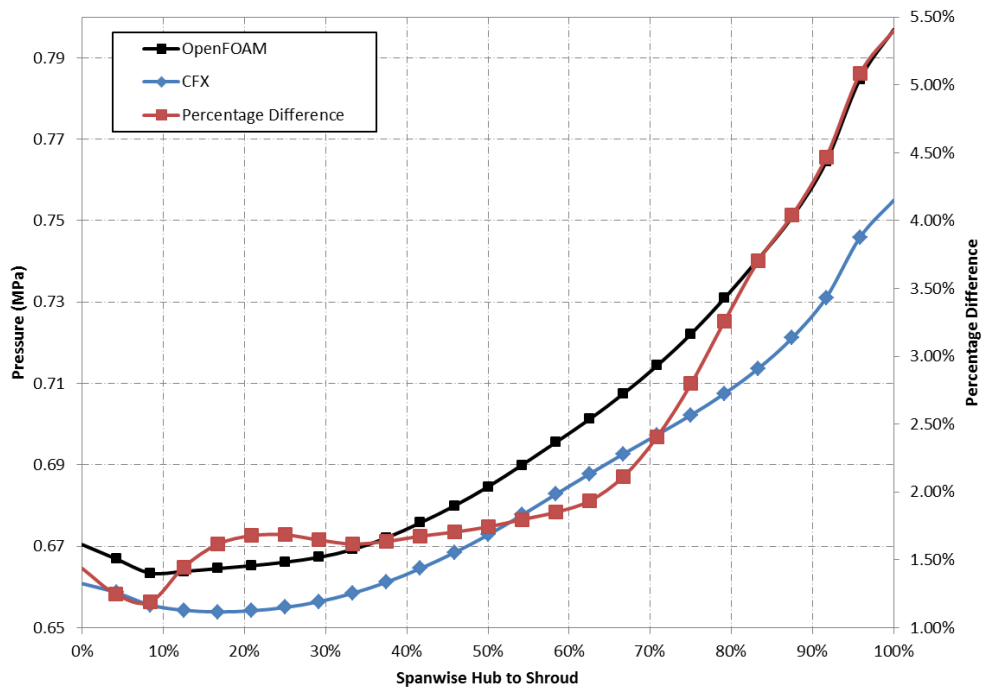


(c)

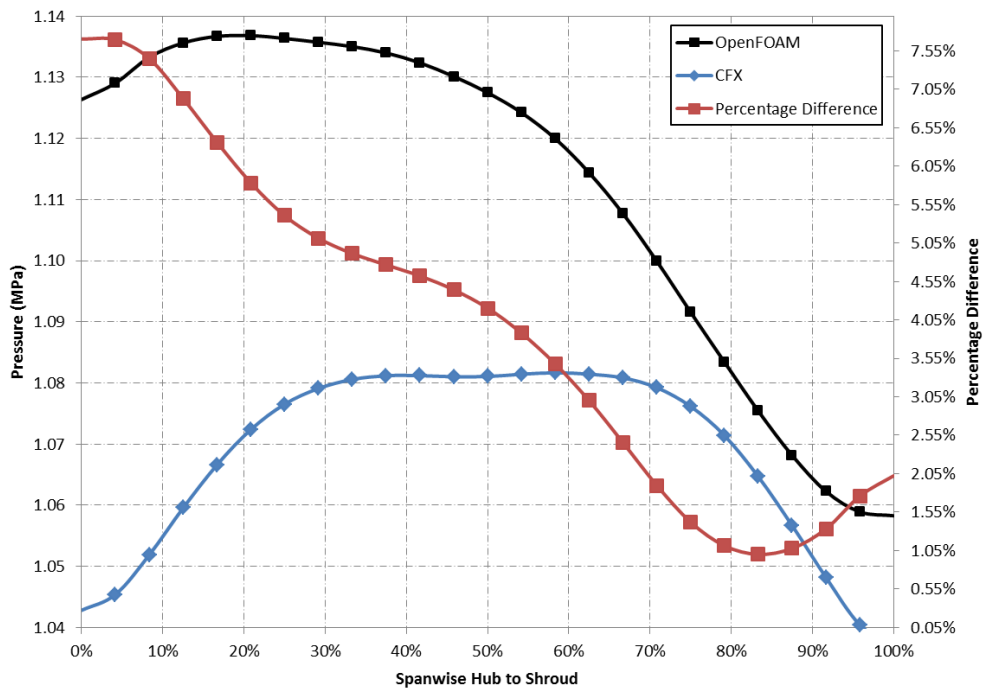
Figure A44 - Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 70% Q_{opt})



(a)

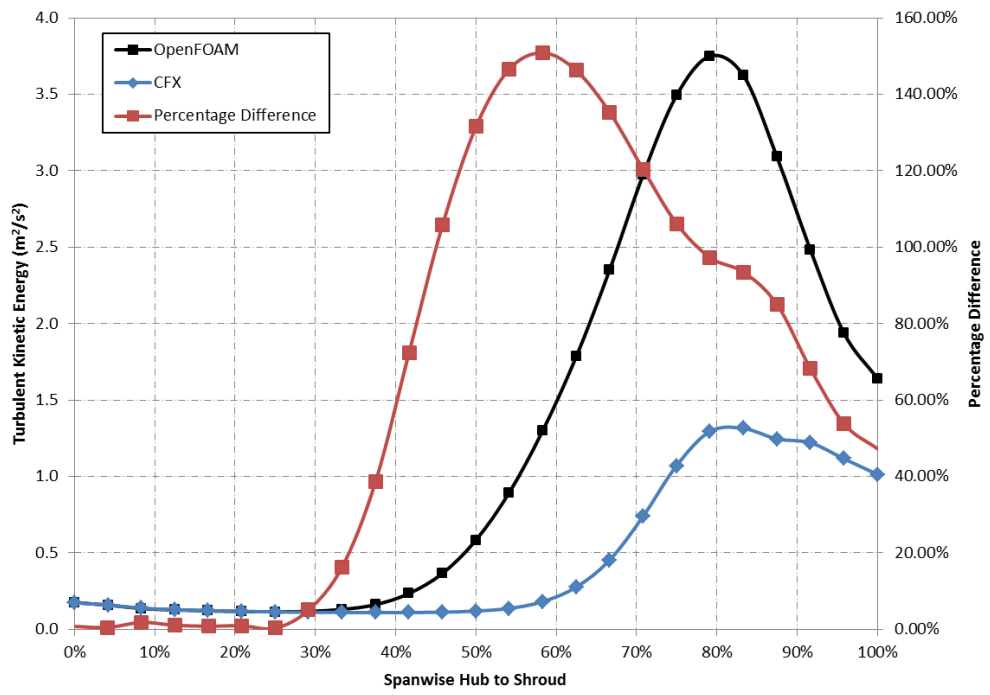


(b)

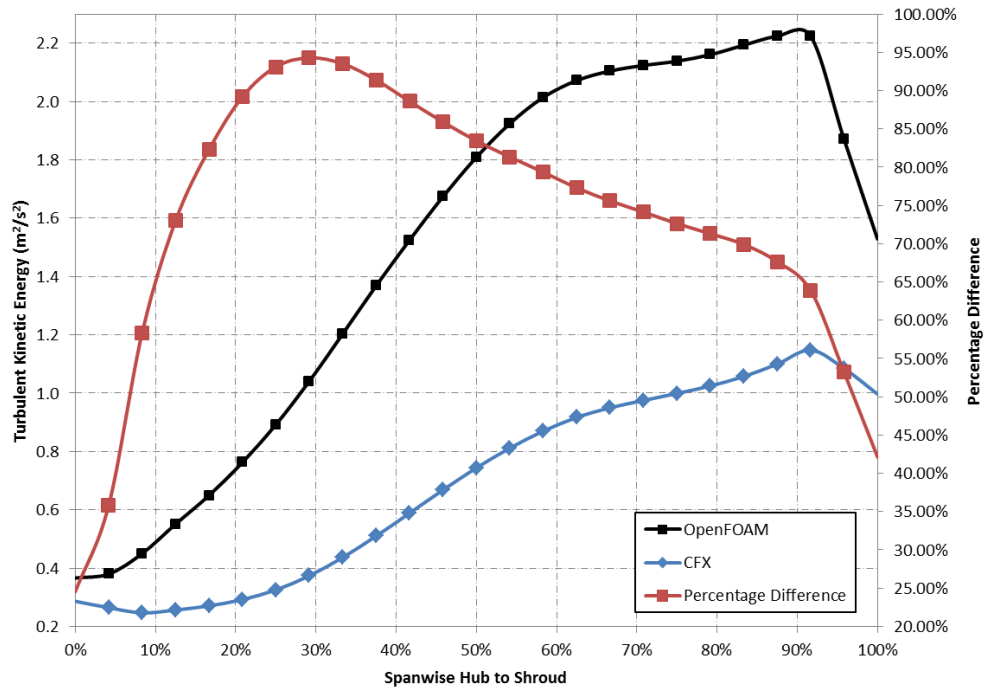


(c)

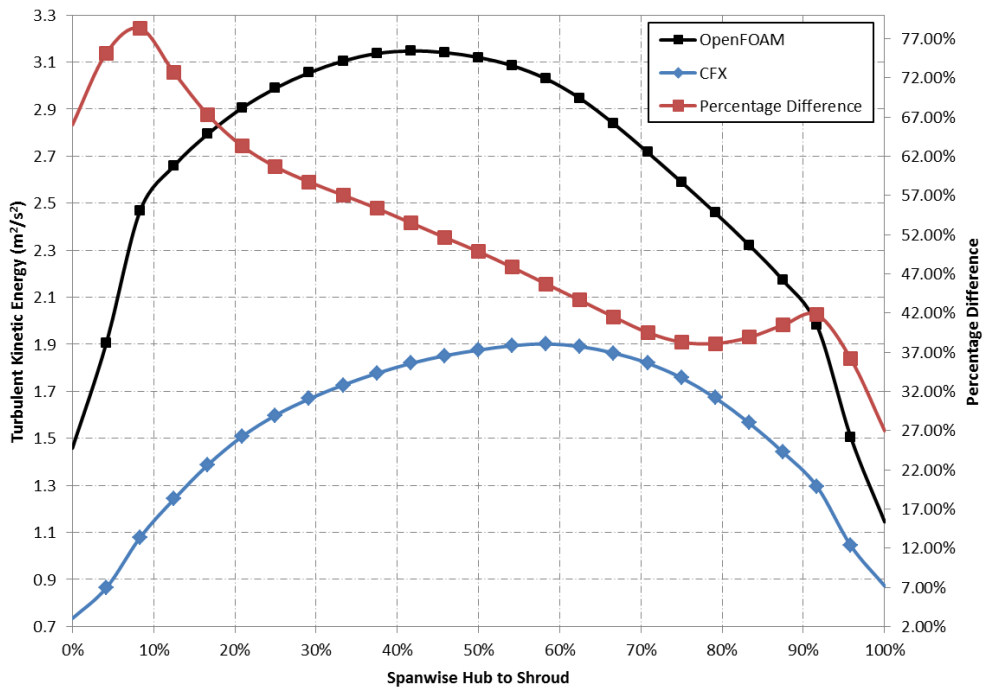
Figure A45 - Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85% (Software Comparison – 70% Q_{opt})



(a)



(b)



(c)

Figure A46 - Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 70% Q_{opt})

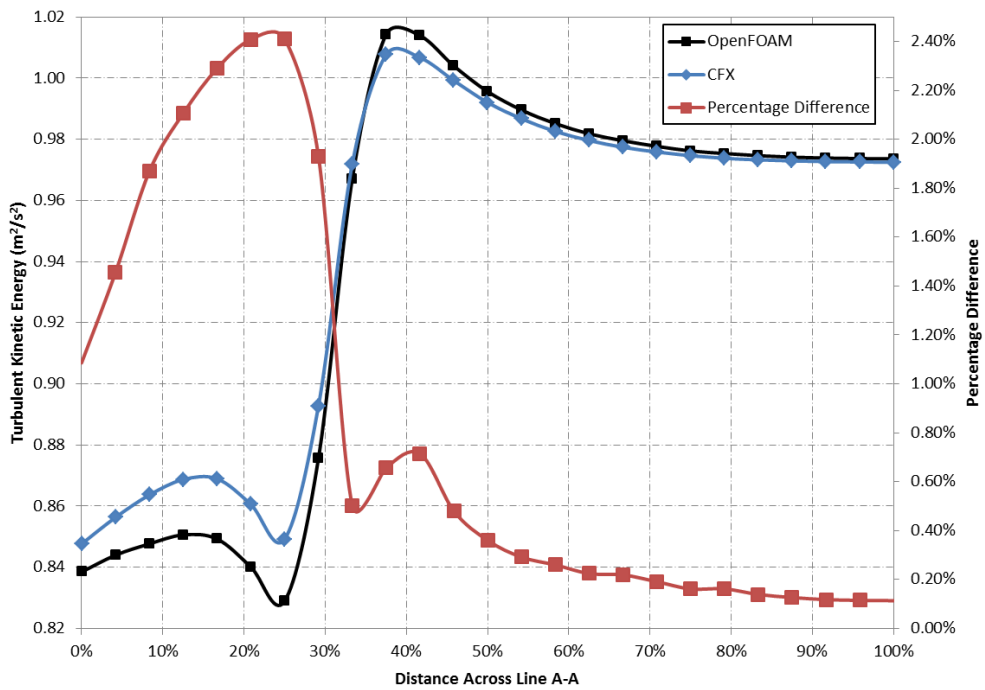


Figure A47 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 70% Q_{opt})

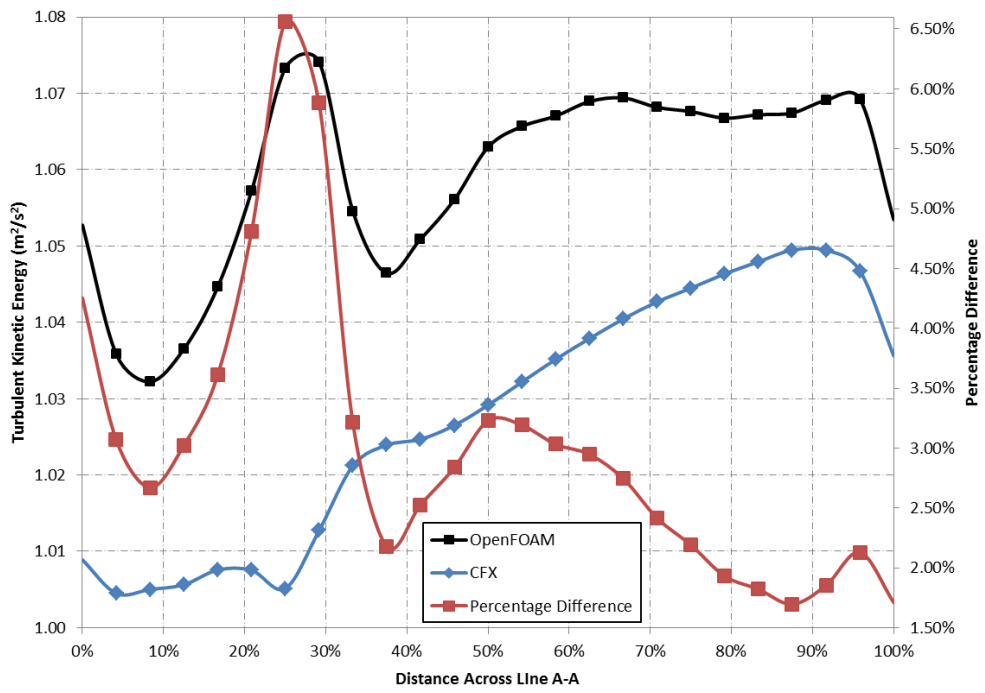


Figure A48 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 70% Q_{opt})

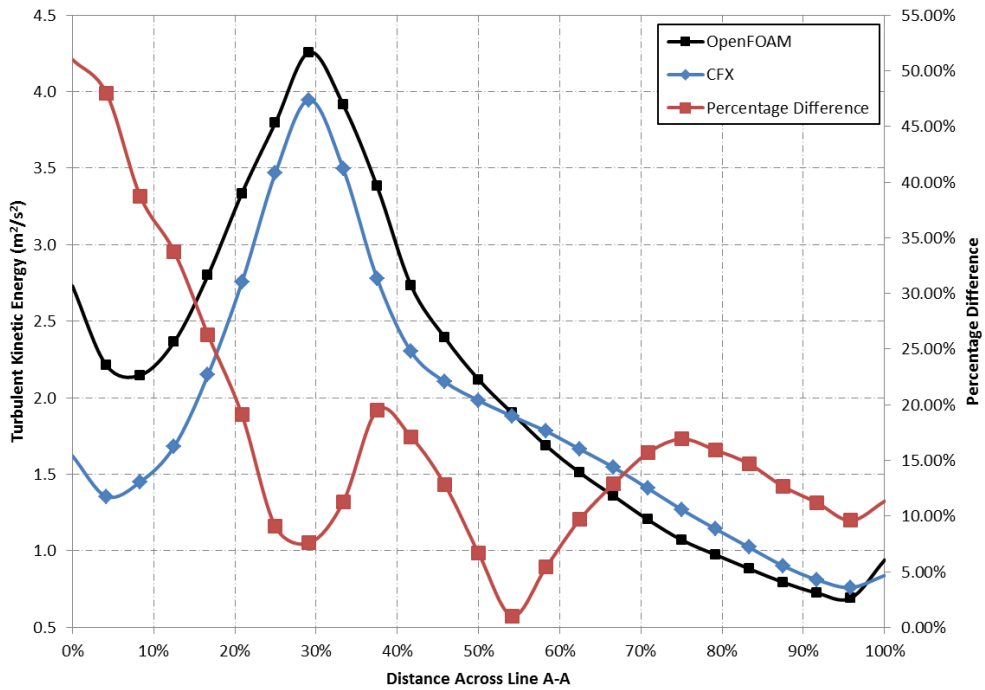
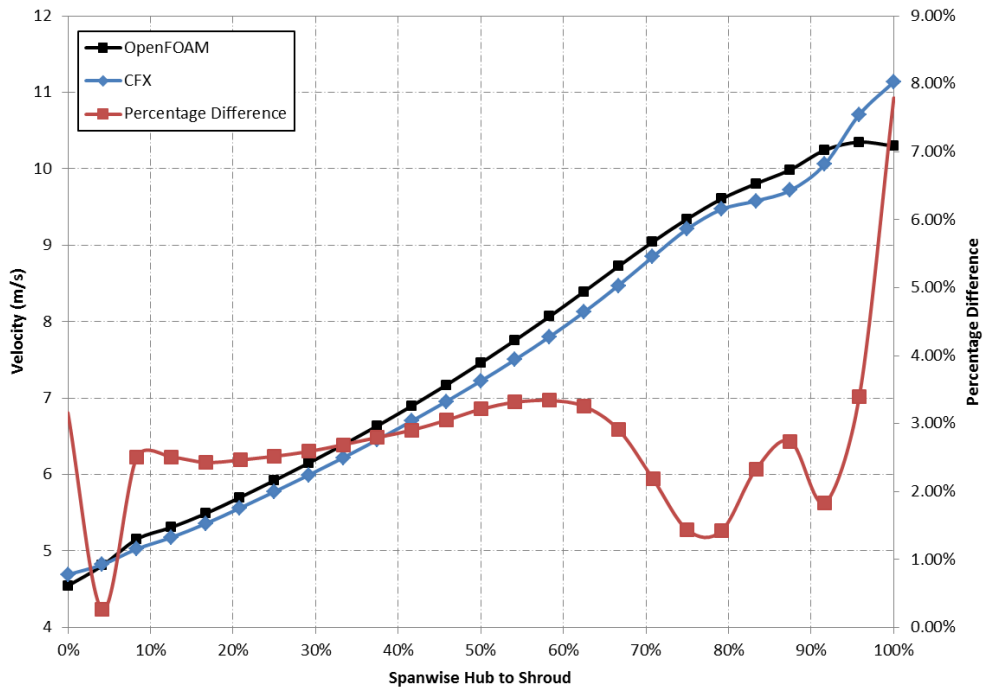
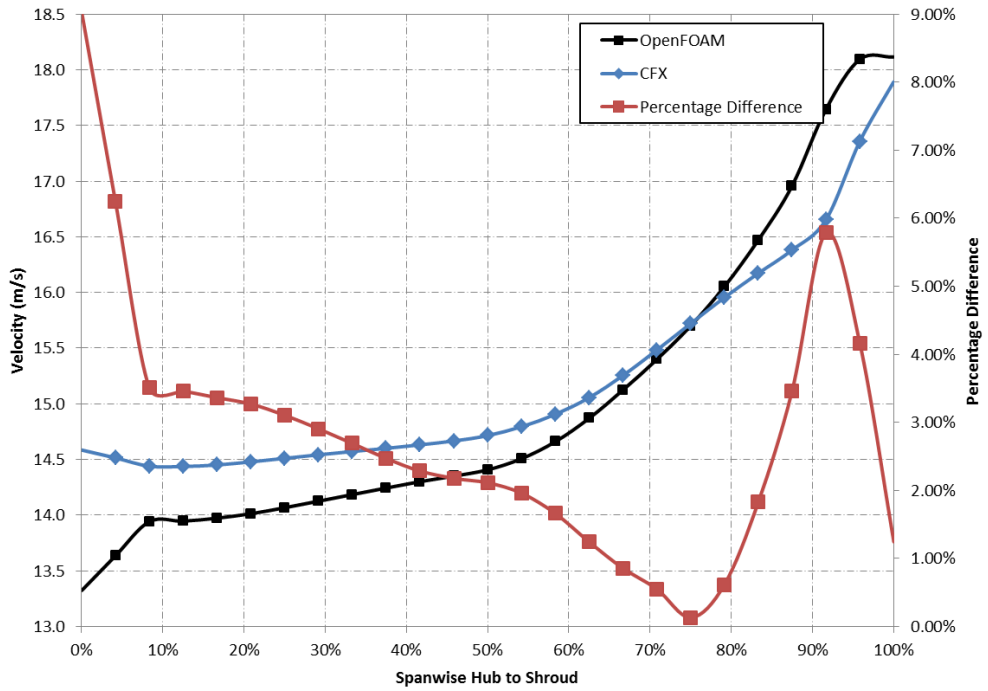


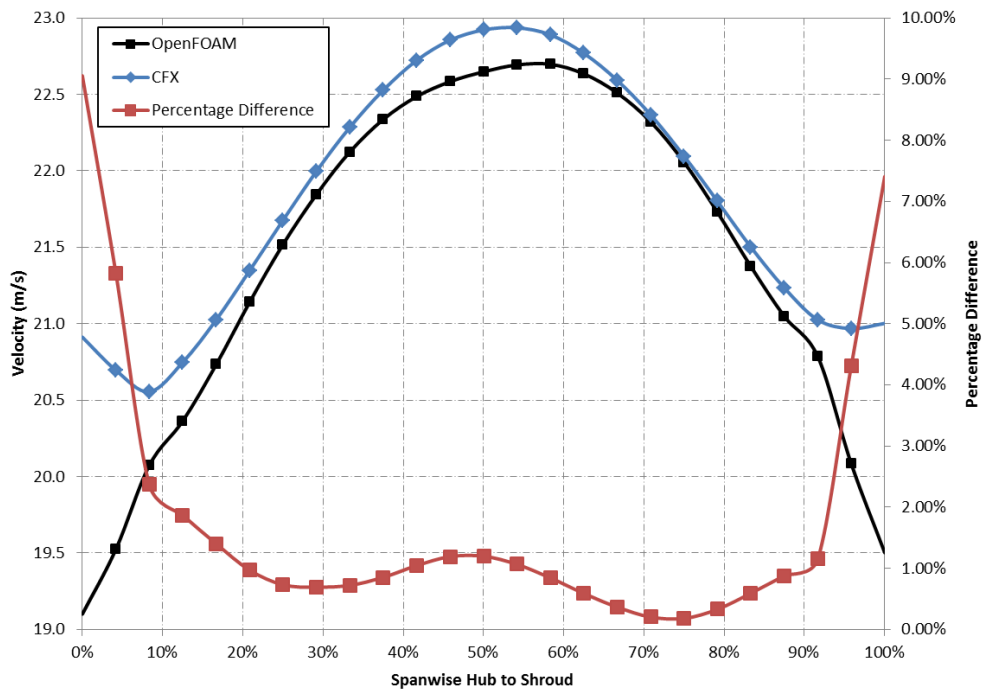
Figure A49 - Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 70% Q_{opt})



(a)



(b)



(c)

Figure A50 - Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 100% Q_{opt})

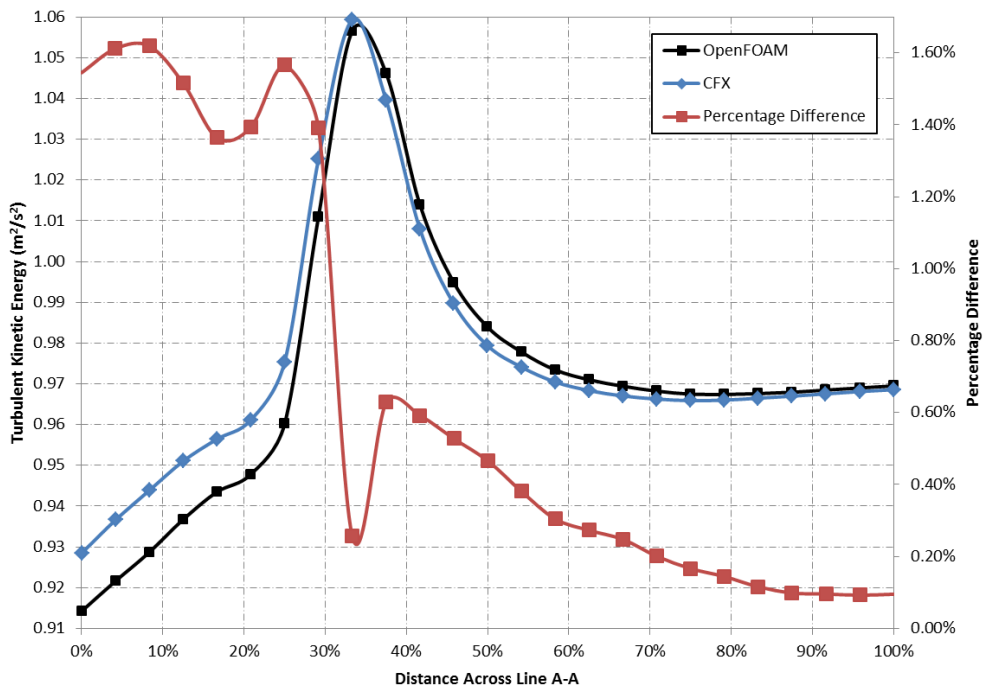


Figure A51 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 100% Q_{opt})

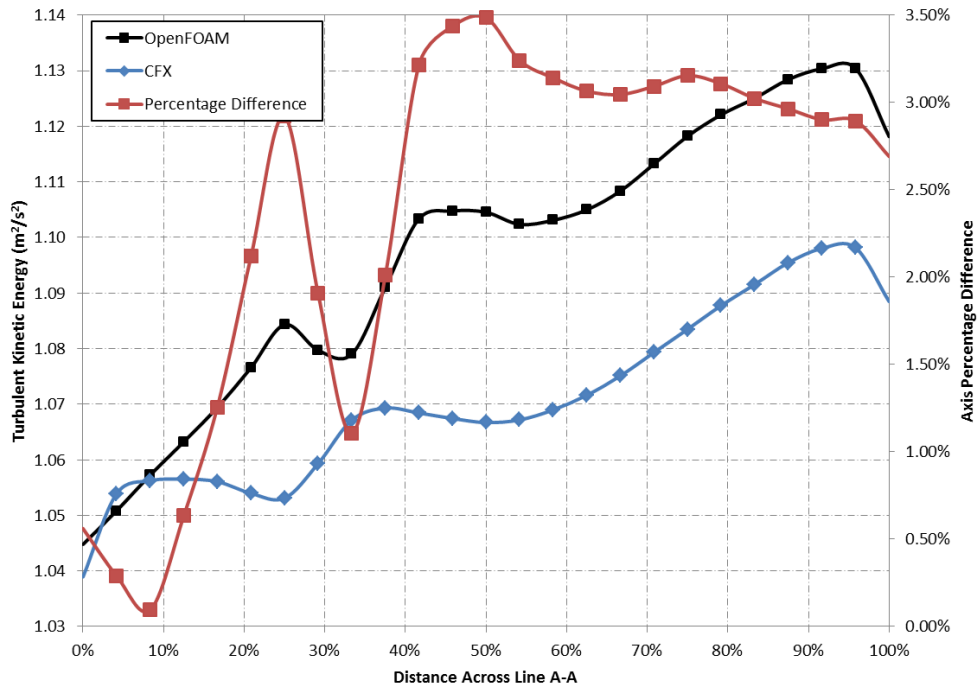


Figure A52 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 100% Q_{opt})

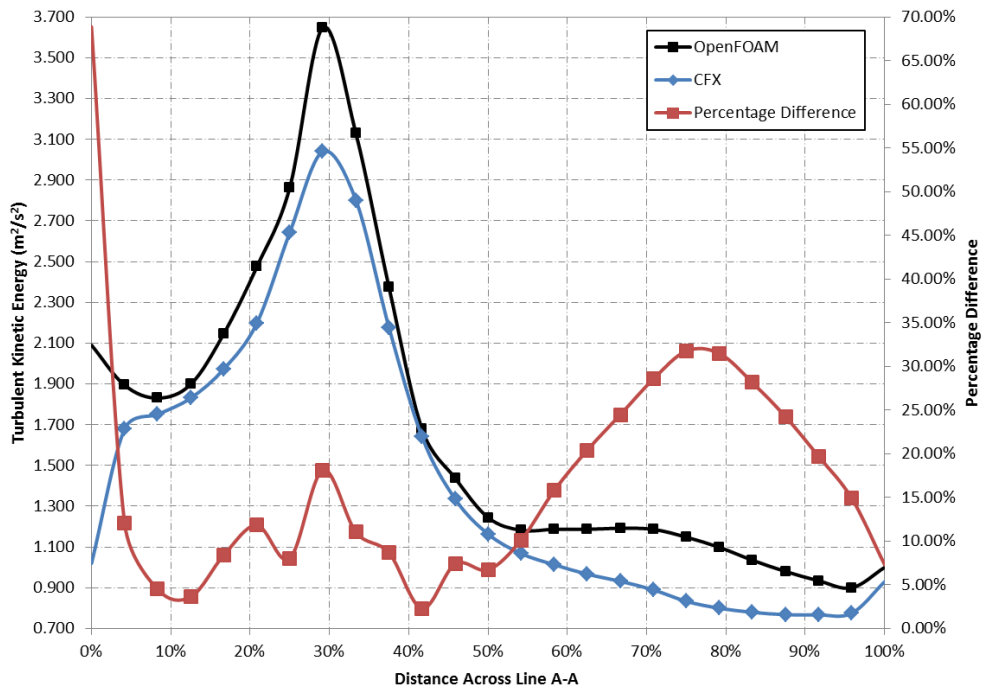
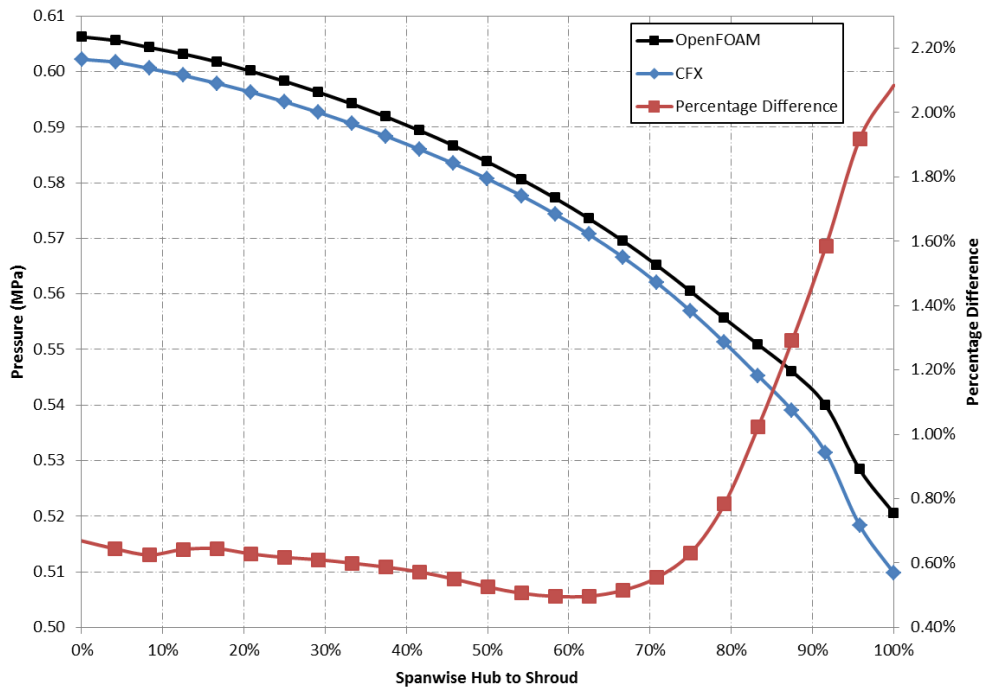
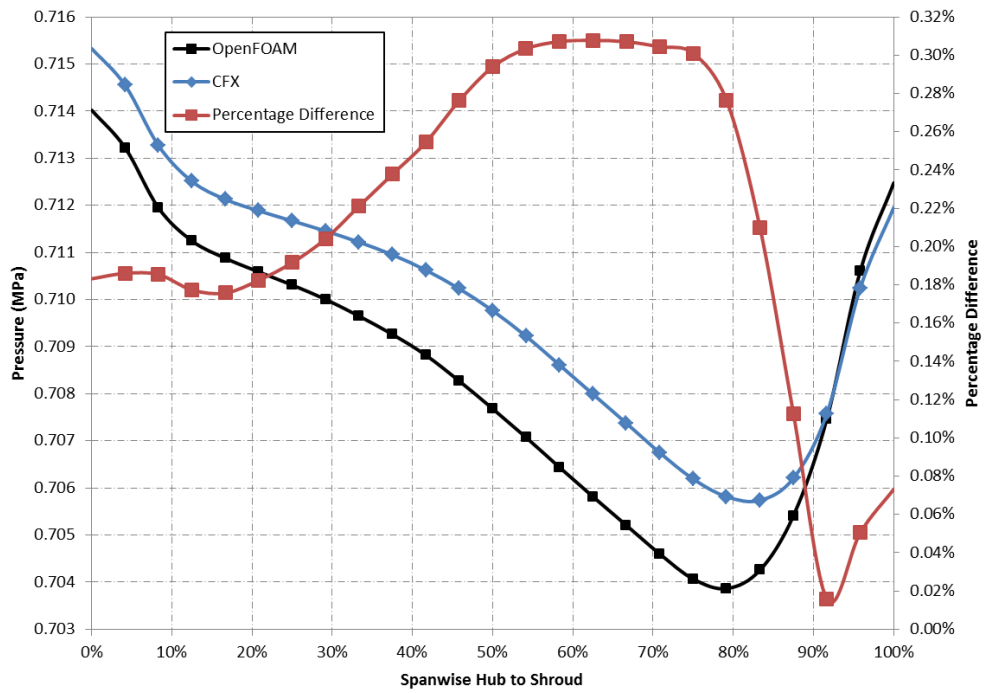


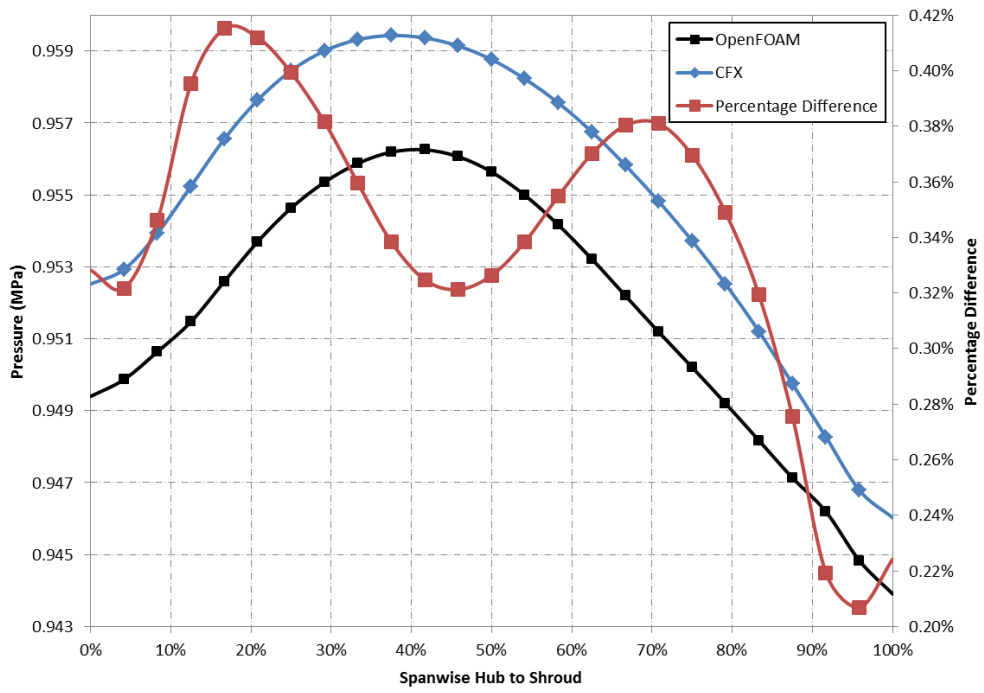
Figure A53 - Turbulent Kinetic Energy across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 100% Q_{opt})



(a)

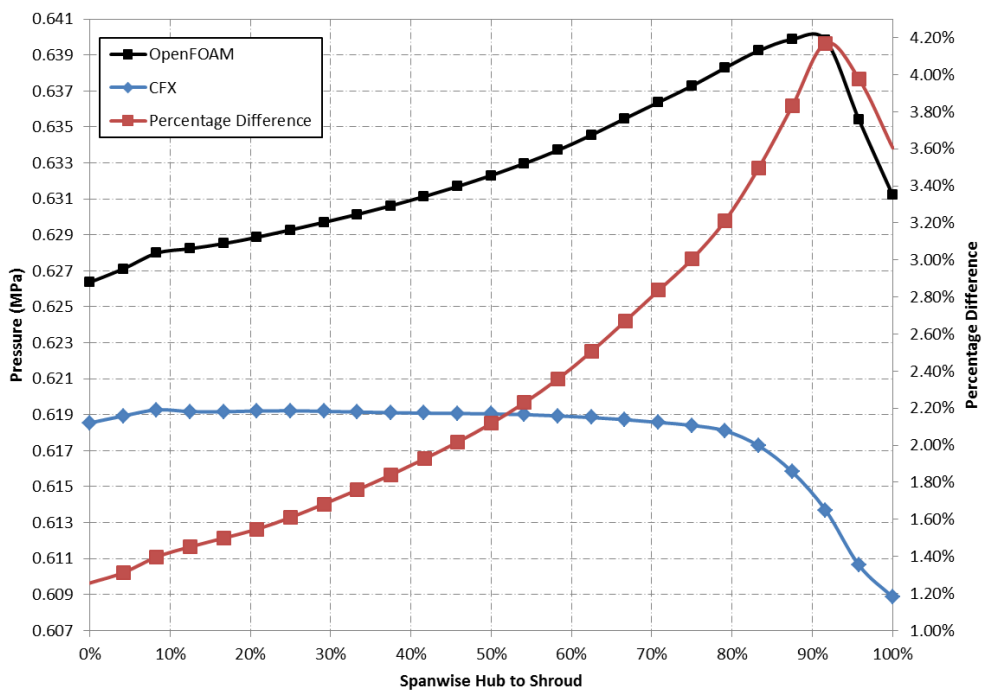


(b)

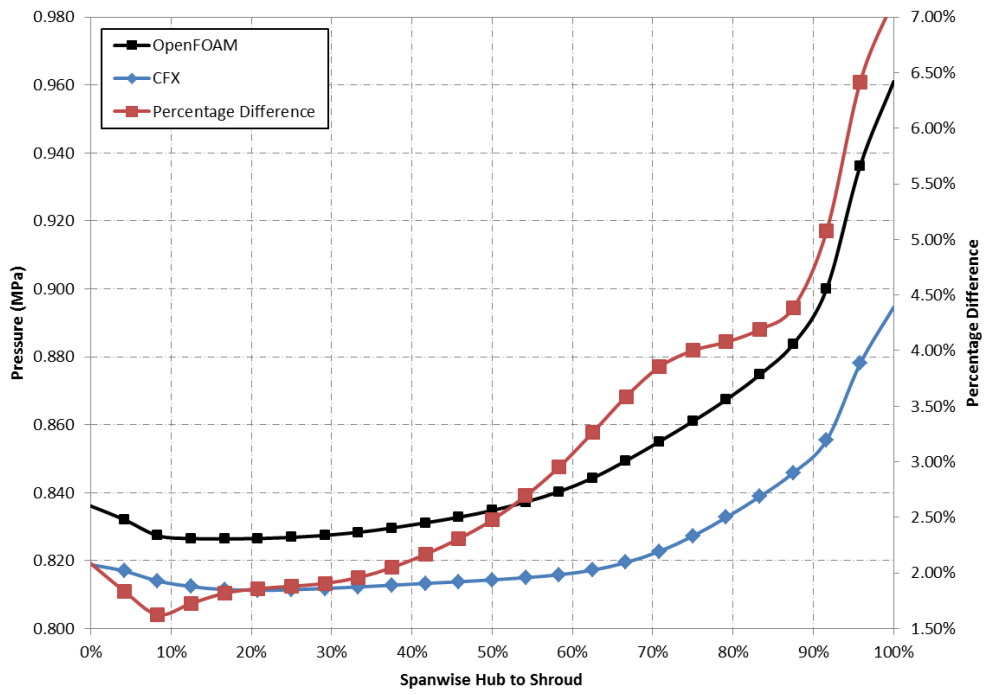


(c)

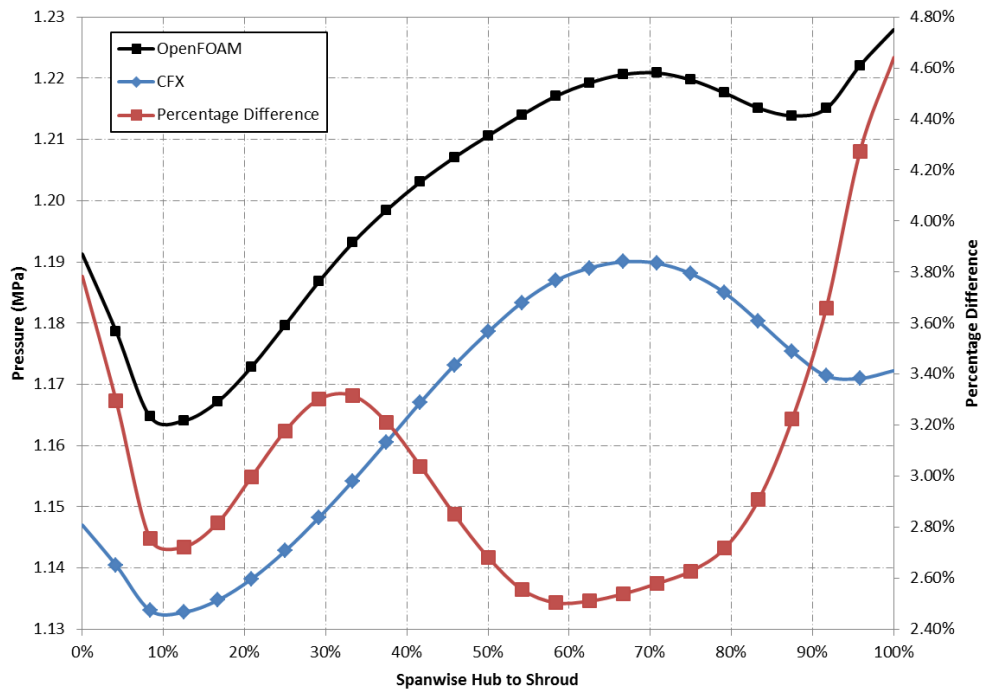
Figure A54 - Static Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 130% Q_{opt})



(a)

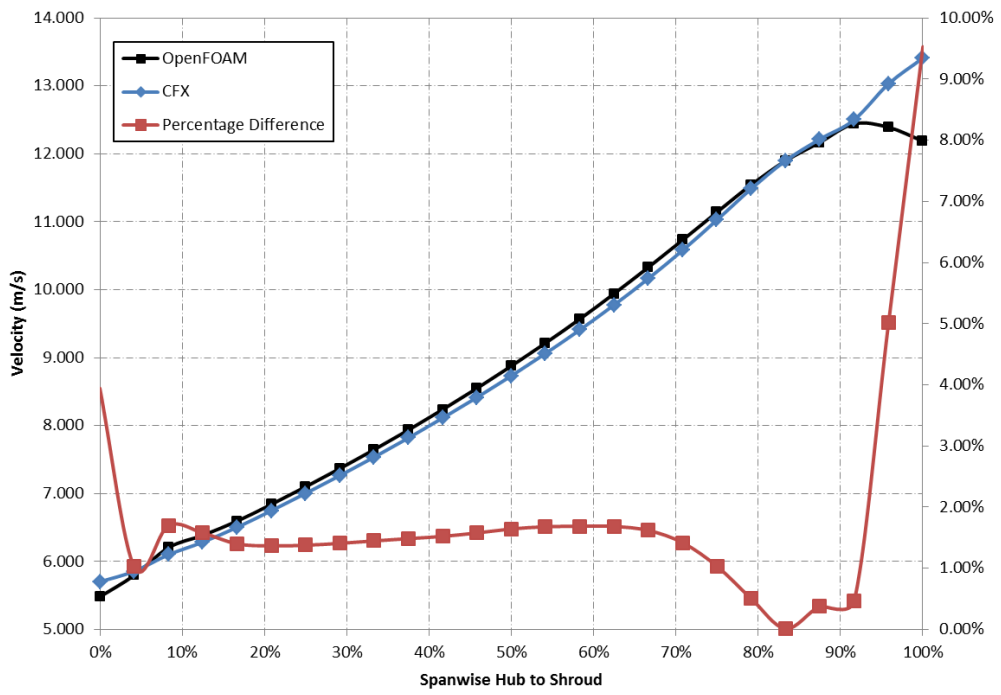


(b)

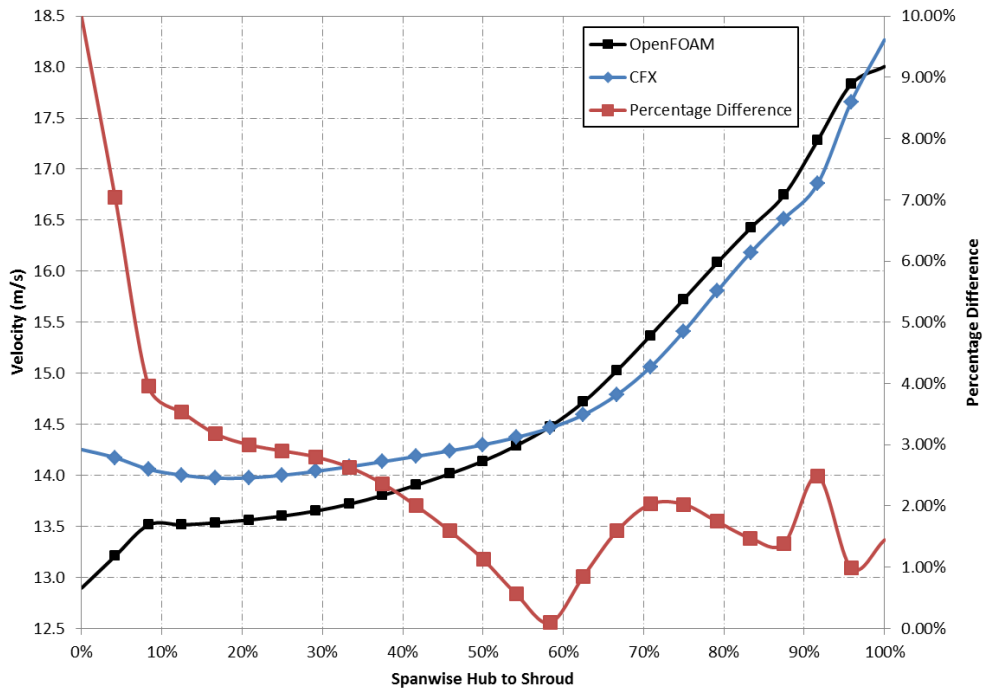


(c)

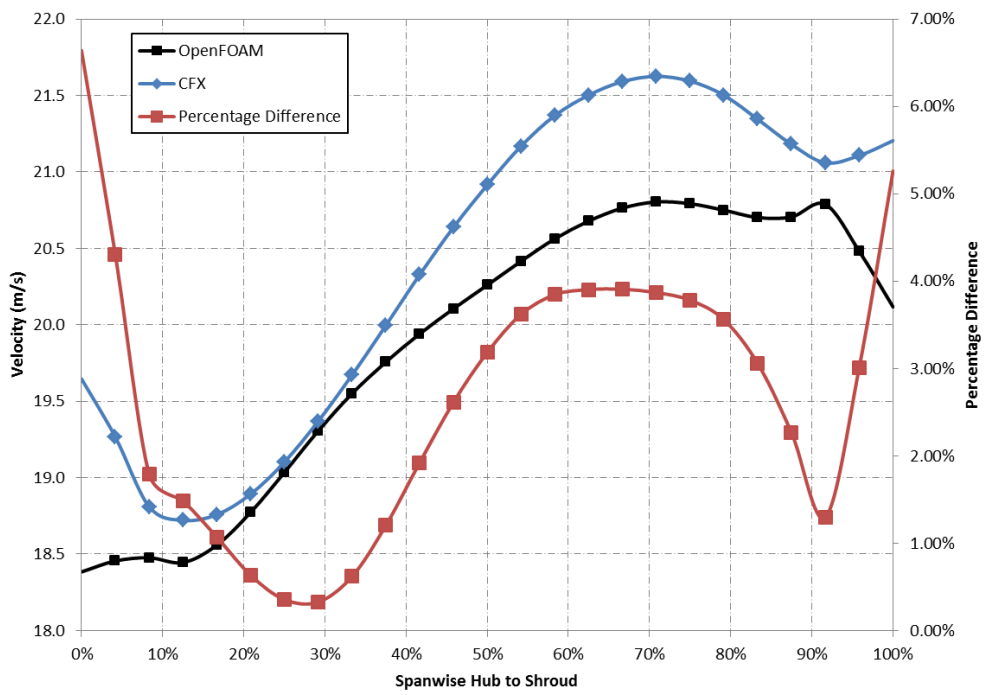
Figure A55 - Total Pressure in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 130% Q_{opt})



(a)

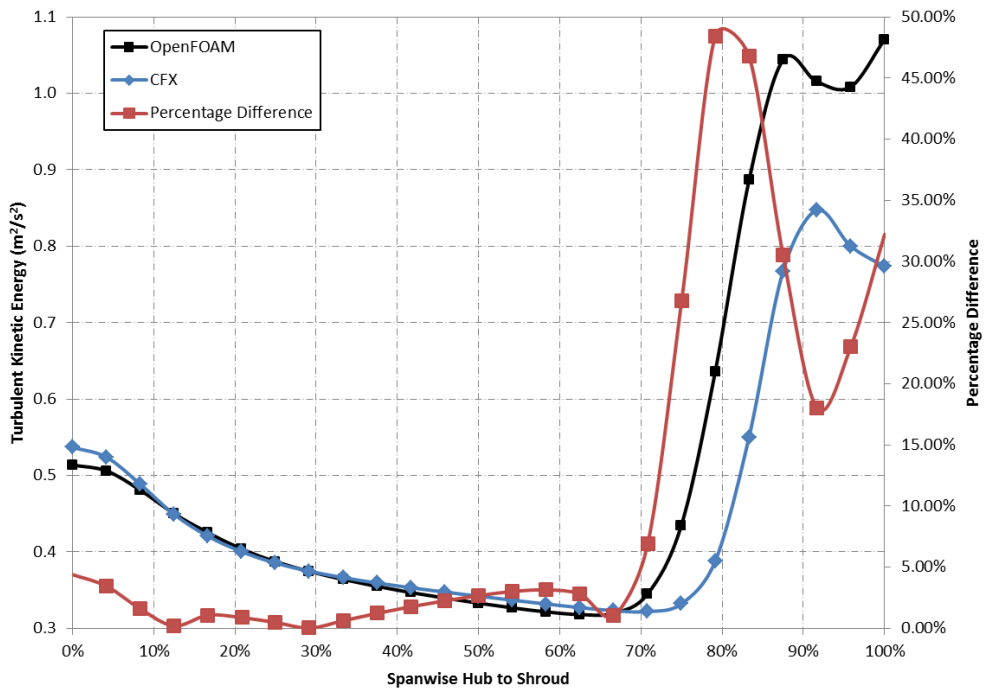


(b)

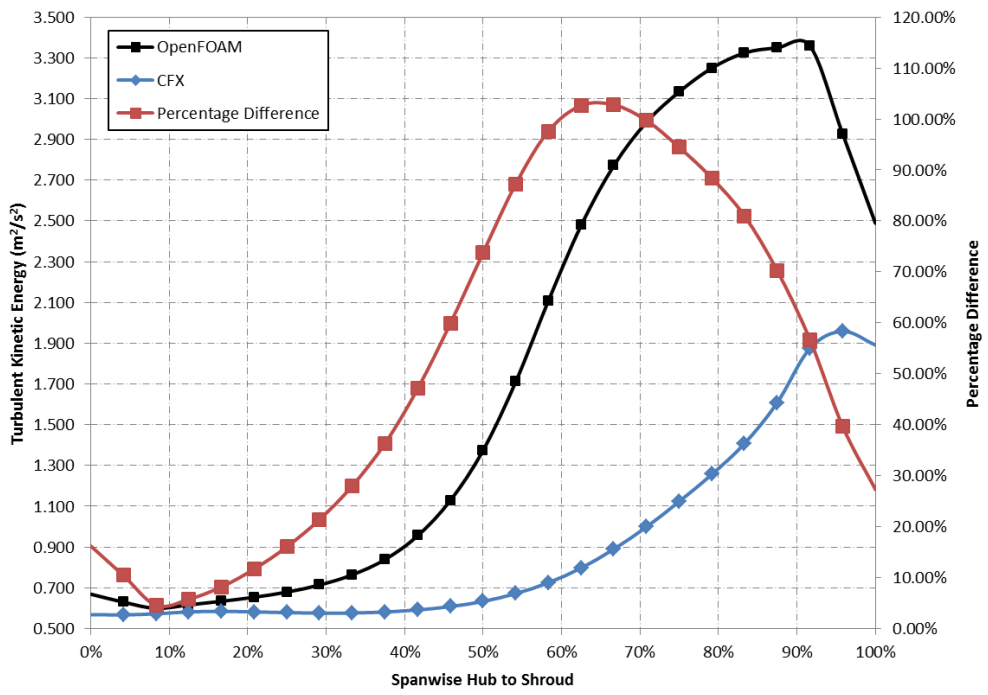


(c)

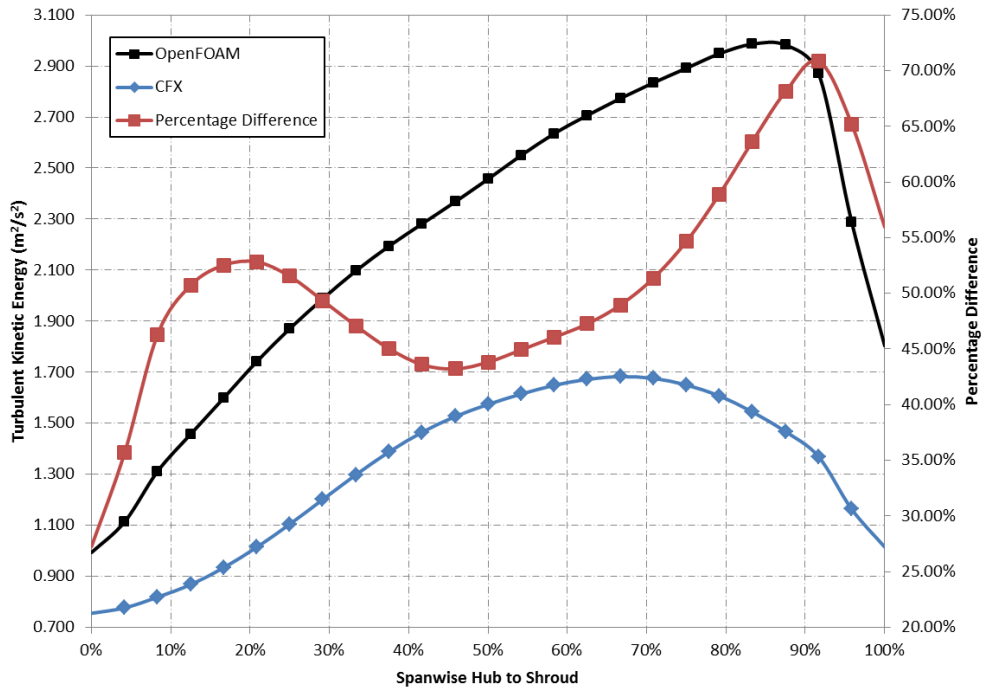
Figure A56 - Velocity in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85 % (Software Comparison – 130% Q_{opt})



(a)



(b)



(c)

Figure A57 - Turbulent Kinetic Energy in Stationary Frame across Impeller Span Hub to Shroud at (a) 15% (b) 50% and (c) 85% (Q_{opt}) (Software Comparison – 130% Q_{opt})

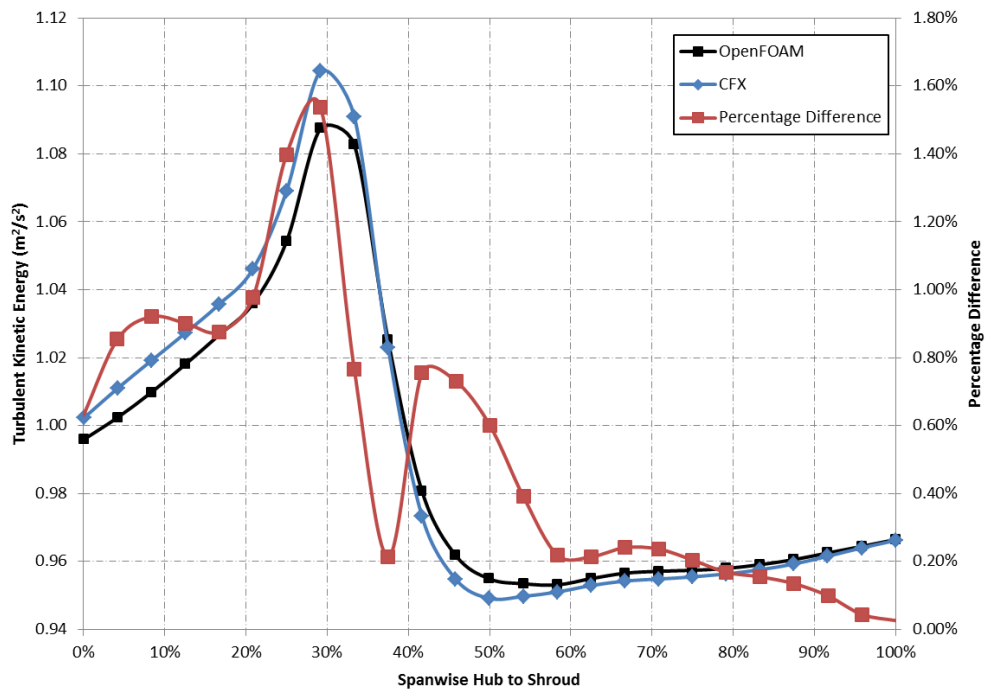


Figure A58 - Static Pressure across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 130% Q_{opt})

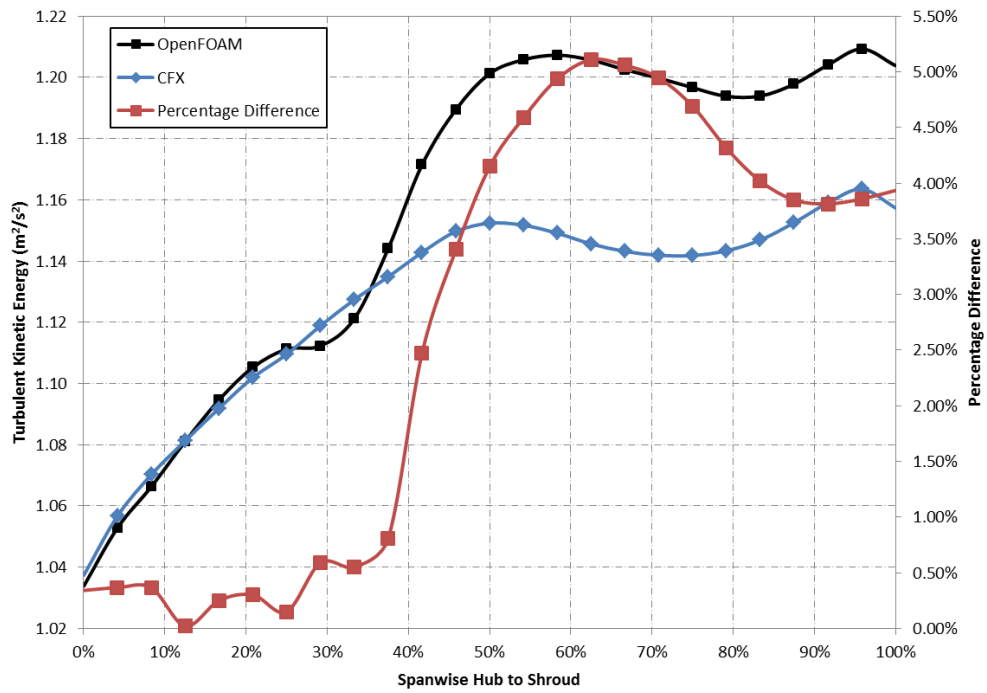


Figure A59 - Total Pressure across Line A-A (Figure 4.11) in Volute Casing (Software Comparison – 130% Q_{opt})

APPENDIX B – Turbulence Coefficients

$$F_2 = \tanh \left[\left[\max \left(\frac{2\sqrt{k}}{\beta^* \omega y}, \frac{500\nu}{y^2 \omega} \right) \right]^2 \right]$$

$$P_k = \min \left(\tau_{ij} \frac{\partial U_i}{\partial x_j}, 10\beta^* k \omega \right)$$

$$F_2 = \tanh \left[\left[\max \left(\frac{2\sqrt{k}}{\beta^* \omega y}, \frac{500\nu}{y^2 \omega} \right) \right]^2 \right]$$

$$F_1 = \tanh \left\{ \left\{ \min \left[\max \left(\frac{\sqrt{k}}{\beta^* \omega y}, \frac{500\nu}{y^2 \omega} \right), \frac{4\sigma_{\omega 2} k}{CD_{k\omega} y^2} \right] \right\}^4 \right\}$$

$$CD_{k\omega} = \max \left(2\rho\sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_i} \frac{\partial \omega}{\partial x_i}, 10^{-10} \right)$$

$$\phi = \phi_1 F_1 + \phi_1 (1 - F_1)$$

Constant	Value
α_1	$\frac{5}{9}$
α_2	0.44
β_1	$\frac{3}{40}$
β_2	0.0828
β^*	$\frac{9}{100}$

Constant	Value
σ_{k1}	0.85
σ_{k2}	1
$\sigma_{\omega 1}$	0.5
$\sigma_{\omega 2}$	0.856

APPENDIX C – OpenFOAM User Guide

Performing Turbomachinery Simulations

with OpenFOAM 2.1.x

User Guide

Version 1

2016

C1. About this Guide

OpenFOAM (Open Field Operation and Manipulation) is a free, open-source CFD software package produced by OpenCFD Ltd. Being open-source, it allows users to customise and extend its existing capability in order to meet their exact requirements. Its user base covers the majority of engineering and science in both commercial and academic environments.

The software has the capabilities to perform a wide range of simulations in parallel and includes parallelised pre- and post-processing tools as standard, utilising the power of the user's computer without the concern for licensing costs associated with commercial software packages.

OpenFOAM is structured modularly, in collections of numerical methods, meshing techniques and physical models, which make up a shared library. Executable applications are then created and linked to these shared libraries in order to solve the relevant case.

This manual aims to serve as an introduction to OpenFOAM, outlining the basics of the platform itself and installation instructions of OpenFOAM 2.1.x. It also provides an opportunity to become familiar in using the software by undertaking one of the tutorial cases built into OpenFOAM. Finally, and most relevant to the preceding body of work, the steps required to perform single-phase analyses of turbomachinery are outlined, examining a case study of a centrifugal pump. While the guide focuses on this particular example, the techniques employed can be utilised and adapted to suit your own requirements.

Whilst this guide is based on the OpenFOAM 2.1.x release, there are numerous versions available for download by users, each with their own updates and modifications from the previous version. Therefore it is always useful to be familiar with any changes that will impact relevant sections of the code.

C2. What is OpenFOAM?

In order to gain some understanding as to what OpenFOAM is and what its capabilities are, this chapter will outline some basic information relating to the language used, structure of the program and the numerical schemes available for solving problems.

OpenFOAM (Open Source Field Operation and Manipulation) is a C++ library used to create executables, referred to as applications. These applications can be subdivided into two categories: solvers, each designed to solve a specific problem in continuum mechanics; and utilities, which perform simple pre- and post-processing tasks, mainly involving data manipulation and calculations.

Each version of OpenFOAM has a number of both solvers and utilities, allowing it to solve a large range of problems from external flows to multiphase and combustion. A distinct advantage of OpenFOAM is that users can directly manipulate solvers and utilities as well as creating new ones, with their existing knowledge of the physical problem and programming techniques.

OpenFOAM not only has the capability to solve problems but has pre- and post-processing functionality built in as shown in Figure C1. However, should it be more practical or efficient to perform pre-processing and mesh generation or post-processing in third party commercial software, then this can also be accommodated.

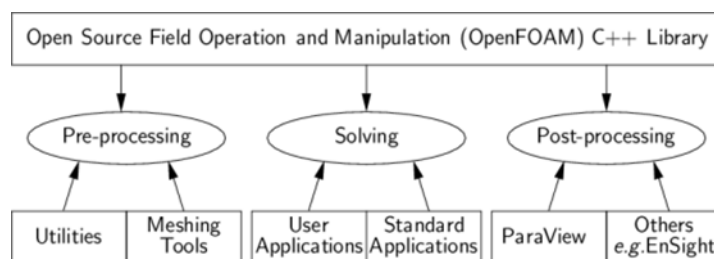


Figure C1 – OpenFOAM Structure

C2.1 OpenFOAM Programming Language

To gain an appreciation of how the OpenFOAM library works, it is useful to have a basic understanding of the C++ programming language used.

Firstly though, by examining language in general, it is clear that both verbal and mathematical languages are extremely useful in allowing complex terms and functions to be expressed efficiently and with relative ease.

As an example, the term 'velocity field' can be represented in a mathematical form by the letter 'U'. This type of representation can be taken a stage further when expressing more specific concepts such as the 'field of velocity magnitude' $|U|$. In such situations, it is clear that the language of mathematics is significantly more efficient than the verbal language.

Looking at continuum mechanics, problems are not represented in the language of bits, bytes and integers that the computer can read. Instead, they are usually translated from verbal language to partial differential equations in dimensions of space and time, constructed from the concepts of scalars, vectors, tensors, with the solutions involving matrices, solvers solution algorithms and discretisation procedures. This is done in OpenFOAM through C++ object-orientation.

C2.2 C++ Object Orientation

Object-oriented programming languages like C++ provide the mechanism, or *classes*, to declare types and operations that form the verbal and mathematical languages used in science and engineering.

Taking the earlier example of the 'velocity field', this can be represented in programming code by the symbol U, with the field of the velocity magnitude being represented by

$\text{mag}(U)$. As the velocity is a vector field, there will be a corresponding `vectorField` class associated with it. The velocity field U is considered to be an 'object' of the `vectorField`.

Having objects that represent physical objects in this manner is extremely useful but also very powerful. By using the 'class' structure, modifications can be made to the classes themselves, making the code easier to manage and new classes more straightforward to create.

C2.3 Representing Equations

Taking this concept a stage further, OpenFOAM solvers are written using these OpenFOAM classes and have a structure based upon the physical partial differential equations being solved, for example in C1.1.

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p \quad (\text{C1.1})$$

is represented as:

solve

```
(  
    fvm::ddt(rho, U)  
  + fvm::div(phi, U)  
  - fvm::laplacian(mu, U)  
  ==  
  - fvc::grad(p)  
);
```

It is interesting to note that the example above and many other aspects of OpenFOAM require a fully object-oriented language. Whilst many languages state that they are completely object-oriented, such as FORTRAN-90, they are not. However, C++ has all of the aspects required as well as the further advantage that it is typically used with standard specifications, enabling reliable compilers to create extremely efficient executables.

C2.4 Solver Codes

As the solvers in OpenFOAM are based on the C++ object-oriented concept, they can be understood and created by users without significant programming knowledge but instead a grasp of basic C++ syntax in conjunction with an appreciation of the physical principals and solution methods behind them. For this reason OpenFOAM can, depending on the situation, be used extensively without significant programming and/or solver manipulation.

C2.5 OpenFOAM Cases

In OpenFOAM, a case is essentially a 'working directory' in which all the files and subdirectories relating to a particular problem are stored. For example, the simple tutorial simulation regarding the flow in a cavity is called *cavity*.

These case directories can be stored anywhere but it is strongly recommended that they are located in the user's main project directory `$HOME/OpenFOAM/user-2.1.x/run`. This directory is set as the `$FOAM_RUN` environment variable as default which allows the user to move to this 'run' folder at any time simple by typing `run` in the command line.

The tutorials directory can be accessed in a similar manner by typing `tut` in the command line at any time. This directory contains a vast database of tutorials ranging across all areas of CFD. The technique is to establish which is the most relevant to the problem you are attempting to solve and then utilise it as a template from which you can construct and

C2.6 Case File Structure

Each case in OpenFOAM follows the same basic structure, with a minimum number of files required to run each application, with an example shown below in Figure C2.

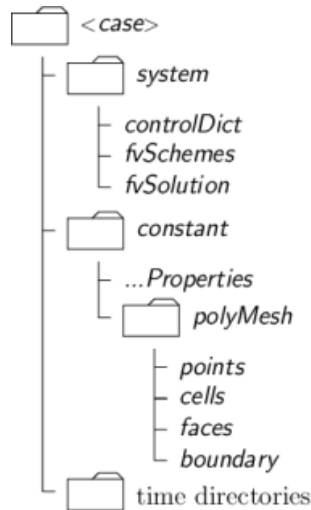


Figure C2 – Case Structure

There are three main directories that are always present: *constant*, *system* and a minimum of one time directory, for example *0*.

The *constant* directory contains a subdirectory *polyMesh* that defines the entire case mesh as well as files describing the physical properties required for the particular problem such as *turbulenceProperties*.

The *system* directory contains the settings for parameters associated with the solution procedure and always includes a minimum of three files; *controlDict*, *fvSchemes* and *fvSolution*. The *controlDict* is used to control various parameters such as the start/end times, time step size, maximum Courant number as well as functions that can be used for data output. The *fvSchemes* file specifies the discretisation schemes to be used during the solution process. The *fvSolution* file defines the equation solvers, tolerances and algorithm controls for the simulation.

The *time* directory contains all of the individual files containing the data of each type of field being solved in any given simulation such as pressure (*p*), velocity (*U*) and specific turbulent kinetic energy (*k*). This information is specified by the user as initial values and boundary conditions when constructing the simulation and written to file by OpenFOAM during the solution process.

C2.7 Dimensional Units

As with any continuum mechanics software package, properties are defined by using a specific type of unit, such as volume in cubic metres (m^3) or pressure in Pascals ($\text{kgm}^{-1}\text{s}^{-2}$). In OpenFOAM, the units relating to each type of field being calculated during a simulation are specified by using a *dimensionSet*, with the format based on the base units shown in Table C1.

Table C1 – Dimensional Units

No	Property	SI Unit
1	Mass	Kilogram (kg)
2	Length	Metre (m)
3	Time	Second (s)
4	Temperature	Kelvin (K)
5	Quality	Kilogram-Mole (kgmol)
6	Current	Ampere (A)
7	Luminous Intensity	Candela (ca)

Each of the values in Table C1 corresponds to the power of each of these base units, so for a velocity term (ms^{-1}), the dimensions would be specified by these seven scalars delimited by square brackets as $[0\ 1\ -1\ 0\ 0\ 0\ 0]$.

C2.8 Numerical Schemes

One of the most important ‘dictionaries’ in OpenFOAM is *fvSchemes*, located in the *system* directory. It specifies the numerical schemes for terms such as derivatives in equations relevant to the application being utilised.

OpenFOAM aims to offer an unrestricted choice to the user and includes a variety of terms that are assigned numerical schemes in this dictionary, such as divergence ($\nabla \cdot$) and time schemes. The main sub-divisions of numerical schemes are shown below in Table C2 within which all the gradients of each type are stored. For example, *divSchemes* contains all the divergence terms such as *div(phi,U)*.

Table C2 – Numerical Schemes

Keyword	Category of Mathematical Terms
interpolationSchemes	Point-to-point interpolation of values
snGradSchemes	Component of gradient normal to a cell face
gradSchemes	Gradient ∇
divSchemes	Divergence $\nabla \cdot$
laplacianSchemes	Laplacian ∇^2
timeScheme	First and second time derivatives $\partial / \partial t, \partial^2 / \partial^2 t$
fluxRequired	Fields which required the generation of a flux

Furthermore, the derivative terms have a choice of discretisation practice, with standard Gaussian finite volume integration being the standard selection. There is also the ability to define the interpolation scheme, with specific schemes being designed for particular divergence terms.

C2.9 Further Information

Whilst this section aims to provide some basic information regarding the basic structure of OpenFOAM, more in-depth material can be found at www.openfoam.org.

C3. Installing and Running OpenFOAM in Ubuntu (Linux)

In order to run OpenFOAM, you must install a Linux/UNIX operating system (OS). Ubuntu is used in this instance and can be installed in one of two ways:

1. Install Ubuntu directly onto your hard drive, side-by-side a Windows OS, creating a partition for this operating system and running it from the boot screen.
2. Create a 'virtual machine' on your Windows PC which will allow the 'guest' Linux OS to be run 'within' your existing setup.

C3.1 Installing Ubuntu Side-by-Side Windows OS

This method of installation will allow you to select either Windows OS or Linux when turning on your PC and is the most effective in terms of performance.

The first thing to do is back up any important files to avoid any potential issues during installation.

Now, visit <http://www.ubuntu.com/download/desktop> and select the 32-bit or 64-bit version depending on your computer. You can determine which version is relevant to you by going to Start/Programs/Accessories/System Tools/System Information. If the 'System Type' is 'X-86-based PC' then you have a 32-bit computer. If this shows 'X-64-based PC' then it is a 64-bit computer.

Click 'Start Download' – as this OS is only around 700MB it shouldn't take too long.

Once you have downloaded the file, you have to 'burn' the installation software onto a CD (or USB stick – an install guide is on the Ubuntu website). With a blank disc in the CD drive, right click the Ubuntu installation file you downloaded and select 'Burn Disk Image'.

When the burning has finished, restart your computer, keeping the disk in the CD/DVD drive. You will now have to change the boot sequence of your PC to allow the CD/DVD drive to be accessed before the hard-disk drive (HDD). This can be done by hitting the relevant function key (F1, F2, etc.) as the computer boots up. Each computer has a different 'F' key, so try both. Search the set-up options in order to move the CD/DVD drive to be first in the boot sequence.

Follow the Ubuntu on-screen instructions for installing Linux side-by-side with Windows OS.

WARNING: BE CAREFUL NOT TO OVERWRITE YOUR WINDOWS OS!

When restarting your computer, alter the boot-sequence back to the original configuration of HDD first. This will allow you to select which OS you wish to use when booting.

You are now ready to use Linux! This may be a good time to learn (or re-learn) some basic commands that will be useful when using OpenFOAM (e.g. www.ee.surrey.ac.uk/Teaching/Unix).

C3.2 Installing Ubuntu as a Virtual Machine

If you would prefer to run Ubuntu as a virtual machine on your Windows OS, or are unable to install Linux side-by-side due to IT restrictions, visit <http://www.openfoam.com/resources/windows.php> where you will find a detailed step-by-step guide on creating a virtual machine with Oracle VirtualBox. Please note that this is not the most efficient method of running Ubuntu and therefore performance may be compromised.

C3.3 Installing OpenFOAM-2.1.x

Whilst there are many version of OpenFOAM that can be installed, the following outlines how to install the OpenFOAM 2.1.x source code, including fixes for all submitted bugs in this release. If you would rather install a pre-packaged or specific older or newer version, please visit <http://www.openfoam.org/download/>.

Note: If you are copying code from this document or the internet, you can use CTRL+C to copy and CTRL+SHIFT+V to paste in the terminal. You can also use the mouse if you prefer.

Once you have started a new session in Ubuntu, open the terminal window by clicking on the icon shown in Figure C3 from the menu bar or going to Applications/Accessories/Terminal.



Figure C3 – Terminal Icon

By default, this should position you in your home/username directory. You can find what directory you are currently in by typing *pwd* (print working directory) followed by the return key. If your username is 'user' then you should be in the directory:

/home/user

C3.3.1 Git Software

The sources in this instance are obtained using the 'Git' open source version control system, which can be installed by executing the command:

```
sudo apt-get install git-core
```

The sudo requires 'super-user' privileges and your password will be required at this stage followed by the return key (your keystrokes will not show on the screen).

C3.3.2 Installing the Source Code – OpenFOAM-2.1.x

In order to install OpenFOAM 2.1.x, the user needs to select the location where the files will be unpacked and become the installation directory of OpenFOAM. This can be done by creating an 'OpenFOAM' folder in your home directory:

```
mkdir OpenFOAM
```

Move into the newly created OpenFOAM directory:

```
cd OpenFOAM
```

The OpenFOAM-2.1.x repository can now be obtained by typing the following:

```
git clone git://github.com/OpenFOAM/OpenFOAM-2.1.x.git
```

This can then be updated to the latest version available by using:

```
cd OpenFOAM-2.1.x
```

```
git pull
```

C3.3.3 Installing the Source Code – Third Party

The third party source-pack containing the additional software required for OpenFOAM-2.1.x can be obtained from SourceForge at http://downloads.sourceforge.net/foam/ThirdParty-2.1.1.tgz?use_mirror=mesh

This will download the file to your 'Downloads' folder, meaning that you will have to copy the it over to your OpenFoam-2.1.x folder that you should currently be in using:

```
cp -r ~/Downloads/ThirdParty-2.1.1.tgz
```

Now, unpack the ThirdParty-2.1.x.tgz file and change its name:

```
tar -xzf ThirdParty-2.1.1.tgz
```

```
mv ThirdParty-2.1.1 ThirdParty-2.1.x
```

Install additional packages required by using the following two commands:

```
sudo apt-get install build-essential flex bison cmake zlib1g-dev qt4-dev-tools libqt4-dev gnuplot libreadline-dev libncurses-dev libxt-dev
```

```
sudo apt-get install libscotch-dev libopenmpi-dev
```

C3.3.4 Environment Variables

The next step is to update the environment variable settings (.bashrc) contained in \$HOME/OpenFOAM/OpenFOAM-2.1.x/etc. Firstly, move into your home directory /home/user:

```
cd $HOME
```

Create a copy of the existing .bashrc file to ensure there is a backup version:

```
cp .bashrc .bashrc_old
```

Now edit the .bashrc file using 'gedit', the application which you will use for editing from now on:

```
gedit $HOME/.bashrc
```

Add the following line to the end of the existing code:

```
source $HOME/OpenFOAM/OpenFOAM-2.1.x/etc/bashrc
```

Save the file and close gedit.

Re-open the file using gedit once again, checking to see that the file has been altered and is formatted correctly.

You can now activate this new .bashrc file (called 'sourcing') by typing:

```
source $HOME/.bashrc
```

in the current terminal window.

To check the system is now ready to build the sources, type:

```
foamSystemCheck
```

Hopefully there are no issues at this point however issues may arise with the version of 'gcc' on your system. Gcc-4.4 is recommended and you can check by typing:

```
gcc --version
```

C3.3.5 Building the Sources

Build the sources by going to the top-level source directory by typing:

```
cd $WM_PROJECT_DIR
```

and execute the following command

```
./Allwmake
```


NOTE: This will take a considerable length of time so it is advised that you leave the computer. More information regarding compiling and the wmake commands can be found at <http://www.openfoam.org/download/git.php>

Compile the post-processing tool Paraview 3.12.0 (updated versions are released periodically) using the following three commands:

```
cd $WM_THIRD_PARTY_DIR
```

```
sed -i -e 's/ClearAndSelect = Clear | Select/ClearAndSelect = static_cast<int>(Clear) | static_cast<int>(Select)/' ParaView-3.12.0/Qt/Core/pqServerManagerSelectionModel.h
```

```
./makeParaView
```

Next, compile to PV3blockMeshReader and PV3FoamReader ParaView plugins:

```
cd $FOAM_UTILITIES/postProcessing/graphics/PV3Readers
```

```
wmSET
```

```
./Allwclean
```

```
./Allwmake
```

In order to confirm that the installation has been successful, execute

```
foamInstallationTest
```

To confirm if the installation has been successful, test the icoFoam application by typing:

```
icoFoam -help
```

A 'usage' should appear, confirming that the installation and configuration is complete.

C3.3.6 Important Directories

The next step is to create a project directory inside \$HOME/OpenFOAM called user-2.1.x and a 'run' directory within it. This is where you will keep your cases being investigated and it can be created by typing:

```
mkdir -p $FOAM_RUN
```

By creating these folder in this way, you can now move to your 'run' directory by typing 'run' from any window. This is exactly what you will do when opening the terminal window with the aim of starting or continuing an OpenFOAM case.

You also want to copy the tutorial files (which form the basis of all cases in OpenFOAM) by typing the following command:

```
cp -r $FOAM_TUTORIALS $FOAM_RUN
```

C3.3.7 transientSimpleDyMFoam

This additional solver is used in the transient analyses of the turbomachinery. It can be obtained and compiled by using the following commands:

```
cd ~/OpenFOAM/OpenFOAM-2.1.x/applications/solvers/incompressible  
  
svn checkout http://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-  
extend/trunk/Breeder_1.5/OSIG/TurboMachinery/applications/solvers/incompressi-  
ble/transientSimpleDyMFoam  
  
cd transientSimpleDyMFoam
```

```
wmake
```

C3.3.8 turboPerformance

This utility allows typical turbomachinery parameters such as generated head and hydraulic efficiency to be tracked over time by printing them to the screen. It can be downloaded by typing:

```
run
```

```
svn checkout http://openfoam-extend.svn.sourceforge.net/viewvc/openfoam-extend/trunk/Breeder_2.0/OSIG/TurboMachinery/src/turboPerformance/
```

Then, unzip the files:

```
tar xzf turboPerformance.tar.gz
```

before moving and compiling them by moving into the following location:

```
cd ~/OpenFOAM/OpenFOAM-2.1.x
```

Now, type the following command in order to compile the utility:

```
wmake libso turboPerformance
```

C3.3.9 Gnuplot

In order to plot the convergence of parameters such as those from the turboPerformance utility, you require the open-source graphics package, Gnuplot. This can be installed by typing the following:

```
sudo su-
```

```
apt-get install gnuplot-x11
```

```
Exit
```

C3.3.10 swak4Foam

One final package to install is swak4Foam which allows the user to track properties such as mass flow and pressure at boundaries within simulations. It can be obtained by using the following one of the various options available at <https://openfoamwiki.net/index.php/Installation/swak4Foam/Downloading>

In order to use this utility with the OpenFoam-2.1.x release, the software entitled 'bison' that was installed earlier has to be changed to an older version which can be done by ensuring you are in the correct location and downloading the file:

```
cd ~/OpenFOAM/user-2.1.x
```

```
http://ftp.gnu.org/gnu/bison/bison-2.4.1.tar.bz2
```

Now, unzip the file:

```
tar -xzf bison
```

Finally, move into the folder and compile:

```
cd swak4Foam
```

```
make
```

```
make install
```

C4. Tutorial – Lid-driven Cavity Flow Case

In order to check that OpenFOAM is working correctly, and to allow an opportunity to experience the setup, this simple ‘icoFoam’ tutorial will be used, which means ‘Incompressible Foam’. The ‘cavity’ case relates to a transient, laminar flow in a lid driven cavity as seen in Figure C4.

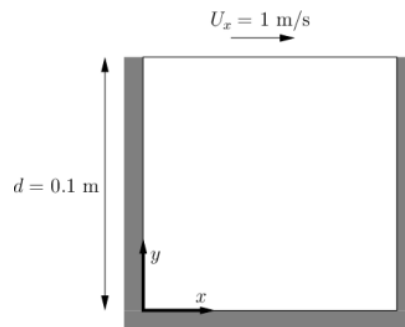


Figure C4 – icoFoam Case Study Description

This is how OpenFOAM should be used, establishing which case is most relevant to your own problem and using it as a template to reflect this.

C4.1 Case Setup

In order to move directly into the tutorials directory, type:

```
tut
```

Now, change into the appropriate directory in order to run the cavity case:

```
cd incompressible/icoFoam/cavity
```

To check that you are in the correct folder, type *pwd*. The terminal should show that you are in the following directory:

```
/home/'user'/OpenFOAM/'user'-2.1.x/run/tutorials/incompressible/icoFoam/cavity
```

In order to view the contents of the cavity case, type:

```
ls
```

This should show three working directories: *0*, *constant* and *system*. The *0* time directory is where all boundary conditions for the 'fields' within a case are set, in this case velocity vector (U) and scalar pressure (p). As the cavity case is run, further time directories are added at 0.1, 0.2, 0.3, 0.4 and 0.5, with the final time step assuming steady state conditions have been reached.

In this cavity case, and any other case for that matter, the time-step properties are set in a 'dictionary'. These dictionaries are used to set boundary conditions, control parameters and properties and set time-steps and much more.

The dictionary most commonly used is the *controlDict* which is located within the *system* directory i.e. *system/controlDict*. This may be a good time to view the contents of the *0*, *constant* and *system* directories using 'gedit', for example:

```
gedit system/controlDict
```

Note: use '*cd <directory>*' to change into a directory and '*cd ..*' to move back up the directory tree.

C4.2 Running the Case

At this point, you should be located within the *cavity* case directory.

In order to use the mesh pre-processor and format the mesh, type:

```
blockMesh
```

Now, to solve the case, type:

icoFoam

Next, type:

ls

in order to view the files in the *cavity* directory once again. This will highlight the new time directories that have been created. Within each of these are the solution data for the p and U fields for the time steps 0.1 to 0.5. This then allows each time step to be post-processed.

C4.3 Post-processing in ParaView

Post-processing in this case will be done using the open-source tool paraView, part of the installation that was performed earlier. It will allow visualisation of the velocity (U) and pressure (p) fields at each of the new time steps.

In order to open paraView, type:

paraFoam &

NOTE: The use inclusion of the ampersand will open a new window for paraView but continue to allow access to the terminal.

Once open, the next step is to view alter which time step you wish to view the data for. To view the data for the final time step, click the 'last frame' button , located at the top of the screen on the toolbar as shown in Figure C5.

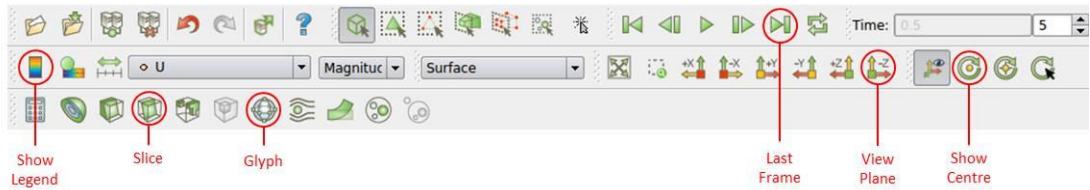


Figure C5 – ParaView Toolbar

Next, select the properties that you wish to load data for, in this case velocity (U) and pressure (p) as shown in Figure C6.

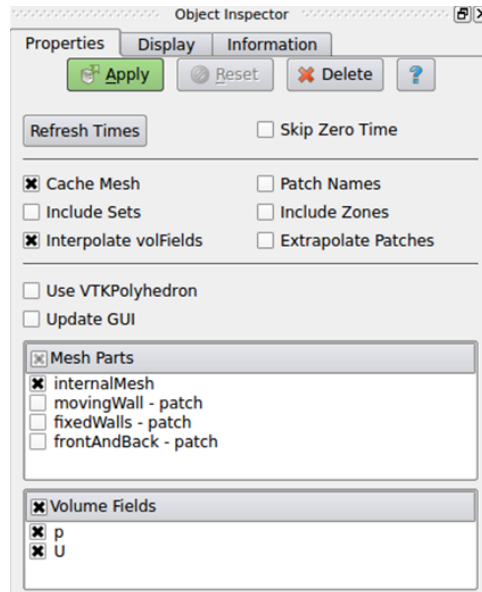


Figure C6 – Object Inspector

And click 'apply'.

Now, in order to view the velocity field, U for the final time step, select 'U' from the 'Solid Color' drop down menu as shown in Figure C7.

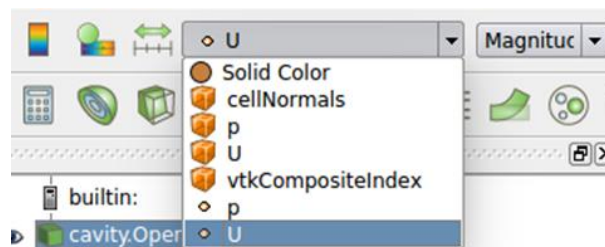


Figure C7 – Field Viewer Selection

By changing this, the following plot should be shown in Figure C8.

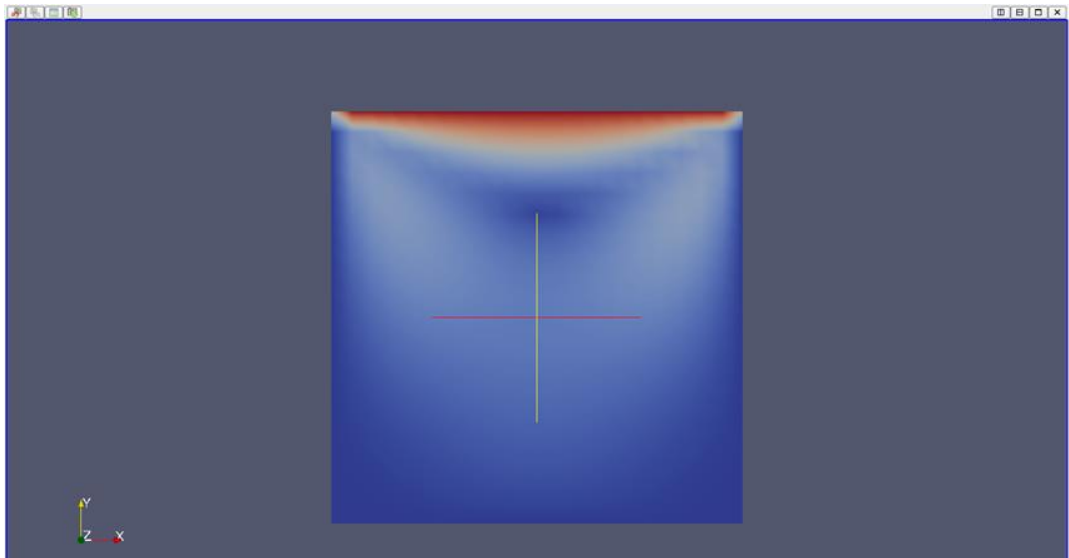


Figure C8 – icoFoam Contour of Velocity

As default, the colour scale is from blue to red. In order to change this, click the ‘Display’ tab in the ‘Object Inspector’ before selecting ‘Edit Color Map/Choose Preset’ and choosing the desired range from the editor that opens as seen in Figure C9.

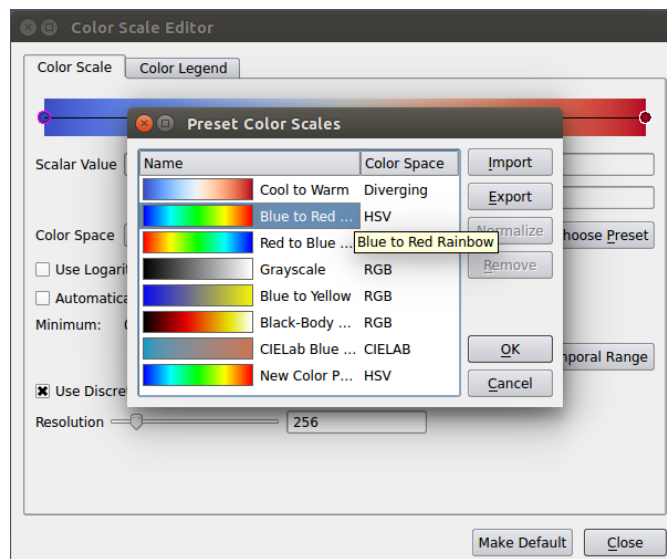


Figure C9 – ParaView Color Scale Editor

In order to view the legend on the screen, select ‘Color Legend’ at the top left hand corner of the screen as seen in Figure C5. This can be moved to a different position on the screen by simply dragging it. To remove the central cross from the screen, click the ‘show-center’ button (Figure C5).

Moving around the environment is straightforward, using the left mouse button to rotate, the right button to zoom and the middle button to pan.

As with other standard packages, you can select which fixed view to use.

To return to the initial view, click on the '-z' coordinate button (Figure C5).

To show the velocity vectors, which are called 'glyphs' in paraView, click the glyph button (Figure C5) followed by 'apply'. Once again, change the 'Solid Color' dropdown menu to 'U'. This should present both vectors and contours of velocity on the same plot as shown in Figure C10.

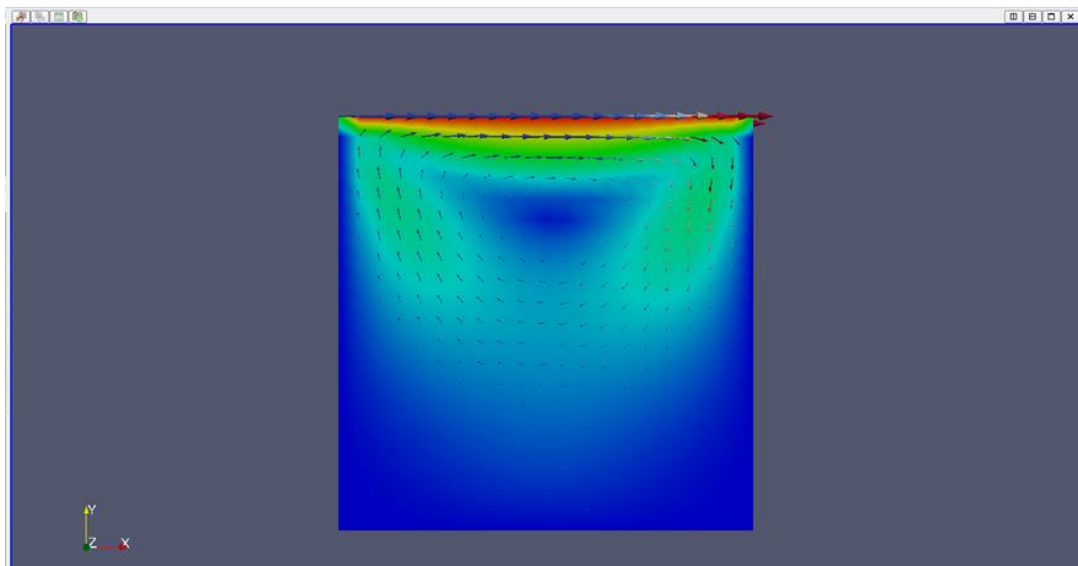


Figure C10 – icoFoam Contours and Vectors of Velocity

Notice on the left hand menu bar that the velocity vector and contour plots created have appeared. You can toggle these on/off by clicking on the eye icon.

Next, create a 2D section of the results by selecting the slice icon (Figure C5) for the z-normal plane and click 'apply'

Combining the last two steps, hide the glyph (vector) plot by selecting the eye symbol next to 'Glyph 1' to show the velocity contour of the slice as seen in Figure C11.

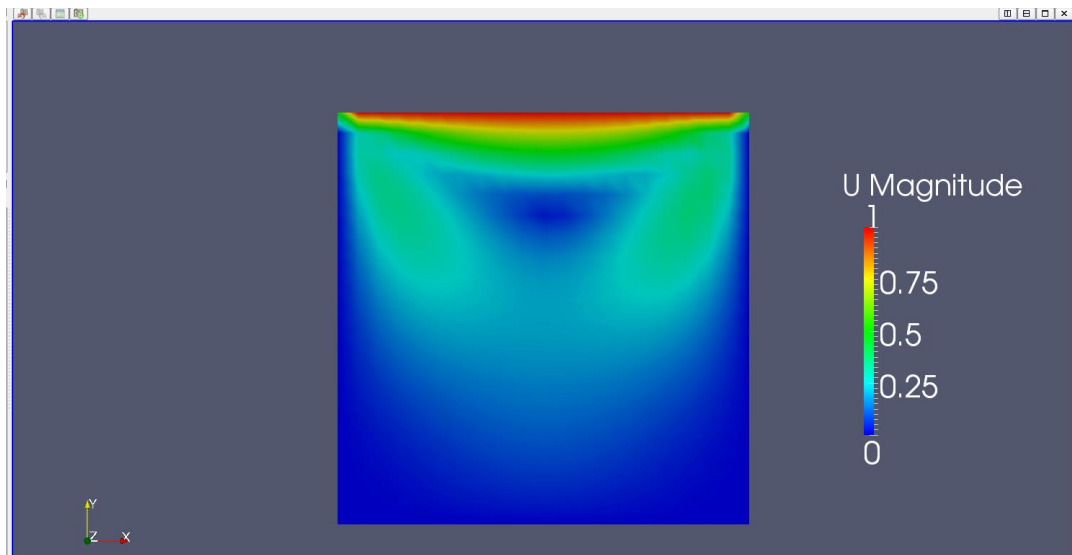


Figure C11 – icoFoam Slice of Contours and Vectors of Velocity

As with similar packages, you can save images in a number of formats including .jpg, .bmp and .png by selecting:

File>Save Screenshot>ok

or by using Ubuntu's built in screenshot facility by selecting:

Applications>Accessories>Take Screenshot

In order to become familiar with the system, plot the pressure field in the same manner.

Now that you have completed the first tutorial case, we will move onto the turbomachinery applications of OpenFOAM, specifically modelling a centrifugal pump.

C5. Centrifugal Pump Case Study

This section focuses on using OpenFOAM to model turbomachinery, with the specific example case of centrifugal pump. It should be used as a guide to enable you to perform simulations on your own specific case.

The steps required to set up and perform single-phase frozen rotor and transient simulations are outlined. Whilst the focus surrounds modelling the pump at best efficiency flow of 235 l/s (@ 1100rpm), information regarding how to use the results for off-design flow conditions, in order to generate a predicted performance curve, is also included.

C5.1 Converting the Meshes

Whilst meshes can be created in OpenFOAM, they are also often created in other platforms such as ANSYS Meshing or ICEM. In this test case, the mesh was originally created in ICEM and exported as a single .msh file. Whilst the meshes can be imported separately, this example shows a .msh file which contained the three component meshes for the parts shown in Figure C12.



Figure C12 – Centrifugal Pump Case Study

Looking at this in more detail, the three components are defined as separate domains/zones in the .msh file and also contain a number of named selections as shown in Table C3. The inclusion of the named selection at this early stage is extremely important as it allows the user to define boundary conditions and monitor properties and expressions based on surface selections such as the blades of the impeller.

Table C3 – Centrifugal Pump Case Study Detailed Setup

Domain/Zone	Named Selection/Surface Group
throatbushzone	Throatbush_Inlet
	Throatbush_Walls
	Throatbush_Outlet
impellerzone	Impeller_Inlet
	Impeller_Blades
	Impeller_Hub
	Impeller_Hub_Side
	Impeller_Shroud
	Impeller_Shroud_Side
	Impeller_Outlet
volutezone	Volute_Inlet
	Volute_Walls
	Volute_Walls_Extension
	Volute_Outlet

The first stage is to move into your 'run' folder by typing:

run

in the command line, followed by return.

Next, create a new directory for your case called 'pump'. This is the main working directory where everything associated with the case will be stored.

```
mkdir pump
```

Move into this folder using:

```
cd pump
```

Now you want to create dummy *0*, *constant* and *system* directories by copying an existing tutorial case to the *pump* directory:

```
cp -r ~/OpenFOAM/'user'-2.1.x/run/tutorials/incompressible/simpleFoam/motorBike.
```

Next, rename the *motorBike* folder to represent the pump:

```
mv motorBike pumpMesh
```

Next, in order for the meshing conversion to work effectively, you need to move the *0.org* directory in this folder to be called *0*:

```
mv pumpMesh/0.org pumpMesh/0
```

Now, move to the *pumpMesh* sub-directory and copy the mesh file (called *pump.msh* in this example) from your external device by using the 'home folder' on the Ubuntu side bar (you can obviously use the terminal to copy across if you are comfortable doing so).

Select your external device on the left task bar and locate your files.

Select the mesh relevant to the case and 'copy' the file.

Click 'Home' then navigate through the folders *OpenFOAM/user-2.1.x/run/pump/pumpMesh* and paste the files here.

Next, convert the mesh to a format that OpenFOAM can utilise. In order to do this, the *fluent3DMeshToFoam* convertor is used:

```
fluent3DMeshToFoam pump.msh
```

Now, return to the main pump directory:

```
cd ..
```

You should be left with one folder in your *pump* directory called *pumpMesh* in which you will have the complete details of the pump in the *constant/polyMesh* sub-directory.

Check this using '*ls*' and then move into *pumpMesh/constant/polyMesh*:

```
cd pumpMesh/constant/polyMesh
```

Return to the main pump directory:

```
cd ../../..
```

You can clean this up again by moving the *polyMesh* folder into your main directory and then deleting the remaining *pumpMesh* folder:

```
mv pumpMesh/constant/polyMesh .
```

```
rm -r
```

Before going forward, the boundaries created in your original meshing tool will have to be edited as they are all set to 'wall' by default. Modify the existing to reflect the example below by typing:

```
gedit polyMesh/boundary
```

and alter the file to follow Figure C13.

```

1 |-----* C++ *-----|
2 |=====|
3 | \ \ \ \ \ | F i e l d | | OpenFOAM: The Open Source CFD Toolbox
4 | \ \ \ \ \ | O p e r a t i o n | | Version: 2.1.x
5 | \ \ \ \ \ | A n d | | Web: www.OpenFOAM.org
6 | \ \ \ \ \ | M a n i p u l a t i o n | |
7 |-----*-----|
8 FoamFile
9 {
10   version      2.0;
11   format       asclI;
12   class        polyBoundaryMesh;
13   location     "constant/polyMesh";
14   object       boundary;
15 }
16 //-----*-----//
17
18 14
19 (
20
21 //-----*THROATBUSH*-----//
22
23   Throatbush_Inlet
24   {
25     type        patch;
26     nFaces      1122;
27     startFace   4484289;
28   }
29
30   Throatbush_Walls
31   {
32     type        wall;
33     nFaces      7539;
34     startFace   4562954;
35   }
36
37   Throatbush_Outlet
38   {
39     type        cyclicAMI;
40     nFaces      1842;
41     startFace   4571765;
42     matchTolerance 0.0001;
43     neighbourPatch Impeller_Inlet;
44     transform   noOrdering;
45   }
46
47   Impeller_Inlet
48   {
49     type        cyclicAMI;
50     nFaces      1272;
51     startFace   4570493;
52     matchTolerance 0.0001;
53     neighbourPatch Throatbush_Outlet;
54     transform   noOrdering;
55   }
56
57   Impeller_Hub
58   {
59     type        wall;
60     nFaces      16359;
61     startFace   4406011;
62   }
63
64   Impeller_Hub_Side
65   {
66     type        wall;
67     nFaces      2028;
68     startFace   4422370;
69   }
70
71   Impeller_Shroud
72   {
73     type        wall;
74     nFaces      21746;
75     startFace   4424398;
76   }
77
78   Impeller_Shroud_Side
79   {
80     type        wall;
81     nFaces      2015;
82     startFace   4446146;
83   }
84
85   Impeller_Blades
86   {
87     type        wall;
88     nFaces      65522;
89     startFace   4458688;
90   }
91
92   Impeller_Outlet
93   {
94     type        cyclicAMI;
95     nFaces      10527;
96     startFace   4446101;
97     matchTolerance 0.0001;
98     neighbourPatch Volute_Inlet;
99     transform   noOrdering;
100  }
101
102 //-----*VOLUTE*-----//
103
104
105   Volute_Inlet
106   {
107     type        cyclicAMI;
108     nFaces      9213;
109     startFace   4524210;
110     matchTolerance 0.0001;
111     neighbourPatch Impeller_Outlet;
112     transform   noOrdering;
113   }
114
115   Volute_Walls
116   {
117     type        wall;
118     nFaces      29531;
119     startFace   4533423;
120   }
121
122   Volute_Walls_Extenson
123   {
124     type        wall;
125     nFaces      3578;
126     startFace   4400631;
127   }
128
129   Volute_Outlet
130   {
131     type        patch;
132     nFaces      680;
133     startFace   4405331;
134   }
135
136 )
137 //-----*-----//

```

Figure C13 – boundary File

Note that the components relating to the interfaces at rotating regions are defined as *cyclicAMI* (Arbitrary Mesh Interface) and paired with the relevant component with which it is interfaced. This type of interface allows modelling of mesh domains that are disconnected but adjacent.

After all of this work, you are now left with a simple *polyMesh* folder. This now allows the setup of frozen rotor simulation to begin.

C5.2 Performing Simulations

This section deals with the construction, analysis and running of the centrifugal pump case, from initial setup through to the steady state and transient simulations as well as post-processing and data analysis.

The boundary conditions will be set as velocity inlet and static pressure outlet.

C5.2.1 Organising the Simulations

With the meshing procedure complete, the next stage is to set up and run simulations from your main working director, *pump*. One of the best ways to set this up is to contain your different types of simulations relating to the single pump in the same folder, such as in Figure C14. This format will be adopted in the sections that follow.

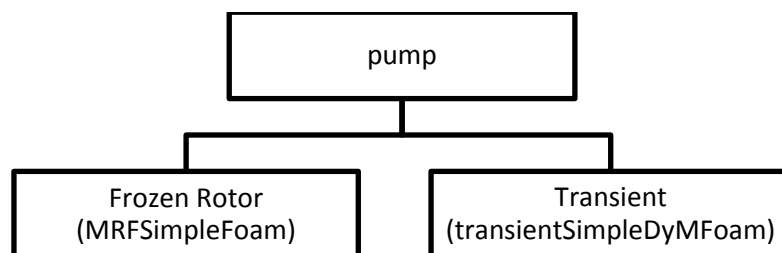


Figure C14 – Initial Case File Organisation

C5.2.2 MRFSimpleFoam (Steady State) Simulation

The first simulation required to be set up is the steady state analysis which is performed using the *MRFSimpleFoam* solver.

Ensuring that you are in your main *pump* directory, copy the *MRFSimpleFoam* tutorial case over from one of the tutorial cases:

```
cp -r ../tutorials/incompressible/MRFSimpleFoam/mixerVessel2D .
```

Typing *'ls'* will show the new contents of the main working directory as shown in Figure C15.

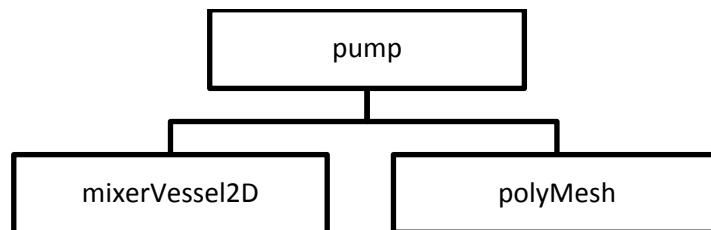


Figure C15 – MRFSimpleFoam Case – Organisation During Setup

Rename the *mixerVessel2D* directory as this is where all the data related to your steady state simulations will be held:

```
mv mixerVessel2D MRFSimpleFoam
```

Next, move into the newly named *MRFSimpleFoam* folder using *'cd'* and familiarise yourself with the contents of each folder, for example *'ls constant'*.

Following this, return to the *MRFSimpleFoam* directory where you can now begin to remove items that are not required in this simulation, such as the *Allrun* file which is a script that can be executed to run the full tutorial case in one go, and *makeMesh*:

```
rm -r Allrun makeMesh 0/nut
```

Now, still in the *MRFSimpleFoam* directory, delete the existing *polyMesh* folder within the 'constant' directory and replace with the one you created earlier for the fully meshed pump:

```
rm -r constant/polyMesh
```

```
mv ../polyMesh constant
```

Now, copy across a *turbulenceProperties* file from the tutorials into the *constant* directory:

```
cp -r ~/OpenFOAM/'user'-
```

```
2.1.x/run/tutorials/incompressible/pimpleDyMFoam/propeller/constant/turbulenceProperti  
es constant
```

In terms of organising the file structure, the final step is to rename the *epsilon* file in the *0* time directory as *omega*, as we will be using the *kOmega-SST* model in this case. From the *MRFSimpleFoam* directory:

```
mv 0/epsilon 0/omega
```

This completes the case structure and you will now begin to edit this file as well as several others in the next section.

To summarise, the stages to this point should yield a working directory in the structure of that seen in Figure C16.

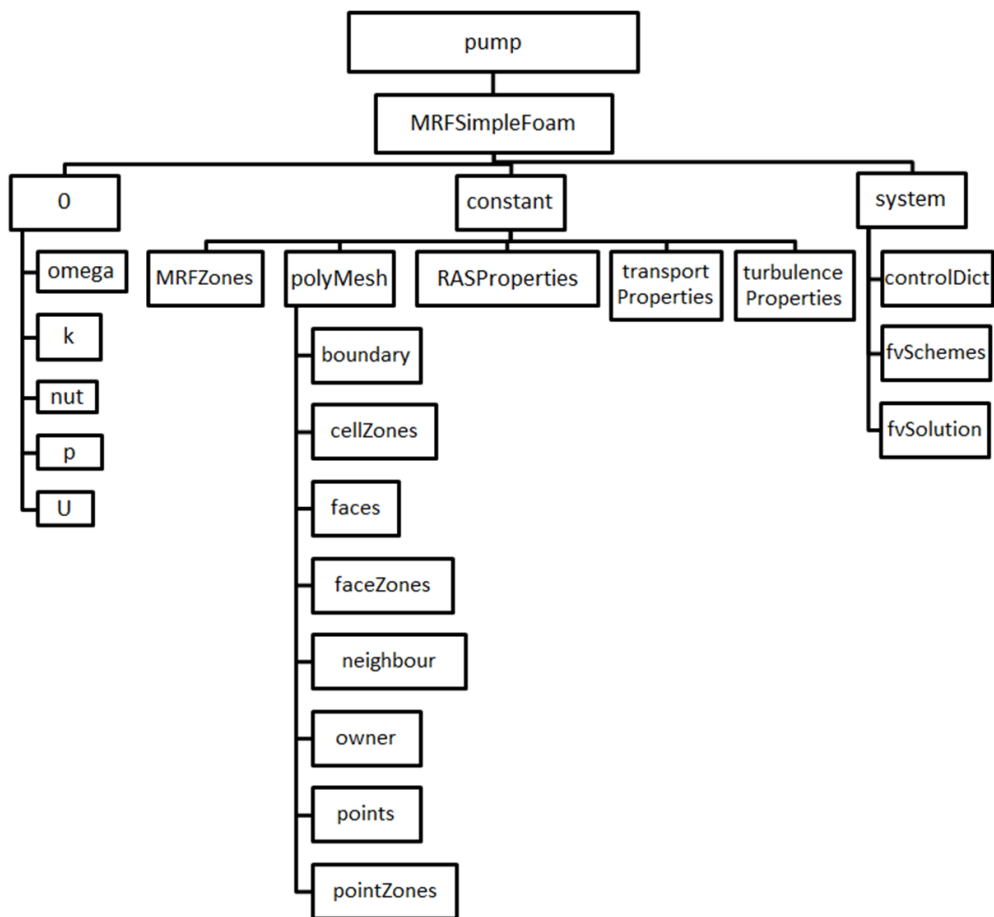


Figure C16 – MRFSimpleFoam – Final Case Structure

C5.2.2.1 '0' Time Directory

The next phase in the setup relates to setting the correct boundary conditions, turbulence models, moving reference frame parameters, solution schemes and general control parameters of the simulation.

The files relevant to these settings are housed in the *0* time directory and the following steps outline how these are modified for this case.

C5.2.2.1.1 k and ω Turbulence Properties

In this case, the k - ω SST turbulence model is used as it provides a good blend of the positive aspects of the purely k - ϵ and k - ω models. The initial conditions for the turbulence

fields can be estimated using the mean flow velocity (U), turbulence intensity (I), and the characteristic length scale of turbulence, L .

For this example, a typical turbulence intensity of 5% has been selected. The inlet velocity of 7.261 m/s is calculated from the best efficiency flow rate of 235 l/s through the 0.203m diameter inlet pipe. Using a characteristic length scale of 10% of this diameter produces 0.0203m.

The specific kinetic energy, k , is required to be calculated using the equation:

$$k = 1.5(U \cdot I)^2$$

where U is the inlet velocity in m/s and I is the turbulence intensity as a percentage.

The specific turbulence dissipation rate, ω , is calculated using:

$$\omega = C_{\mu}^{-0.25} \frac{\sqrt{k}}{L}$$

where C_{μ} is a constant of 0.009, k is the specific kinetic energy calculated above and L is the characteristic length scale.

In this case, the following values were used:

$$k = 1.5(7.261 \cdot 0.05)^2 = 0.198 \text{ m}^2/\text{s}^2$$

$$\omega = 0.09^{-0.25} \frac{\sqrt{0.198}}{0.0203} = 39.989 \text{ s}^{-1}$$

With this information calculated, the k and ω files can be modified using the 'gedit' command in line with Figure C17 and Figure C18.

```

1 /*-----* C++ *-----*/
2 |=====|
3 | \ / \ / | F i e l d | OpenFOAM: The Open Source CFD Toolbox
4 | / \ / \ | O p e r a t i o n | Version: 2.1.x
5 | / \ / \ | A n d | Web: www.OpenFOAM.org
6 | \ / \ / | M a n i p u l a t i o n |
7 |-----*-----*/
8 FoamFile
9 {
10     version      2.0;
11     format        asCll;
12     class         volScalarField;
13     location      "0";
14     object        k;
15 }
16 // *****
17
18 dimenstons      [ 0 2 -2 0 0 0 ];
19
20 internalField    uniform 0.198;
21
22 boundaryField
23 {
24
25 //*****THROATBUSH*****//
26
27     Throatbush_Inlet
28     {
29         type        fixedValue;
30         value        uniform 0.198;
31     }
32
33     Throatbush_Walls
34     {
35         type        kqRWallFunction;
36         value        $internalField;
37     }
38
39     Throatbush_Outlet
40     {
41         type        cyclicAMI;
42         value        $internalField;
43     }
44
45 //*****IMPELLER*****//
46
47     Impeller_Inlet
48     {
49         type        cyclicAMI;
50         value        $internalField;
51     }
52
53     Impeller_Hub
54     {
55         type        kqRWallFunction;
56         value        $internalField;
57     }
58
59     Impeller_Hub_Side
60     {
61         type        kqRWallFunction;
62         value        $internalField;
63     }
64
65     Impeller_Shroud
66     {
67         type        kqRWallFunction;
68         value        $internalField;
69     }
70
71     Impeller_Shroud_Side
72     {
73         type        kqRWallFunction;
74         value        $internalField;
75     }
76
77     Impeller_Blades
78     {
79         type        kqRWallFunction;
80         value        $internalField;
81     }
82
83     Impeller_Outlet
84     {
85         type        cyclicAMI;
86         value        $internalField;
87     }
88
89 //*****VOLUTE*****//
90
91     Volute_Inlet
92     {
93         type        cyclicAMI;
94         value        $internalField;
95     }
96     |
97
98     Volute_Walls
99     {
100        type        kqRWallFunction;
101        value        $internalField;
102    }
103
104    Volute_Walls_Extenson
105    {
106        type        kqRWallFunction;
107        value        $internalField;
108    }
109
110    Volute_Outlet
111    {
112        type        zeroGradient;
113    }
114 }
115
116 // *****

```

Figure C17 – Properties File - k

C5.2.2.1.2 Pressure and Velocity Properties

In a similar way to the turbulence properties, each boundary must have a pressure and velocity property associated with it. The velocity at the inlet (Throatbush_Inlet) is determined from the flow rate through the particular diameter of this pipe (0.152 m). The outlet 'pressure' condition is actually defined in this file as pressure divided by the fluid density and no specific value is required for this due to the single-phase nature of the setup. These values in addition to details on how all other boundaries were defined are shown in Figure C19 and Figure C20.

C5.2.2.2 'Constant' Directory

The next directory to investigate is the constant directory. Move here:

```
cd ..
```

```
cd constant
```

C5.2.2.2.1 MRFZones

Next, you want to define the 'rotating' properties of the pump impeller, which is done in the *MRFZones* file. If you are not already in the *constant* directory, change directory (*cd*) so that you are and use *gedit* to open the file for editing:

```
gedit MRFZones
```

The first thing to do is change the item *rotor* to *impellerzone*, which is the 'zone' that you created in the early stages of merging the meshes.

Next, add the non-rotating patches appropriately, in this case *Throatbush_Outlet*, *Impeller_Inlet*, *Impeller_Outlet* and *Volute_Inlet*.


```

1  /*-----* C++ -----*/
2  =====
3  \ \ \ \ \ Field      | OpenFOAM: The Open Source CFD Toolbox
4  \ \ \ \ \ Operation  | Version: 2.1.x
5  \ \ \ \ \ A nd       | Web: www.OpenFOAM.org
6  \ \ \ \ \ Manipulation |
7  /*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         volVectorField;
13     location      "0";
14     object        U;
15 }
16 // *****
17
18 dimensions      [ 0 1 -1 0 0 0 ];
19
20 internalField   uniform (0 0 0);
21
22 boundaryField
23 {
24
25 //*****THROATBUSH*****//
26
27     Throatbush_Inlet
28     {
29         type      fixedValue;
30         value      uniform (0 0 -7.261);
31     }
32
33     Throatbush_Walls
34     {
35         type      fixedValue;
36         value      uniform (0 0 0);
37     }
38
39     Throatbush_Outlet
40     {
41         type      cyclicAMI;
42         value      uniform (0 0 0);
43     }
44
45 //*****IMPELLER*****//
46
47     Impeller_Inlet
48     {
49         type      cyclicAMI;
50         value      uniform (0 0 0);
51     }
52
53     Impeller_Hub
54     {
55         type      fixedValue;
56         value      uniform (0 0 0);
57     }
58
59     Impeller_Hub_Side
60     {
61         type      fixedValue;
62         value      uniform (0 0 0);
63     }
64
65     Impeller_Shroud
66     {
67         type      fixedValue;
68         value      uniform (0 0 0);
69     }
70
71     Impeller_Shroud_Side
72     {
73         type      fixedValue;
74         value      uniform (0 0 0);
75     }
76
77     Impeller_Blades
78     {
79         type      fixedValue;
80         value      uniform (0 0 0);
81     }
82
83     Impeller_Outlet
84     {
85         type      cyclicAMI;
86         value      uniform (0 0 0);
87     }
88
89 //*****VOLUTE*****//
90
91     Volute_Inlet
92     {
93         type      cyclicAMI;
94         value      uniform (0 0 0);
95     }
96
97     Volute_Walls
98     {
99         type      fixedValue;
100        value      uniform (0 0 0);
101    }
102
103    Volute_Walls_Extension
104    {
105        type      fixedValue;
106        value      uniform (0 0 0);
107    }
108
109    Volute_Outlet
110    {
111        type      zeroGradient;
112    }
113 }
114 }
115 // *****

```

Figure C20 – Properties File - U

The other two parameters to be altered are the *origin* which requires the z-axis to be changed and *omega* which represents the rotational speed of the rotating *impellerzone* (impeller) zone in rad/s.

The file should now look as seen in Figure C21.

```

1 /*-----* C++ *-----*/
2 =====
3 // Field      | OpenFOAM: The Open Source CFD Toolbox
4 // Operation  | Version: 2.1.x
5 // And        | Web: www.OpenFOAM.org
6 // Manipulation
7 -----*/
8 FoamFile
9 {
10  version      2.0;
11  format       ascii;
12  class        dictionary;
13  location     "constant";
14  object       MRFZones;
15 }
16 // *****
17
18 1
19 (
20  impellerzone
21  {
22    // Fixed patches (by default they 'move' with the MRF zone)
23    nonRotatingPatches (Throatbush_Outlet Impeller_Inlet Impeller_Outlet Volute_Inlet);
24
25    origin      origin [0 1 0 0 0 0] (0 0 0.01262);
26    axis        axis   [0 0 0 0 0 0] (0 0 1);
27    omega       omega  [0 0 -1 0 0 0] 115.19; //1100 RPM
28  }
29 )
30
31 // *****

```

Figure C21 - MRFZones

Save the file and close, returning to the terminal.

C5.2.2.2.2 transportProperties

Use *gedit* to open the *transportProperties* file in order to specify the properties of the pumped fluid. Also remove everything below (and including) *CrossPowerLawCoeff* as they are not required.

Add in a *rho* term as seen below and edit the exiting *nu* term.

The file should now look as seen Figure C22.

Replace the term *kEpsilon* with *kOmegaSST* as seen in Figure C24.

```
1 /*-----* C++ *-----*/
2 |=====|
3 | \ \ \ / | F i e l d | OpenFOAM: The Open Source CFD Toolbox
4 | \ \ \ / | O p e r a t i o n | Version: 2.1.x
5 | \ \ \ / | A n d | Web: www.OpenFOAM.org
6 | \ \ \ / | M a n i p u l a t i o n |
7 /*-----*
8 FoamFile
9 {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     location      "constant";
14     object        RASProperties;
15 }
16 // *****
17
18 RASModel        kOmegaSST;
19
20 turbulence      on;
21
22 printCoeffs     on;
23
24
25 // *****
```

Figure C24 - RASProperties

Save and return to the terminal.

C5.2.2.3 'System' Directory

The next directory to investigate is the *system* directory which contains the files relating to the setup of the parameters associated with the actual simulation solution.

Move from the *constant* directory to the *system* directory:

```
cd ..
```

```
cd system
```

C5.2.2.3.1 controlDict

The first thing to alter in this file is the *controlDict*, where all of your run settings are stored. You will spend a lot of time modifying this file when running your simulations in OpenFOAM.

Open the file with *gedit* in order to make modifications.

Firstly, specify how many iterations you wish to run the simulation for by changing the *endTime*. In this case, a final time of 2000 was used.

You can then change how often you want OpenFOAM to write the solution data to file by changing the *writeInterval* setting. Change it to 100 or an alternative value if you wish.

An important thing to change here is the *purgeWrite* option which is the number of written data files that are saved at any one time, automatically replacing existing files with the most recent. Setting this to '2' should result in a new file being written every 100 timesteps but only timesteps 1900 and 2000 will be available to view once the simulation is complete.

The next important step is to include all of the *functions* which are relevant to this case. These are pieces of information and calculations that are performed during the simulation and are written to the log file (and terminal screen – more on this later) in order to assist the user in understanding what is happening during the simulation.

The mass flows and forces are calculated using OpenFOAM's in-built utilities. The more complex data of generated head, absorbed power, hydraulic power and efficiency are calculated using the *turboPerformance* and *fluidPower* functions which were included in the *turboPerformance* utility folder installed earlier.

Based on this information, you should modify the *controlDict* file so that it matches that seen in Figure C25, remembering to edit the file to match the names of your inlet and outlet patches.

Save the file and return to the terminal.

```

1 |-----* C++ -----*
2 |#####
3 | \ \ \ \ / F leld      | OpenFOAM: The Open Source CFD Toolbox
4 | \ \ \ \ / 0 peration  | Version: 2.1.x
5 | \ \ \ \ / A nd        | Web: www.OpenFOAM.org
6 | \ \ \ \ / M antipulation
7 |-----*-----*
8 |FoamFile
9 |
10 | version      2.0;
11 | format       ascii;
12 | class        dictionary;
13 | location     "system";
14 | object       controlDict;
15 |
16 | * * * * *
17 |
18 | application  MRFSimpleFoam;
19 |
20 | startFrom    startTime;
21 |
22 | startTime    0;
23 |
24 | stopAt       endTime;
25 |
26 | endTime     2000;
27 |
28 | deltaT       1;
29 |
30 | writeControl timeStep;
31 |
32 | writeInterval 100;
33 |
34 | purgeWrite   2;
35 |
36 | writeFormat  ascii;
37 |
38 | writePrecision 6;
39 |
40 | writeCompression off;
41 |
42 | timeFormat   general;
43 |
44 | timePrecision 6;
45 |
46 | runTimeModifiable true;
47 |
48 | functions
49 |
50 |
51 | *****TURBOPERFORMANCE*****
52 |
53 | turboPerformance
54 | {
55 |   type turboPerformance;
56 |   functionObjectlibs ("libturboPerformance.so");
57 |
58 |   turbine false; // Turbine mode, (false if Pump)
59 |
60 |   log true; // write data to screen (true/false)
61 |   outputControl timeStep; // write data to file (same options as case output)
62 |   outputInterval 1; // interval to write data to file
63 |
64 |   inletPatches (Throatbush_Inlet); // inlet patches, can be multiple
65 |   outletPatches (Volute_Outlet); // outlet patches, can be multiple
66 |   patches (Impeller_Hub Impeller_Hub_Side Impeller_Shroud Impeller_Shroud_Side Impeller_Blades);
67 |   // rotor/impeller patches, again can be multiple
68 |
69 |   rhoInf 997; // density
70 |   CoR (0 0 0.01202); // center of rotation
71 |   omega (0 0 -115.1917); // Rotational velocity (rad/s)
72 | }
73 | /*
74 |   pName p; //Optional: if p field is not called "p", give a new name here
75 |   uName Uabs; //Optional: if U field is not called "u", give a new name here
76 |   phiName phi; //Optional: if phi (flux) field is not called "phi", give a new name here
77 | */
78 |
79 | *****FLUID POWER*****
80 |
81 | fluidPower
82 | {
83 |   type fluidPower;
84 |   functionObjectlibs ("libturboPerformance.so");
85 |
86 |   turbine false; // Turbine mode, (false if Pump)
87 |
88 |   log true; // write data to screen (true/false)
89 |   outputControl timeStep; // write data to file (same options as case output)
90 |   outputInterval 1; // interval to write data to file
91 |
92 |   inletPatches (Throatbush_Inlet);
93 |   outletPatches (Volute_Outlet);
94 |   rhoInf 997;
95 | }
96 |
97 | *****MASS FLOW RATES*****
98 |
99 | massFlow
100 | {
101 |   type patchMassFlow;
102 |   functionObjectlibs
103 |   (
104 |     "libsimpleFunctionObjects.so"
105 |   );
106 |   verbose true;
107 |   patches
108 |   (
109 |     Throatbush_Inlet
110 |     Impeller_Inlet
111 |   );
112 |   factor -997;
113 | }
114 |
115 | massFlow
116 | {
117 |   type patchMassFlow;
118 |   functionObjectlibs
119 |   (
120 |     "libsimpleFunctionObjects.so"
121 |   );
122 |   verbose true;
123 |   patches
124 |   (
125 |     Impeller_Outlet
126 |     Volute_Outlet
127 |   );
128 |   factor 997;
129 | }
130 |
131 | *****
132 |
133 |
134 |
135 | libs (
136 |   "libOpenFOAM.so"
137 |   "libgrompp.so"
138 |   "libsimpleFunctionObjects.so"
139 |   "libswakFunctionObjects.so"
140 |   "libswakTopoSources.so"
141 |   // Needed to decompose on 1.7
142 |   "libcompressibleASModels.so"
143 | );
144 |
145 |
146 |
147 | * * * * *

```

Figure C25 – controlDict (MRFSimpleFoam)

C5.2.2.3.2 fvSchemes

The next file to edit is *fvSchemes* where the numerical schemes for terms, such as derivatives in equations, are set.

In this instance, the second order linearUpwind scheme is used for the velocity convection term. Open the file with *gedit* and modify the *divSchemes* and *laplacianSchemes* accordingly in order to resemble Figure C26.

```
1 /*-----* C++ -*-----*/
2 =====
3 \ V /   F i e l d   |   OpenFOAM: The Open Source CFD Toolbox
4 \ O /   O peration  |   Version: 2.1.x
5 \ A /   A nd        |   Web: www.OpenFOAM.org
6 \ M /   M anipulation
7 /*-----*/
8 FoamFile
9 {
10     version      2.0;
11     format       ascii;
12     class        dictionary;
13     location     "system";
14     object       fvSchemes;
15 }
16 // ***** //
17
18 ddtSchemes
19 {
20     default      steadyState;
21 }
22
23 gradSchemes
24 {
25     default      Gauss linear;
26     grad(p)      Gauss linear;
27     grad(U)      Gauss linear;
28 }
29
30 divSchemes
31 {
32     default      none;
33     div(phi,U)   Gauss linearUpwind Gauss;
34     div(phi,k)   Gauss upwind;
35     div(phi,omega) Gauss upwind;
36     div((nuEff*dev(T(grad(U)))) Gauss linear;
37 }
38
39 laplacianSchemes
40 {
41     default      none;
42     laplacian(nuEff,U) Gauss linear corrected;
43     laplacian(DKEff,k) Gauss linear corrected;
44     laplacian(DomegaEff,omega) Gauss linear corrected;
45 //     laplacian((1|A(U)),p) Gauss linear corrected;           //for MRF
46 //     laplacian(1,p) Gauss linear corrected;                 //for potentialFoam
47 }
48
49 interpolationSchemes
50 {
51     default      linear;
52     interpolate(U) linear;
53 }
54
55 snGradSchemes
56 {
57     default      corrected;
58 }
59
60 fluxRequired
61 {
62     default      no;
63     p            ;
64 }
65
66
67 // ***** //
```

Figure C26 – fvSchemes (MRFSimpleFoam)

It is worth mentioning that there are two lines that are currently ‘commented out’. These correspond to lines that are required only for a single type of solver run, in this case either

the `potentialFoam` and `MRFSimpleFoam` simulations. The requirement for these will become clearer in the next section.

Save the file and return to the terminal.

C5.2.2.3.3. fvSolution

The final file to edit is the `fvSolution` file in which the equation solvers, tolerances and algorithms are contained.

Open the file in `gedit` and change `epsilon` to `omega` under the `solvers` and `relaxationFactors` sections to reflect Figure C27.

Save the file and return to the terminal.

C5.2.2.4 Monitors

One of the important aspects when performing any simulations is to be able to monitor the progress of the simulation visually, viewing the convergence and identifying any issues. Here we will set up monitors in order to track residuals, total generated head, absorbed power, hydraulic power and efficiency.

C5.2.2.4.1 monitorResiduals

As an example, a monitor for tracking the residuals of the simulation will be outlined. This can then be used in order to create the additional monitors required.

Before starting, ensure that you are located in the *MRFSimpleFoam* directory.

Now, create the first monitor by typing:

```
gedit monitorResiduals
```

and modify the file to match the text as seen in Figure C28.

```
1 set logscale y
2 set title "Residuals"
3 set xlabel "Iterations"
4 set key below
5 plot "< cat log.MRF | grep 'Ux' | cut -d ' ' -f9" title 'Ux' w l, \
6 "< cat log.MRF | grep 'Ux' | cut -d ' ' -f9" title 'Ux' w l, \
7 "< cat log.MRF | grep 'Uy' | cut -d ' ' -f9" title 'Uy' w l, \
8 "< cat log.MRF | grep 'Uz' | cut -d ' ' -f9" title 'Uz' w l, \
9 "< cat log.MRF | grep 'Solving for omega|' | cut -d ' ' -f9" title 'k' w l
10 "< cat log.MRF | grep 'Solving for k' | cut -d ' ' -f9" title 'k' w l
11
12 pause 10
13 reread
```

Figure C28 – monitorResiduals

Note the use of *cat log.MRF*. This command tells the monitor to search the file *log.MRF* in order to collect information (this will be your output file when running the simulation). This search is narrowed by using the *grep* function, which searches each line for a particular string such as 'Solving for p'. Once these lines have been identified, the *cut* command is used in order to extract the specific data required to produce the plot.

Save the file and return to the terminal where you can repeat the process for the additional monitors used to track the performance figures related to the pump.

C5.2.2.4.2 monitorGeneratedHead

This monitor tracks the total generated head produced by the pump, calculated through the *turboPerformance* utility and should be set up as in Figure C29.

```
1 set yrange [0 : 100]
2 set title "Generated Head - 100% BEF"
3 set key below
4 set xlabel "Iterations"
5 set ylabel "Total Generated Head (m)"
6 plot 53.8 title 'Experimental' w l, \
7 "< cat log.MRF | grep 'Generated Head' | cut -d ' ' -f7" title 'MRF' w l
8
9 pause 10
10 reread
```

Figure C29 – monitorGeneratedHead

Note the use of the items at the top of the file such as *yrange* and *set title*. This allows your monitor to be customised. You can also add in fixed target/comparable/experimental values by simply adding them into the code, such as '53.8' in the example. More details regarding customisation can be found at www.gnuplot.info

C5.2.2.4.3 monitorPower

This monitor tracks both the power absorbed by the pump and hydraulic power produced by using the *turboPerformance* utility and is built in a similar manner to the previous monitor, with experimental data included as shown in Figure C30.

```
1 set yrange [0 : 350]
2 set title "Power"
3 set key below
4 set xlabel "Iterations"
5 set ylabel "Power (kW)"
6 plot 136.85 title 'Hydraulic Experimental' w l, \
7 179.21 title 'Absorbed Experimental' w l, \
8 "< cat log.MRF | grep 'Absorbed Power' | cut -d ' ' -f6" title 'Absorbed Power' w l, \
9 "< cat log.MRF | grep 'Hydraulic Power' | cut -d ' ' -f6" title 'Hydraulic Power' w l
10
11 pause 10
12 reread
13
```

Figure C30 – monitorPower

C5.2.2.4.4 monitorEfficiency

This monitor tracks the efficiency of the pump based on the absorbed and hydraulic powers calculated using the *turboPerformance* utility and can be seen in Figure C31.

```
1 set yrange [0 : 100]
2 set title "Efficiency"
3 set xlabel "Iterations"
4 set ylabel "Efficiency (%)"
5 plot "< cat log.MRF | grep 'Efficiency' | cut -d ' ' -f5" title 'Efficiency 8' w l, \
6 pause 10
7 reread
8
```

Figure C31 – monitorEfficiency

C5.2.2.5 Running the Simulation

Now that you have all the relevant files set up, you are in the position to run the frozen rotor simulation. This can be done in two ways; using a single processor or multiple processors. Select the route relevant to you, ensure that you are in the *MRFSimpleFoam* directory and follow the steps below.

Tip: Before doing so, it is advisable that you create a backup of the '0' time directory as a precaution:

```
cp -r 0 0.org
```

C5.2.2.5.1 Initialising Run

Before the 'main' solver runs are started, it is useful for the pressure fields to be initialised using the *potentialFoam* solver.

To begin, allow line 46 in Figure C26 to be read by removing the '//' text at the start of the line by using:

```
gedit system/fvSchemes
```

Next, return to the terminal and run the *potentialFoam* solver:

potentialFoam -writep

Now, before moving on to the more complete solver work, modify the *fvSchemes* file once again, this time, commenting out line 46 in Figure C26 and allowing line 45 to be read by the solver.

C5.2.2.5.2 Single Processor

Running this job on a single processor is very straightforward. To start the simulation, execute:

```
MRFSimpleFoam > log.MRF &
```

The items in this command can be explained in more detail. *MRFSimpleFoam* executes the specific solver that you desire to use, *> log.MRF* writes the data to a log file called *log.MRF* and using *&* allows this job to be run in the background in order for you to have access to the terminal during the simulation

In order to view the log file as it is being generated and ‘track’ the simulation progress, type:

```
tail -f log.MRF
```

This command tracks the ‘tail’ of the log file and print it to screen. You can stop tracking the file by typing:

```
CTRL + C
```

C5.2.2.5.3 Multiple Processors

Running the job on multiple processors is slightly more complex. First, you need to create a new file, *decomposeParDict*, that determines how you wish your mesh to be decomposed.

Copy an existing file from one of the tutorial cases to the *system* directory:

```
cp -r ~/OpenFOAM/user/
```

2.1.x/run/tutorials/incompressible/pisoFoam/les/motorBike/motorBike/system/decompose

```
ParDict system
```

Next, edit the file using *gedit*:

```
gedit system/decomposeParDict
```

and modify the file to match the following, choosing your *numberOfSubdomains* based on how many processors you wish to use:

```
1 /*----- C++ -----*/
2
3 =====
4 \ \ \ \ \ F i e l d
5 \ \ \ \ \ O p e r a t i o n
6 \ \ \ \ \ A n d
7 \ \ \ \ \ M a n i p u l a t i o n
8
9 FoamFile
10 {
11     version      2.0;
12     format       ascii;
13     class        dictionary;
14     location     "system";
15     object       decomposeParDict;
16 }
17 // *****
18 numberOfSubdomains 8;
19
20 method            scotch;
21
22 manualCoeffs
23 {
24     dataFile       "cellDecompositton";
25 }
26
27
28 // *****
```

Figure C32 - decomposeParDict

Save and close the file.

Next, you need to decompose the mesh. You can do this by executing one of the following commands:

```
decomposePar
```

```
decomposePar -time 0
```

decomposePar -latestTime

The latter will be more useful as you progress in using OpenFOAM as you may wish to start simulations from existing results as opposed to an initial time.

By executing this command, OpenFOAM has decomposed your mesh into the relevant number of components you stated in the *decomposeParDict* file and created new directories called *processor** in the *MRFSimpleFoam* directory, seen by using the 'ls' command.

In order to run the job, the following command requires to be executed:

```
mpirun -np 8 MRFSimpleFoam -parallel > log.MRF &
```

where '8' should be replaced with however many processors you have defined in the *dynamicMeshDict* file.

As mentioned in the steps for running the job with a single processor, *> log.MRF* writes the data to a log file called *log.MRF* and using *&* allows this job to be run in the background in order for you to have access to the terminal whilst it is running.

In order to view the log file as it is being generated and 'track' the simulation progress, type:

```
tail -f log.MRF
```

This command tracks the 'tail' of the log file and print it to screen. You can stop tracking the file by typing:

```
CTRL + C
```


C5.2.2.5.4 Tracking the Simulation

You will notice on the screen, if you are 'tailing' *log.MRF*, that the information regarding generated head, power and efficiency are printed. This is not available in the standard download of OpenFOAM but is available as you installed the *turboPerformance* utility during the installation sequence.

This is extremely beneficial and can be built upon by using the monitors that you created. In order to view one of the monitors, for example *monitorResiduals*, open a new window in the terminal by typing:

SHIFT + CTRL + T

or clicking *File>Open Tab*.

In this new blank tab, use the *gnuplot* command to view whichever monitor you like, e.g.:

gnuplot monitorResiduals

The result should look something like that seen in Figure C33.

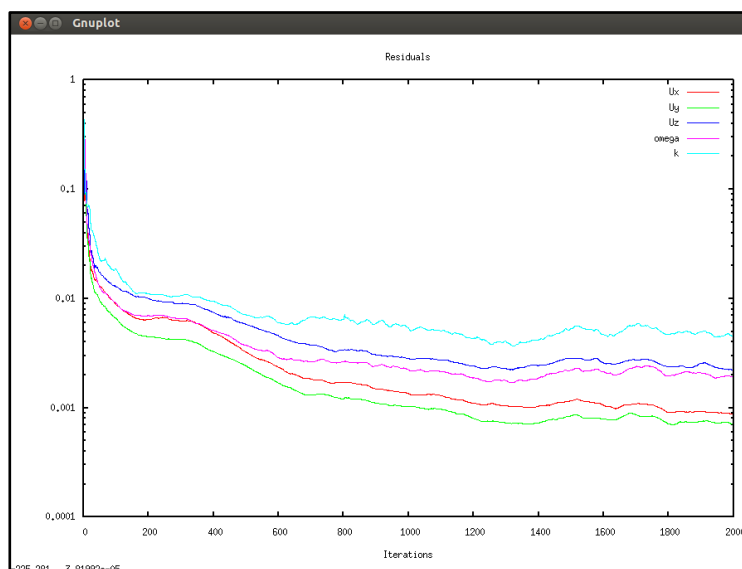


Figure C33 – Monitor of Residuals Using gnuplot

In order to close the monitor, click on the terminal and type *CTRL + C*.

C5.2.2.6 Reviewing the Simulation

Once the simulation complete, you are now able to view the results.

If you have run your simulation using multiple processors, the first thing to do is reconstruct the mesh. From the *MRFSimpleFoam* level, type:

```
reconstructParDict -latestTime
```

NOTE: If you ran the simulation in serial then you can ignore this step.

In order to take average of the *turboPerformane* parameters, you need to use the *fluidPower* and *turboPerformance* folders that have been created during the simulation. The *fluidPower* folder contains the data for hydraulic power output and generated head whilst the *turboPerformance* folder contains the data for absorbed power, efficiency, generated head (again) as well as the relevant forces.

To view and average results you firstly need to open these files using:

```
gedit fluidPower/1/fluidPower.dat
```

or

```
gedit turboPerformance/turboPerformance.dat
```

With the file open, use *CTRL + A* to select all of the data contained in the file.

Now you have an option to open 'LibreOffice Calc', an equivalent to Microsoft Excel, by clicking the icon on the Ubuntu menu bar on the left hand side of the screen.

Next, right-click where you want the information to be placed and select *paste*. Ensure that the *tab* checkbox is selected under the *Separated by* option and click *OK*.

With the information now in the correct format, you can average the data using the 'average' function, for example, over the last 100 iterations.

C5.2.2.6.1 paraView

In order to view the final results of the simulation in more detail, you can utilise OpenFOAM's post-processing tool paraView.

Now, as in the previous tutorial, open the paraView tool by typing:

paraFoam &

in the command line whilst in the 'MRFSimpleFoam' directory.

The following window should appear:

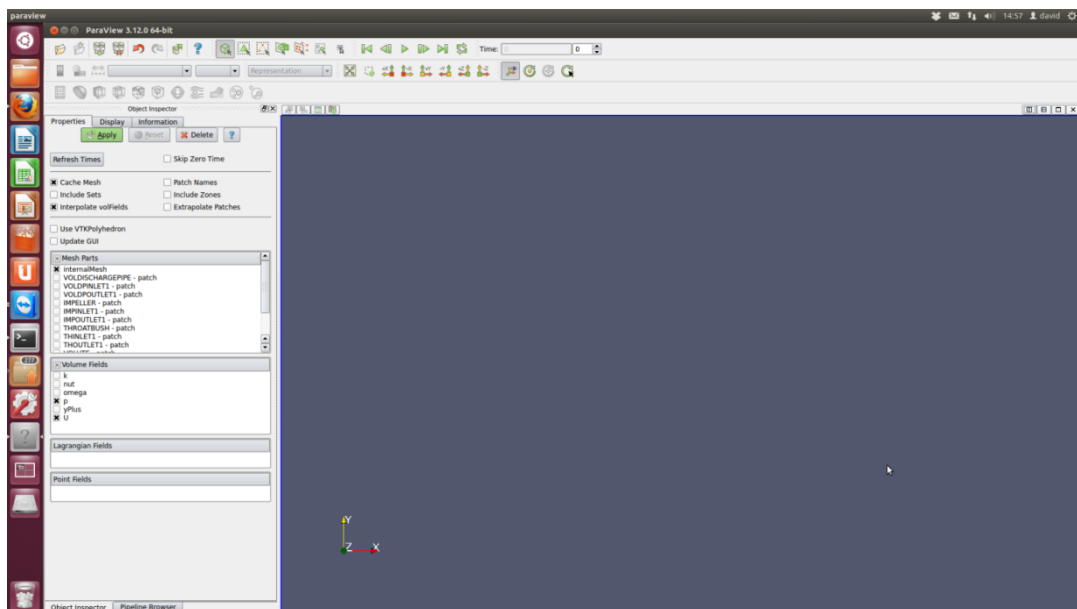


Figure C34 – Initial View in ParaView

Now, click on the 'last frame' icon as seen in Figure C5 in order to select the latest time step.

Include the 'zone' that you created in the earlier stages by selecting 'include zones' and choose the relevant components that you wish to view.

For this example, to view the impeller, select *impellerzone* from the list and then select the fields that you wish to view, in this case pressure and velocity.

Clicking Apply should provide you with the component(s) that you have selected. The default representation of the components may be as an 'outline'. This can be changed using the following dropdown menu and selecting 'surface'.

Next, select velocity (U) to be shown.

By selecting these options and clicking the rescale to data range icon, a similar image to that in Figure C35 should be seen.

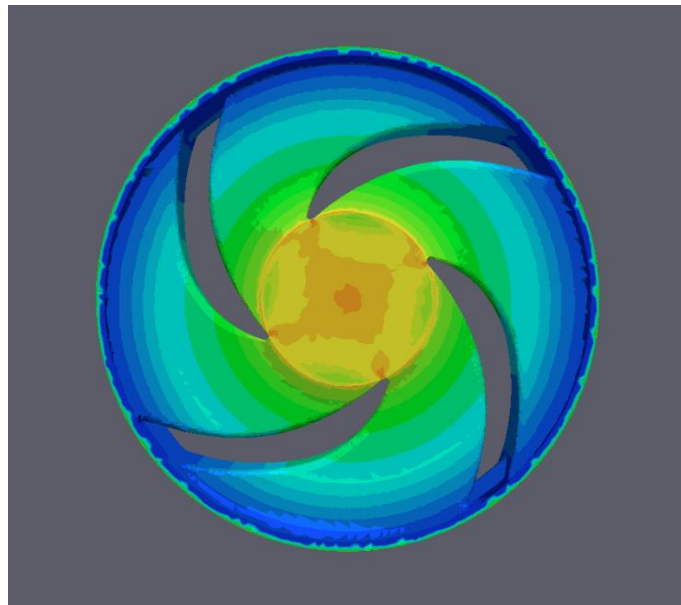


Figure C35 – Contour of Velocity on Impeller

Now, create a 2D section of the velocity results by clicking on the slice icon (Figure C5) and the z-normal plane (or whichever is suited to your particular case):

Clicking apply should result in a representation similar to Figure C36.

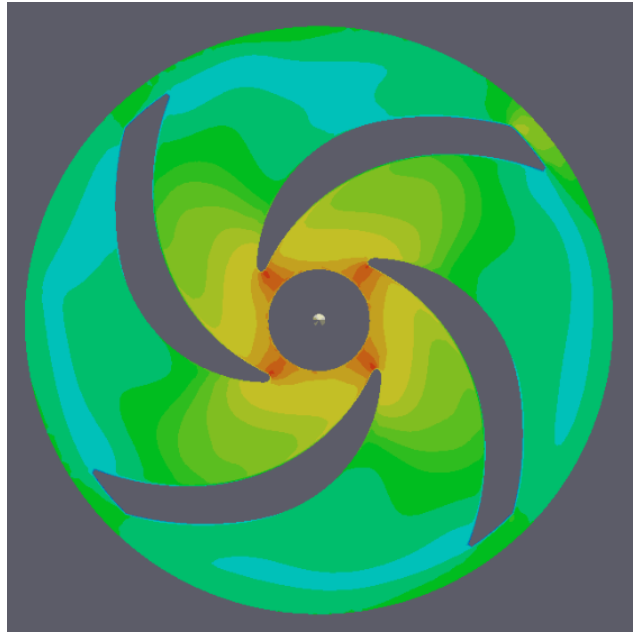


Figure C36 – Slice of Velocity Contours within Impeller

As before, the colour scale can be changed by clicking the 'Display' tab in the 'Object Inspector' before selecting 'Edit Color Map/Choose Preset' and choosing the desired range.

In order to view the legend on the screen, select 'Color Legend' at the top left hand corner of the screen (Figure C5). This can be moved to a different position on the screen by simply dragging it.

To remove the central cross from the screen, click the 'show-center' button (Figure C5).

Similar methods can be used to represent other results and you can spend time investigating this if you desire.

As with similar packages, you can save images in a number of formats including .jpg, .bmp and .png by selecting:

File>Save Screenshot>ok

or by using Ubuntu's built in screenshot facility by selecting:

Applications>Accessories>Take Screenshot

C5.2.3 transientSimpleDyMFoam (Transient) Simulation

Now that you have the results from the *MRFSimpleFoam* frozen rotor simulation, you can progress to the transient simulation which will utilise the *transientSimpleDyMFoam* solver that you downloaded and compiled during the installation sequence.

The *transientSimpleDyMFoam* solver is a transient solver for incompressible turbulent flow. The term 'DyM' in the name informs the user that it has the capability of modelling a moving (in this case rotating) mesh. It is different from the OpenFOAM's in-built solvers in that a SIMPLE-based algorithm is utilised within the time-stepping mode and that the turbulence model solution is moved inside this loop, making it more robust.

The setup for this new simulation will utilise the case structure from the *MRFSimpleFoam* simulation as a template which will be modified accordingly.

The first step is to return to the main *pump* directory and copy the frozen rotor case, creating a new directory *transientSimpleDyMFoam* from which the transient simulation will be housed:

```
cp -r MRFSimpleFoam transientSimpleDyMFoam
```

Next, move in the newly created folder:

```
cd transientSimpleDyMFoam
```

and clear up the folder by deleting items generated by the *MRFSimpleFoam* solver that are not required:

```
rm -r 0 0.org fluidPower log.* turboPerformance patchMassFlow* processor*
```

You should now be left with your time directory from the frozen rotor simulation (2000), the *constant* and *system* directories and the monitors you created. You can check this using the *'ls'* command. Whilst mentioning this latest time directory, you should remove the 'uniform' folder contained in the time directory:

```
rm -r 2000/uniform
```

Regarding the monitors, you can edit these now in *gedit*, changing all references of *log.MRF* to *log.transient* as this is the name of the log file that will be used during this simulation.

Tip: A quick way of doing this is by using the 'find and replace' function in *gedit*.

C5.2.3.1 Directory Modifications

There are a few modifications that are required to be made in order for the new solver to run effectively. These are highlighted below.

C5.2.3.1.1 dynamicMeshDict (constant Directory)

In order for the solver to work correctly, you must define the properties of the moving region, similar to the *MRFZones* in the frozen rotor simulation. For this transient simulation, the details of the rotating region are stored in the *dynamicMeshDict* file within the *constant* directory.

As there is no file of this name at present, copy an existing file from a tutorial case and open in *gedit*:

```
cp -r ~/OpenFOAM/user-
```

```
2.1.x/run/tutorials/incompressible/pimpleDyMFoam/propeller/constant/dynamicMeshDict
```

```
constant
```

```
gedit constant/dynamicMeshDict
```

Now, modify the file to suit this case, with region *impellerzone* rotating around the z-axis in this example as shown in Figure C37.

```
1 /*-----* C++ -*-----*/
2 |=====|
3 | \ \ \ \ | F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4 | \ \ \ \ | O p e r a t i o n | Version: 2.1.x |
5 | \ \ \ \ | A n d | Web: www.OpenFOAM.org |
6 | \ \ \ \ | M a n i p u l a t i o n | |
7 /*-----*/
8 FoamFile
9 {
10  version      2.0;
11  format       ascii;
12  class        dictionary;
13  location     "constant";
14  object       dynamicMeshDict;
15 }
16 // ***** //
17
18 dynamicFvMesh  solidBodyMotionFvMesh;
19
20 motionSolverLibs ( "libfvMotionSolvers.so" );
21
22 solidBodyMotionFvMeshCoeffs
23 {
24  cellZone     impellerzone;
25
26  solidBodyMotionFunction  rotatingMotion;
27  rotatingMotionCoeffs
28  {
29    CofG      (0 0 0.01262);
30    radialVelocity (0 0 6600); // deg/s
31  }
32 }
33
34
35 // ***** //
```

Figure C37 - dynamicMeshDict

C5.2.3.1.2 controlDict (system Directory)

The next file to modify is the *controlDict*, in which a few items require to be altered.

Firstly, the *application* should be changed to *transientSimpleDyMFoam*. Whilst this isn't actually necessary or read by OpenFOAM, it is good practice to ensure the names of the applications are correct.

Next, change the *startTime* from 0 to 2000 in order to utilise the frozen rotor simulation as the initial conditions for this analysis.

Following this, alter the timestep size, *deltaT*, to suit the transient nature of the simulation. In this instance, a time step 5.45×10^{-4} s used, i.e. 100 timesteps per revolution.

The *endTime* can also be altered in order to account for a chosen number of revolutions that you wish to run the simulation for. In this case, a final time of 0.2725 s is used to reflect 5 impeller revolutions.

The *timePrecision* and *writePrecision* options should also be altered in order to account for the new number of significant figures involved.

Make the relevant changes in order to match Figure C38.

```

18 application      transientSimpleDyMFoam;
19
20 startFrom        latestTime;
21
22 startTime        0;
23
24 stopAt           endTime;
25
26 endTime          0.2725;
27
28 //deltaT          0.001364;           //40 iterations
29 deltaT           0.000545;           //100 iterations
30 //deltaT          0.000273;           //200 iterations
31 //deltaT          0.000136;           //400 iterations
32
33
34 writeControl     timeStep;
35
36 writeInterval    10;
37
38 purgeWrite       2;
39
40 writeFormat      ascii;
41
42 writePrecision   7;
43
44 writeCompression off;
45
46 timeFormat       general;
47
48 timePrecision    7;
49
50 runTimeModifiable true;
51

```

Figure C38 – controlDict (transientSimpleDyMFoam)

C5.2.3.1.3 fvSolution (system Directory)

The next file to modify is *fvSolution* in order to account for the internal pressure correcting loops (note the appearance of *pcorr* and *pFinal*) and additional, more complex, transient effects.

By using *gedit*, modify the file from the existing code to that as seen in Figure C39.

```

18 solvers
19 {
20     pcorr
21     {
22         solver          GAMG;
23         tolerance       1e-7;
24         relTol          0.005;
25         smoother        GaussSeidel;
26         nPreSweeps      0;
27         nPostSweeps     2;
28         cacheAgglomeration off;
29         agglomerator    faceAreaPair;
30         nCellsInCoarsestLevel 20;
31         mergeLevels     1;
32         maxIter         100;
33         minIter         1;
34     }
35     p
36     {
37         Spcorr;
38         tolerance       1e-5;
39         relTol          0.01;
40     }
41     pFinal
42     {
43         Sp;
44         tolerance       1e-6;
45         relTol          0;
46     }
47 }
48
49 "(U|k|omega)"
50 {
51     solver          PBiCG;
52     preconditioner  DILU;
53     tolerance       1e-7;
54     relTol          0;
55     minIter         1;
56     maxIter         100;
57 }
58
59 "(U|k|omega)Final"
60 {
61     solver          PBiCG;
62     preconditioner  DILU;
63     tolerance       1e-8;
64     relTol          0;
65     minIter         1;
66     maxIter         100;
67 }
68 }
69 }
70
71 }
72
73 PISO
74 {
75     nCorrectors      2;
76     nOuterCorrectors 10;
77     nNonOrthogonalCorrectors 0;
78     correctPhi       true;
79
80     pRefCell         0;
81     pRefValue        1.013e5;
82 }
83
84 relaxationFactors
85 {
86
87     p          0.3;
88     U          0.7;
89     k          0.4;
90     omega      0.4;
91
92 }

```

Figure C39 – fvSchemes (transientSimpleDyMFoam)

C5.2.3.1.4 fvSchemes (system Directory)

Still in the *system* directory, open *fvSchemes* using gedit in order to make the relevant modifications as seen in Figure C40.

```

18 ddtSchemes
19 {
20     default Euler;
21 }
22
23 gradSchemes
24 {
25     default Gauss linear;
26     grad(p) Gauss linear;
27     grad(U) Gauss linear;
28 }
29
30 divSchemes
31 {
32     default none;
33     div(phi,U) Gauss upwind;
34     div(phi,k) Gauss upwind;
35     div(phi,omega) Gauss upwind;
36     div((nuEff*dev(T(grad(U)))) Gauss linear;
37 }
38
39 laplacianSchemes
40 {
41     default Gauss linear corrected;
42 }
43
44 interpolationSchemes
45 {
46     default linear;
47 }
48
49 snGradSchemes
50 {
51     default corrected;
52 }
53
54 fluxRequired
55 {
56     default no;
57     pcorr ;
58     p ;
59 }

```

Figure C40 – fvSchemes (transientSimpleDyMFoam)

Note that the time discretisation scheme used in this case is the first order Euler scheme. The *divSchemes* used for *U*, *k* and *omega* are also *upwind* i.e. first order. These can clearly be modified here in order to improve the results of the simulation however, these settings will be used in the following simulation as an example.

C5.2.3.1.5 U (2000 Time Directory)

The final file to edit is the velocity fields file, copied from the last timestep of the frozen rotor simulation, in order to take the dynamic mesh into consideration.

Do this by moving to the *2000* time directory and open the *U* file using *gedit*. Once you have the file open, use *CTRL +F* to open the search tool. Search for the relevant items (in this case *Impeller_Hub*, *Impeller_Hub_Side*, *Impeller_Shroud*, *Impeller_Shroud_Side* and

Impeller_Blades) and modify the *type* from *fixedValue* to *movingWallVelocity* as seen in Figure C41.

```
Impeller_Blades
{
    type          movingWallVelocity;
    value         nonuniform List<vector>
}
```

Figure C41 – Modification Required Moving Walls

C5.2.3.2 Running the Simulation

Now that you have modified the relevant files for the purposes of the transient analysis, you are in the position to run the simulation. Once again, this can be done in two ways – using a single processor or multiple processors. Choose the route relevant to you, ensure that you are in the *transientSimpleDyMFoam* directory and follow the steps below.

Before doing so, it is always good practice to create a backup copy of these directories in case any issues arise. Do this now:

```
cp -r 2000 2000.org
```

C5.2.3.2.1 Single Processor

In order to run this job on a single processor, the process is very straightforward. To start the simulation, execute:

```
transientSimpleDyMFoam > log.transient &
```

In order to view the log file as it is being generated and ‘track’ the simulation progress, type:

```
tail -f log.transient
```

This command tracks the 'tail' of the log file and print it to screen. You can stop tracking the file by typing:

CTRL + C

C5.2.3.2.2 Multiple Processors

As you have copied the *decomposeParDict* file from the steady state simulation, the multiple processor route is now more straightforward (assuming you still wish to use the same number of processors as in for the previous simulation)

Decompose the mesh once again by executing one of the following commands:

decomposePar

decomposePar -time 2000

decomposePar -latestTime

In order to run the job, the following command requires to be executed:

mpirun -np 8 transientSimpleDyMFoam -parallel > log.transient &

where '8' should be replaced with however many processors you have elected to use.

In order to view the log file as it is being generated and 'track' the simulation progress, type:

tail -f log.transient

This command tracks the 'tail' of the log file and print it to screen. You can stop tracking the file by typing:

CTRL + C

C5.2.3.3 Post-Processing

In order to view the results of the transient simulations, the same process as seen in the steady state/frozen rotor discussion previously. This time, however, the results will differ at each time step.

C5.2.3.4 Varying the Flow Rate

It is often required to run simulations at several off-design flow rate conditions. This can be done relatively easily in OpenFOAM.

First, you have to reconstruct the results file as before, for example:

```
reconstructPar -latestTime
```

Next, open the velocity field file relating to this latest time using gedit:

```
gedit 'latestTime'/U
```

Now, search for the name of the patch that you wish to vary the boundary condition for, such as the inlet pipe in this case (Throatbush_Inlet) using CTRL + F.

Edit the velocity outlet condition to represent the flow rate you wish to use, such as increasing the original value by 30% as seen in Figure C42.

```
Throatbush_Inlet
{
    type          fixedValue;
    value         uniform (0 0 -9.439);
}
```

Figure C42 – Modification Required for Change in Inlet Velocity

Save the edited file.

If you are running the new simulation in parallel, remove the existing 'processor' files and decompose the new case setup:

```
rm -r processor*
```

```
decomposePar -latestTime'
```

NOTE: Before starting the simulation, ensure that the *controlDict* is set up to start from the correct time step.

Run the simulation as before using either a serial or parallel technique and use the methods or post-processing learned to view the appropriate results.

C5.3 Converting Results from OpenFOAM to CFD-Post

Whilst the post-processing tools of paraView are extremely useful, it is sometimes preferred to post-process using ANSYS CFD-Post, particularly for turbomachinery applications in order to review designs in a particular manner. This can be done with some effort using the following steps and the use of the *foamMeshToFluent* and *foamDataToFluent* commands, with the resulting data being automatically written into a newly created folder within the *constant* directory called *fluentInterface*.

Firstly, as *foamMeshToFluent* uses the *startTime* specified in the *controlDict* from which to copy the mesh, this should be modified accordingly, either specifying *latestTime* or a specific time you wish to review in this field.

Next, you are required to create a new file in order to utilise the functionality of *foamDataToFluent*. Located in the *system* directory, use *gedit* to create the file as seen in Figure C44.


```

1 /*-----* C++ -*-----*/
2 |=====|
3 | \ \ \ \ \ | F i e l d | OpenFOAM: The Open Source CFD Toolbox
4 | \ \ \ \ \ | O p e r a t i o n | Version: 2.1.x
5 | \ \ \ \ \ | A n d | Web: www.OpenFOAM.org
6 | \ \ \ \ \ | M a n i p u l a t i o n |
7 /*-----*-----*/
8 FoamFile
9 {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     location      "system";
14     object        foamDataToFluentDict;
15 }
16 // ***** //
17
18 p      1;
19 U      2;
20
21 // ***** //

```

Figure C43 - foamDatatoFluent

This specific example only requests that the values of p and U be converted, however further details in relation to this can be found in the official OpenFOAM User Guide.

Next, run the following commands:

foamMeshtoFluent

foamDatatoFluent

By doing so, a new folder named *fluentInterface* should have been created and should now hold a .msh file and .dat file relating to the time step being investigated.

Now transfer the .msh and .dat files for the relevant time step to a machine running ANSYS CFD in which ever method is easiest.

The next step is to load ANSYS Workbench.

Add a Fluent module and right-click on 'Setup' before selecting 'Edit' as seen in Figure C44.

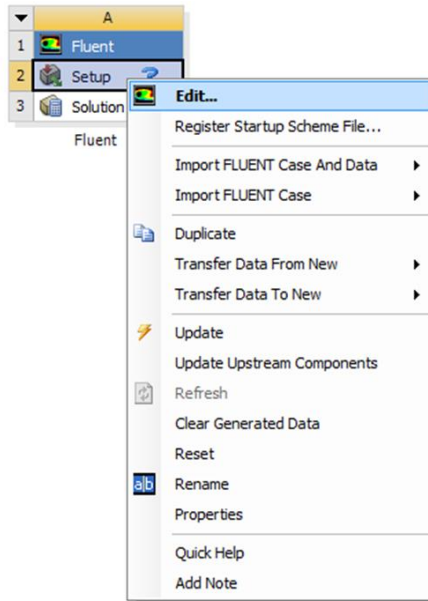


Figure C44 – Fluent Module

The information obtained from OpenFOAM can now be loaded into Fluent by using:

File > Import > Mesh (selecting the relevant .msh file)

File > Import > Data (selecting the relevant .dat file)

Now save this case for use in CFD Post by using File > Export > Solution Data which will present the screen as seen in Figure C45.

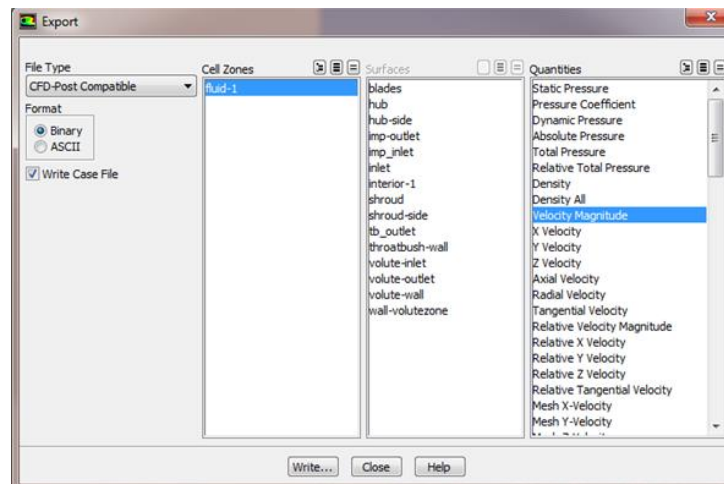


Figure C45 – Export Options for OpenFOAM Results in ANSYS CFD-Post

Apart from the options selected, the only additional modifications that are required are in relation to which properties (quantities) you wish to export to file.

Once this has been decided, select 'Write' which will save a .cas and .cdat file to a specified location.

Next, exit from the Fluent module as this is no longer required.

Finally review the results in CFD-Post by loading a standalone 'Results' module in Workbench as seen in Figure C46 and then using the File>Load Results.

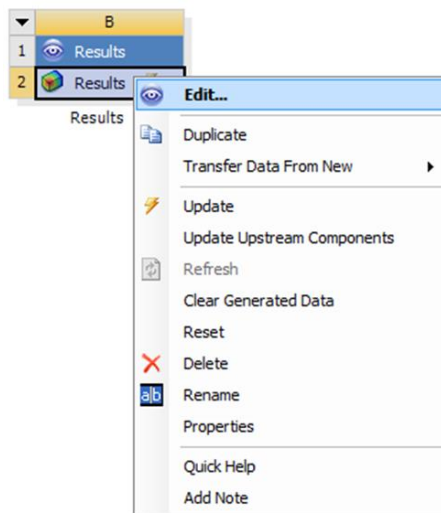


Figure C46 – Standalone Results Module in ANSYS Workbench

This should allow you to view the results as if the original solution had been generated using an ANSYS software product and is particularly useful as it allows the use of the 'Turbo' tool that has been developed specifically for turbomachinery applications.