The Pennsylvania State University

The Graduate School

College of Engineering

FLUID–STRUCTURE INTERACTION AND INVERSE DESIGN SIMULATIONS FOR FLEXIBLE TURBOMACHINERY

A Dissertation in

Mechanical Engineering

by

Robert L. Campbell

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2010

The dissertation of Robert L. Campbell was reviewed and approved[1] by the following:

Eric G. Paterson
Professor of Mechanical Engineering
Dissertation Adviser
Co-Chair of Committee

Robert F. Kunz
Professor of Aerospace Engineering
Co-Chair of Committee

John M. Cimbala
Professor of Mechanical Engineering

Stephen A. Hambric
Professor of Acoustics

Boris Leschinsky
Vice-President Technology, Datascope Corporation
Special Member

Karen A. Thole
Professor of Mechanical Engineering
Head of the Department of Mechanical and Nuclear Engineering

[1] Signatures on file in the Graduate School.

Abstract

Highly flexible turbomachinery comprised of soft polymeric impellers, exhibit large, time-dependent deformation when subjected to fluid stresses during operation. The large deformations of the impeller blades and the close proximity of the blades to the pump casing require simulations that consider the interaction of the fluid flow and the structural deformations. This thesis explores the use of fluid–structure interaction (FSI) modeling to perform time-accurate simulations of flexible polymeric turbomachinery and also explores the use of an inverse structural analysis to account for blade deformations over an initial startup period. The purpose of the inverse analysis is to determine the shape of a blade that, when acted on by fluid stresses, will deform into the design shape.

A partitioned FSI solver is developed, using the OpenFOAM software and an author-developed finite element (FE) structural solver, to perform FSI simulations of flexible turbomachinery. The flow and structural solvers are tightly coupled using fixed-point iterations to ensure fully converged structural and flow solvers for each solution time step. The solver interface supports disparate flow and structural meshes through interpolation and load mapping algorithms. A water tunnel test of a modified NACA 66 viscoelastic fin is performed at multiple angles of attack to generate validation data for the FSI solver. The validated solver is applied to an expandable impeller pump to simulate time-accurate performance changes that result from impeller elastic and viscoelastic deformation under application of the fluid stresses.

An inverse FE structural solver is developed and used to compute inverted structural shapes that account for deformations due to fluid loads so that the structures deform into their design shapes after elastic and viscoelastic deformations

occur. The inverse solver is validated for several cases based on numerical simulations. Time-accurate performance estimates of the expandable impeller pump for the inverted impeller shape demonstrate the accuracy of the inverse procedure when subjected to time-varying fluid forces. FSI simulation results for inverted modified NACA 66 fins are also presented. The deformed inverse shapes show good agreement with the intended (design) shapes, but slight discrepancies exist between the prescribed and simulated time at which these shapes are achieved. The slight discrepancies in the target times are attributed to inaccuracies in the load histories assumed during the inverse analyses.

Table of Contents

List of Tables

List of Figures

Acknowledgments

This research was conducted at the Applied Research Laboratory (ARL) of the Pennsylvania State University. Financial support was provided by Datascope Corporation in the form of a gift to Penn State. Special thanks goes to Mr. Boris Leschinsky, formerly of Datascope, for making possible this financial gift. Without this sponsorship, the work presented in this dissertation would not have been possible. I would like to also thank Dr. Richard Stern and the ARL Exploratory and Foundational (E&F) Program for the financial support to write this dissertation.

I would like to acknowledge my advisor Dr. Eric Paterson for his guidance, encouragement, and persistance during this endeavor. Throughout this process he has served as both a mentor and a colleague. I appreciate our candid conversations and have much respect for his technical prowess. A special thanks goes to my Office Head at ARL, Mr. Charles Brickell. Not only did he graciously pay for some of the test equipment required for the water tunnel tests, he also persistently inquired about my thesis progress and reminded me of the importance of this degree to my family. Drs. Stephen Hambric and Dean Capone were instrumental in ensuring I had sufficient time away from my regular duties to complete this research. I am grateful for their support and acknowledge the importance of their interest in the education of others.

Most importantly, I would like to acknowledge my wife April and son Kenton. Completion of this research would not have been possible without their support and understanding.

Chapter 1

Introduction

Fluid–structure interaction (FSI) is the interaction of a moveable and/or deformable structure that is immersed in a fluid and/or contains a fluid. Motion of the structure causes a change in the fluid's stresses that act on the structure's wetted surface, which in turn causes a change in the structure's motion. A model that captures such an interaction must use two-way coupling, where the fluid motion affects the structure's motion and the structure's motion affects the fluid's motion. Moreover, FSI might involve oscillatory or non-oscillatory interactions. Oscillatory interactions occur when the structure experiences strain due to fluid forces, deforms toward its original configuration to reduce the strain, but is forced back into the strained configuration once again by the fluid forces. This interaction continues causing oscillatory motion of the structure. Non-oscillatory interactions are those that cause a steady or quasi-steady strain in the structure due to fluid forces.

Fluid–structure interaction modeling has been a very active area of research in recent years as evidenced by numerous papers in the literature. Advances in computer capacity concurrent with the maturation of flow and structural modeling have made feasible these coupled simulations. The long-term goal of research in this area is to make FSI simulations commonplace in the design and analysis environment for real-world applications. Such capability would be very beneficial to many industries, including the biomedical industry for the understanding and treatment of disease. Several examples of such have appeared in the recent literature. For example, Vierendeels, DeHart, and others have investigated the

ventricle filling process of the human heart [143] and the flow field and forces acting on the heart valves with the aim of improving the design of prosthetic valves [142, 26], Wolters et al. [149] simulated aortic wall stress with the intent of predicting rupture risk for abdominal aortic aneurysms (AAA), and Wall et al. [144] have simulated airflow patterns in both diseased and healthy lungs to improve drug delivery through improved understanding of particle and aerosol depositions.

1.1 Motivation

This thesis focuses on the challenging problem of FSI modeling for an expandable impeller pump that could be implanted in the human heart [96]. The pump would collapse prior to insertion, be inserted through a small incision in the skin, traverse the arteries until correctly positioned in the heart's left ventricle, and then be deployed for operation. The impeller for such a device is required to undergo large elastic strains to enable the initial collapse and then expand under stored strain energy. The requirement of a large elastic strain is satisfied through the use of a soft polymeric material. The downside to a polymer for this application, however, is that the impeller will exhibit time-dependent deformation during operation as a result of the operating surface pressures acting on the blade surfaces and the viscoelastic nature inherent to all polymeric materials.

Experience has shown that blade deformations for such a pump cause changes to the impeller tip clearance (i.e., the clearance between the blade tips and the pump housing), thereby impacting pump performance. Impeller tip clearance changes, obtained using a one-way coupling (fluid stresses transferred to the structure, but no displacements transferred to the fluid, which differs from a two-way coupling as described earlier) between a computational fluid dynamics (CFD) model and a structural model with the fluid loads applied in the reverse direction, are shown in Figure 1.1. The purpose of this figure is to provide the reader with

a qualitative understanding of the problem at hand. The details of the analyses used to create this figure are not further discussed.

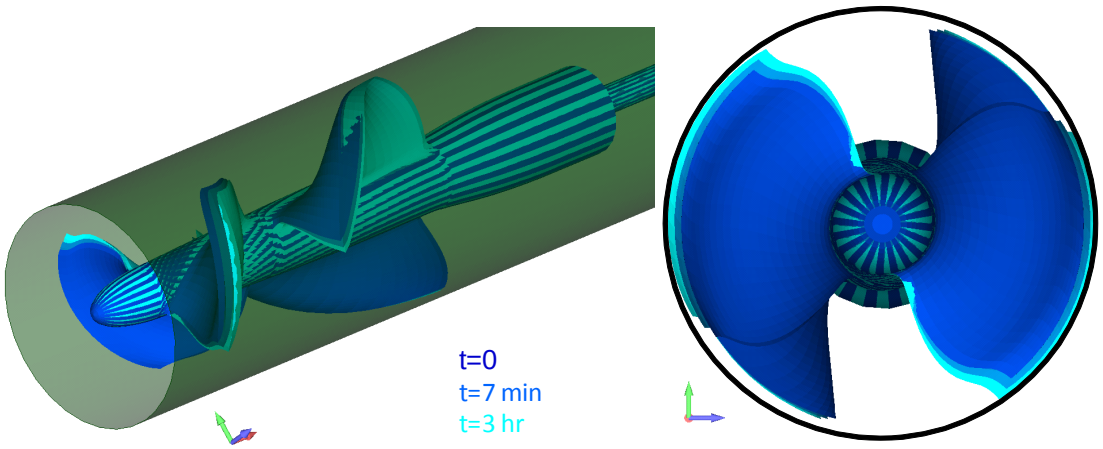

Fig. 1.1. Impeller tip clearance dependency on time; obtained from a one-way fluid-solid coupling model with negative loads applied

Knowledge of impeller performance is imperative to the design cycle in order to ensure the final pump design will meet the performance goals. For viscoelastic pump impellers, initial performance change with time is significant and must be considered and accounted for during the design phase. As described above, the *inverse* impeller shape shown in Figure 1.1 was created with a one-way analysis. Ideally, this shape would deform into the design shape at some time during operation of the pump. The ability to assess the correctness of this inverted shape and also to predict time-accurate pump performance requires two-way coupled FSI modeling because the fluid stresses depend upon the structural geometry, which in turn depends upon the fluid stresses and time for this viscoelastic material.

Fluid–structure interaction for the viscoelastic materials and flow conditions considered here is of the non-oscillatory type. If the material did not exhibit

viscoelastic effects, the fluid stresses would deform the structure into a static position where the structure's strains were sufficient to balance the steady fluid forces. However, the presence of viscoelasticity results in structural strains that continue to increase so that the structural stresses balance the fluid stresses, resulting in a structure that continues to deform slowly over a long period of time and only asymptotically approaches a steady state condition. The FSI simulations required for this system are therefore quasi-steady.

It is the intent of this work to first develop and validate an FSI solver, second develop an inverse modeling approach for the viscoelastic materials, and third apply the FSI solver to the inverted system to assess the validity of the inverse approach for the large-deformation, nonlinear, and time-dependent problem. In the process of achieving these goals, a method inherently will be developed and validated to simulate the time-accurate performance of a pump comprised of a viscoelastic impeller. Before discussing the present work, however, a summary of the work that has been done by others is provided below for some relevant topics.

1.2 Previous and Related Work

Fluid–structure interaction modeling is a field that has existed for some time, but recently has been gaining substantial interest in the engineering community as the ability to accurately perform these very demanding simulations for real-world systems is becoming feasible with current computer capacities. As described in the current section, many researchers are investigating FSI for various applications, but there have been very few validation studies, and to the author's knowledge nobody has performed FSI simulations of large-deformation viscoelastic turbomachinery.

The calculation of inverse models is also a field that has existed for some time, but has not been applied to systems comprised of viscoelastic materials. A summary of the work done in this field of study is also provided below.

1.2.1 Fluid–Structure Interaction

Fluid–structure interaction modeling has a long history dating back to the late 1970's. Research in this field emerged independently at three locations in the United States [42]: Northwestern University (Belytschko and Mullen), Cal Tech (Hughes and Liu), and Lockheed Palo Alto Research Laboratories (DeRuntz, Felippa, Geers, and Park → later moved to the University of Colorado at Boulder as the Center for Aerospace Structures, CAS). FSI was researched heavily for several years for application to the aerospace field, wherein predictions of flutter and other similar phenomena have been pursued. The general approach to FSI in this field of research is to employ what is referred to as a loosely-coupled partitioned approach (explained below). Researchers in the field are discovering that application of FSI to some types of problems render this approach inadequate for various reasons, and thus other approaches have come into play. A brief introduction to these approaches is provided next.

The various modeling approaches used in the field of FSI can essentially be grouped into two categories: *monolithic* and *partitioned*. The monolithic approach casts the governing equations for both the fluid and solid domains in terms of the same primitive variables (usually pressure and velocity [54, 61, 74]), and discretizes the entire domain using the same scheme. The main advantage of a monolithic approach is the seamless coupling of the fluid and structure domains, which can lead to improved solution stability. The drawbacks to this approach are cited as:

1. specialized, highly complex software is required [39, 108],

2. the rigidities of the fluid and the structure can be vastly different (e.g., $E/(\rho_f {c_f}^2) \sim 10^6$ for a steel structure in air, where E is Young's modulus of steel and ρ_f and c_f are the density and sound speed of air, respectively) and therefore the solutions for such cases are dominated by structural attributes rather than the combined interaction effects [103, 73]; in some cases,

the system matrix can be ill-conditioned with zeros on the diagonal leading to difficulties implementing a solver [61, 67],

3. the resulting set of equations might not be solvable due to prohibitively large amount of required memory because of the need to simultaneously store the unknown variables for both the fluid and structure domains, and therefore may require the use of a matrix-free approach [55, 67],

4. a single time step is used for all domains, which can lead to inefficiencies if disparate time scales are present [61], and

5. the use of a single mesh creates a challenge to generate meshes of suitably high quality for both domains [115].

The partitioned approach is by far the most heavily used approach based on the literature. This approach retains separate domains for the fluid (Ω_F) and structure (Ω_S) and separate solvers with independent discretizations are used in the modeling of each domain. The extent of the domains generally varies during an FSI simulation because of deformation of the structural domain in response to unbalanced fluid stresses. The union of these two domains comprises the total domain, $\Omega = \Omega_F \cup \Omega_S$. There are three boundaries that must be considered for the domain: fluid (Γ_F), solid (Γ_S), and the fluid/solid interface ($\Gamma_{F/S} = \partial\Omega_F \cap \partial\Omega_S$). These domains and boundaries are depicted in Figure 1.2. The most important

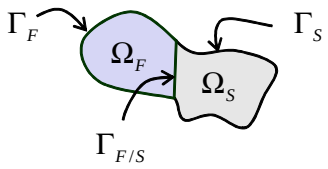

Fig. 1.2. Notional FSI problem domain showing the fluid, solid, and boundaries for each

advantages of the partitioned approach are the ability to separately maintain and advance the flow modeling and structural modeling software [41, 93, 121, 153] and to employ separate meshes for the structure and the fluid, which often require different mesh resolutions. The drawbacks to this approach are cited as:

1. the need to accurately and efficiently couple the two domains on $\Gamma_{F/S}$ [121], and the degradation of solution stability due to small errors in this coupling [42], and

2. poor solution stability for loosely coupled schemes [97] (described in Section 2.5.2), or costly sub-iterations of a tightly coupled scheme [97] (also described in Section 2.5.2).

The partitioned approach has been chosen for the current application based on a comparison of the benefits and drawbacks of each approach. While the monolithic approach offers a seamless fluid/structure communication, the ill-conditioned system of equations that sometimes arises due to disparate domain stiffnesses could be a limiting factor. The partitioned approach will require some consideration of the interface communication at $\Gamma_{F/S}$ to ensure an accurate and stable solution, but the use of a tightly coupled algorithm will provide results that are equivalent to those from a monolithic approach [93, 94, 140]. In addition, separate solvers can be developed, maintained, and advanced independently.

Fluid–structure interaction simulations have been applied to a vast range of applications in recent years. Applications have included parachutes [117, 132] and other cloth dynamics [111, 129], singing hydrofoils [104], nuclear reactor steam generator tube bundles [113], bridges [88], rotor dynamics [68], shape optimization studies [90], and a vast number of biomedical applications including arterial blood flow [43, 48, 60, 105, 128, 135, 136, 153], aortic heart valves [27, 25, 28, 26, 142], heart and ventricle [64, 81, 84, 86], lung modeling [144], and aortic aneurysms [35, 83, 130, 149]. Surprisingly, very few of these investigations performed verification

and validation studies of their FSI solvers. There has been a proposal by Turek and Hron [138] for a numerical benchmark for FSI, but the proposal has not yet gained any ground. The proposed benchmark problem is a cylinder with a trailing elastic plate subject to incompressible, laminar channel flow. The flow over the cylinder interacts with the trailing plate causing large oscillations of the plate. A similar problem, but for a square body instead of a cylinder, was introduced by Wall and Ramm [145] and has been used by others [33, 61, 95, 126, 150] to perform numerical evaluations of their solver. Both of these problems are two-dimensional. Experimental data are available for the Turek benchmark [51], but the author is not aware of similar data for the Wall and Ramm problem. A review of FSI modeling for turbomachinery is described separately in the following section.

1.2.1.1 Turbomachinery FSI Modeling

The body of literature related to FSI simulations and turbomachinery is very limited.[1] The FSI work that has been reported employs simplified flow models, simplified structural models, or is limited to steady-state conditions. A brief summary of relevant publications is provided below.

Lin and Lin [87] performed one of the earliest FSI simulations of a marine propeller using shell finite elements for the propeller and lifting surface theory with the steady Bernoulli equation for the flow. Their use of lifting surface theory does not include any effects of blade thickness on the flow field and the propeller material was modeled as linear elastic.

Gnesin and Rządkowski [50] investigated the aeroelastic behavior of an oscillating blade row of a turbomachine using an inviscid flow model and a modal

[1] Note that this review does not consider the large body of work related to small-deformation, dynamic aeroelastic/hydroelastic modeling of lifting surfaces (for such things as flutter), which represents a specialized subset of FSI (see, for example, [21] for additional information).

representation of the structure. Their approach employed a linear elastic material model and varied the modal coefficients with time to compute the structure's response.

Benra [11] investigated flow-induced oscillations of a single-blade, single-stage sewage water pump using commercial software and data exchange at the interface via output files. A one-way coupling of fluid pressures to the structure was employed for this work because of coupling difficulties resulting from disparate meshes at the interface. Benra concludes that a two-way coupling is needed for this model to improve agreement with experimental results.

Lastly, Young [152] employed a FSI model of a composite marine propeller to investigate the hydroelastic behavior in subcavitating and cavitating flows. The approach employs a low-order Boundary Element Model (BEM) with a non-commercial solver for the fluid and employs the commercial software Abaqus for the structure. Structural deformations are incorporated by updating the BEM geometry and iterating between the structural and flow solvers until the system converges. Comparisons for steady-state operation are made for computed and measured thrust and cavitation patterns.

1.2.1.2 Commercial FSI Software

There has been some movement in the commercial sector to develop multi-physics modeling capability, of which FSI represents a subset. The approach that has garnered the most attention for FSI seems to be a partitioned approach with coupling accomplished through the use of separate coupling software. The Mesh-based parallel Code Coupling Interface (MpCCI) [99] is a commonly referenced coupling software, which is supported by the finite element (FE) solver Abaqus [57], and possibly others. A thorough evaluation of commercial software capability has not been performed, but a limited investigation at the onset of this research has found that the available software tends to only support loosely coupled simulations.

The use of commercial software can have advantages over non-commercial codes, but the ability to apply these codes to new applications often requires several cycles of code releases, which can take years. The author performed some preliminary work and found the use of Abaqus for the current problem to be hindered by the need to communicate with the flow solver via output files, as further described in Chapter 2.

The *loosely coupled* approach employed by the commercial sector has been found by some researchers to be a limiting factor due to added-mass instability effects (see Section 2.5.2 for an explanation of the added-mass instability). For example, Timperi et al. [134] employed commercial coupling software to couple Fluent and Abaqus using MpCCI, and Star-CD and Abaqus using ES-FSI (both Star-CD and ES-FSI are products of the commercial company CD-adapco) to model a transient event of a commercial nuclear reactor. In both cases, the loosely coupled approaches failed to solve the problem and a one-way coupling (also loosely coupled by definition) approach was used instead. The one-way coupling approach caused high-frequency oscillations in the results and also contributed to large differences between the simulation and empirical results.

The quasi-steady nature of the expandable impeller pump investigated here should alleviate added-mass instabilities common to loosely-coupled solvers. However, reduced restrictions on time step size with a tightly coupled solver are attractive for this application because of the anticipated long simulation times. A tightly-coupled solver is therefore pursued for this work.

1.2.2 Inverse Structural Analysis

The discussion involving Figure 1.1 earlier in this chapter was meant to introduce the reader to the idea of an inverse problem. The use of the word *inverse* here is in contrast to the traditional 'inverse problem' used with measurements to ascertain material property or load information (see for example [16]) or a

die shape for material processing (see [77]). The inverse problem described here represents the inverse of the classical, or direct, problem of elasticity: determine the deformed shape of an object subjected to known loads in its reference (i.e., unloaded) configuration. The objective of the inverse analysis is to determine the *reference* configuration knowing its *deformed* shape. The focus here is on large-deformation problems in finite elasticity because the solution for small-deformation (i.e., linear) elasticity is trivial.

Shield [112] proposed the original formulation for inverse analysis in finite elasticity. His method exploited the duality in the governing equations when the role of the deformed and reference (or unloaded) configurations are interchanged. Chadwick [20] later revisited this duality to formulate Shield's equilibrium equations using Eshelby's energy-momentum tensor.

Yamada [151], Govindjee and Mihalic [52, 53], and Fachinotti et al. [36] separately developed inverse finite element formulations for finite deformations of hyperelastic bodies. The work of Govindjee and Mihalic involved the Eulerian formulation, Yamada used an arbitrary Lagrangian-Eulerian approach, and Fachinotti et al. used a Lagrangian formulation so that the derived approach could be implemented into existing software without substantial modifications.

Govindjee and Mihalic applied their solver in [52] to determine an undeformed gasket profile and applied their solver of [53] to a deformed seal and a rubber forming tool. In each of these cases, the finite element solver alone can be used to evaluate the efficacy of the inverted model. Fachinotti et al. [36] applied their inverse finite element solver to gas turbine blades to predict the inverted, or manufactured, shape of the blades. This analysis, however, requires the blade pressures in the design shape and do not account for load variations with blade deformation. In the extreme case, it could be argued that the loading would be insufficient for the blade to ever arrive at the design shape. One of the goals of

this work is to evaluate such inverted models to ascertain their robustness. The existence of a validated FSI solver is paramount to satisfying this goal.

1.3 Agenda

This thesis explores the use of inverse structure analysis for highly flexible turbomachinery, including a viscoelastic expandable impeller pump, to determine the manufactured shape required for the blades to deform into their design shape while operating. Fluid–structure interaction simulations are required to evaluate the inverted structural shapes and thus substantial effort is invested in developing and validating an FSI solver. The solver validation is accomplished with water tunnel testing of a simplified problem. An accurate viscoelastic material model is necessary for the simulations and thus additional component testing and simulation is performed with a focus on structural model validation. The large deformations of the viscoelastic structures complicates the fluid mesh motion of the FSI solver and therefore some effort is spent on the mesh motion solver development.

The specific contributions of this research to the field of computational mechanics include:

1. demonstration of FSI simulations for a highly flexible viscoelastic hydrofoil, with comparisons to experimental data,
2. acquisition of validation data for FSI simulations of a viscoelastic hydrofoil,
3. demonstration of FSI simulations for a highly-flexible turbomachine,
4. development of a novel approach for an inverse finite element solver, and
5. demonstration of the inverse finite element technique for a flexible, viscoelastic turbomachine.

1.4 Thesis Outline

Chapter 2 describes the implementation of the FSI solver used to achieve the objectives outlined above. Specific information is provided about the individual solvers that comprise the FSI solver and details of their interface communication. Details of the inverse structural solver and parallel processing for the FSI solver are also included in this chapter. Chapter 3 is dedicated to developing and validating a material model for the viscoelastomer used in this research. Beam and fin specimens are tested and compared to FE models to evaluate and refine the viscoelastic material model. Chapter 4 describes a water tunnel test of a viscoelastic fin. Data from this test are used to validate the FSI solver, which is the subject of Chapter 5. Chapter 5 also describes the mesh generation for the water tunnel FSI model, validation of the flow and structural solvers, and solver performance. Chapter 6 focuses on inverse simulations of the fin to determine inverted structural shapes for various angles of attack and describes FSI simulations of the inverted shapes to evaluate the use of the inverse technique for the viscoelastic fin. This chapter provides additional validation of the inverse structural solver. Chapter 7 applies the FSI solver and the inverse structural solver to the expandable impeller pump. Details of the mesh generation and the efficacy of the inverse solver for the pump are provided. The final chapter summarizes and provides conclusions of the research, and recommends future work.

Chapter 2

FSI Solver Implementation

The vast majority of the research reported in the introduction employed partitioned FSI solvers with the Arbitrary Lagrangian-Eulerian (ALE) formulation for the flow solver [63].[1] This formulation enables the fluid mesh to be deformed in response to structural deformations. Alternatives to the ALE approach include the immersed boundary method, fictitious domain method, and the mortar finite element method, all of which involve to some degree the more general Lagrange multiplier approach [4, 91, 79, 123, 124] and use fixed fluid meshes. Tezduyar et al. [131] describe the advantage of the ALE approach over the fixed-mesh alternatives is the ability to maintain high-quality meshes near the structure's interface, resulting in more accurate fluid mechanics in that region. For geometries of high complexity, remeshing is often required in addition to mesh motion. Based on this argument, and the availability of existing ALE flow solvers, the ALE formulation has been chosen for this work.

It should be noted that prior to implementing the solver described below, initial FSI simulations were performed using Fluent coupled to Abaqus [57] via Fluent User-Defined Functions [45] and OpenFOAM [102, 146] coupled to Abaqus using a custom OpenFOAM solver to facilitate the solver communication. These solvers functioned well for simple test problems, but were very inefficient because communication with the commercial structural solver took place with file input

[1] Flow solvers typically employ an Eulerian formulation where the computational mesh is fixed in space and the fluid particles move relative to the mesh. Structural solvers often use a Lagrangian formulation where the computational mesh moves with the associated material particle during motion. An ALE formulation allows an arbitrary motion of the computational mesh, allowing parts of the mesh to be moved with the particles similar to the Lagrangian approach, other parts of the mesh can be stationary like an Eulerian approach, or the mesh can be moved arbitrarily with no correlation to the particle motion.

and output, and the structural solver was restarted for every solve. Discussions with Abaqus technical representatives about the solver's FSI capability revealed their intentions of providing an enhanced interface to improve its multi-physics capability, but at that time the capability did not exist. At that point it was decided to move forward with all non-commercial software to implement a tightly-coupled FSI solver. Because of the lack of an open-source, validated structural solver capable of modeling large deformation viscoelastic response, a structural solver was created especially for this effort. Note that the creation of a structural solver has facilitated the implementation of an inverse FE solver and a displacement interpolation scheme for solver coupling. Both of these would have required substantial more effort had a commercial structural solver been used. Details of the structural solver are provided below after a discussion of the governing equations and a description of the OpenFOAM flow solvers.

2.1 Governing Equations

The governing equations for both the fluid and solid domains differ only in their constitutive relationships and therefore the general equations for continuum mechanics are first introduced, followed by the specifics for each domain. The equations are cast in an ALE form, which provides a very general framework that captures the Eulerian, Lagrangian, or an arbitrary frame of reference. The first equation to consider is the continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left[\rho \left(\boldsymbol{v} - \boldsymbol{v}^m \right) \right] = 0, \tag{2.1}$$

where ρ is mass density, $\boldsymbol{v}$ is the (fluid or solid) particle velocity, and $\boldsymbol{v}^m$ is the grid point velocity (which is required in this work to deform the fluid mesh to accommodate structural deformation). For a Lagrangian implementation, $\boldsymbol{v}^m = \boldsymbol{v}$,

and for an Eulerian implementation, $\boldsymbol{v}^m = 0$. Performing a force balance and making use of the continuity equation leads to the following momentum equation:

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho\left[\left(\boldsymbol{v} - \boldsymbol{v}^m\right)\cdot\nabla\right]\boldsymbol{v} = \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{b}, \tag{2.2}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\boldsymbol{b}$ is the body force.

An additional constraint for the ALE approach is that the mesh velocity satisfy the Geometric Conservation Law (GCL) [73, 114, 133]:

$$\frac{\partial V^{ce}}{\partial t} + \nabla\cdot\boldsymbol{v}^m = 0, \tag{2.3}$$

where V^{ce} is the volume of a control element. The GCL requires the change in volume of each control volume between two adjacent time steps equal the volume swept by the cell boundary during the time step.

The two predominant solution techniques are the finite-element method (FEM) in which the functional form of the solution to these equations is expanded in terms of a predetermined basis set and its residual minimized, and the finite-volume method (FVM). The structural solver in this work uses the FEM, while the flow solver uses the FVM. In the FVM, the computational domain is divided into a set of discrete volumes δV_i which fill the computational domain D without overlap. The fluid-flow equations are then volume integrated over each individual finite volume δV_i. Gauss's theorem is used to convert the divergence terms in Equations 2.1 and 2.2 into surface-integrated flux terms, reducing the problem of discretizing these terms to one of finding difference approximations for the fluxes at the surface of the control volume based on the known cell-center values.

Application of the constitutive relationships then provides the necessary closure of the governing equations. The constitutive relationships and resulting equations for the fluid and solid domains and details of the equation solvers are

provided next, followed by the procedure used to couple these domains at the interface $\Gamma_{F/S}$ and details of the implementation.

2.2 Flow Solver

Models for Newtonian fluids undergoing incompressible flow often make use of the following approximation to the stress tensor:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{S}, \tag{2.4}$$

where p is the thermodynamic pressure, μ is the absolute viscosity, and $\boldsymbol{S}$ is the strain-rate tensor. Substitution of Equation 2.4 into the momentum equation (Equation 2.2) and using $2\nabla \cdot \boldsymbol{S} = \nabla^2\boldsymbol{v}$, yields the Navier-Stokes equations:

$$\frac{\partial \boldsymbol{v}}{\partial t} + \left[\left(\boldsymbol{v} - \boldsymbol{v}^m\right) \cdot \nabla\right] \boldsymbol{v} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{v}, \tag{2.5}$$

where ν is the kinematic viscosity.

The motion of the fluid mesh is often considered a third field of the otherwise two-field fluid–structure problem because its solution is not trivial. Generally the mesh motion is computed using one of three approaches: 1) use a spring analogy where all point-to-point mesh connections are replaced with springs, 2) cast the mesh as a pseudo solid, or 3) model the mesh motion with the Laplace operator as described by Jasak and Tuković [70]. Jasak and Tuković have implemented their Laplacian approach to mesh motion in OpenFOAM. The approach involves first a decomposition of OpenFOAM's arbitrary polyhedral mesh into tetrahedral elements that are then moved according to the Laplace equation:

$$\nabla \cdot \left(\gamma \nabla \boldsymbol{v}^m\right) = 0, \tag{2.6}$$

where γ is the diffusion coefficient that can be constant or variable throughout the fluid domain. Within OpenFOAM, the standard options for a variable diffusion coefficient are 1) inversely proportional to the distance from the moving boundary, or 2) proportional to the density of the deformation energy. Further discussion of mesh motion is provided below in Section 2.4.

OpenFOAM is the flow solver of choice for this effort because it facilitates custom integration with third-party solvers, has a pre-existing, robust mesh motion capability that satisfies the GCL, and its source code is freely available through the GNU General Public License. OpenFOAM is an object-oriented library for numerical simulations in continuum mechanics, written in the C++ language. OpenFOAM does have some finite element capability, but it is best known for its cell-centered finite volume solvers. OpenFOAM versions 1.4.1-dev and 1.5-dev have been used for this research. However, all FSI and other flow simulation results reported in this thesis are from version 1.5-dev.

2.2.1 SimpleFoam

The flow problems to be modeled in this work are treated as incompressible and steady (the FSI problem is quasi-steady, as described below, but the flow field is treated as steady), and therefore OpenFOAM's `simpleFoam` solver provides a good starting point for the solver development. Because OpenFOAM employs a segregated approach to solve the coupled continuity and momentum equations (Equations 2.1 and 2.2), which requires equations to be formulated for each dependent variable and solved sequentially, an iterative scheme is required to solve the systems of equations. The `simpleFoam` solver uses the SIMPLE (Semi Implicit Method for Pressure Linked Equation) algorithm to solve for the pressure and velocity fields. The basic idea of the SIMPLE algorithm is to compute a pressure derivative $\partial p/\partial x_i$ such that the flow field is divergence-free. The procedure is:

1. compute velocity field from momentum equations (Equations 2.2) using an assumed pressure field,

2. compute pressure from the Poisson equation (Equation 2.7) using the previously computed velocity field,

3. correct the velocity field using the new pressure field, and

4. repeat steps 2 and 3 until the velocity field is divergence free.

The Poisson equation is derived by first applying the divergence to the momentum equation and then making use of continuity to eliminate terms to arrive at the final Poisson equation:

$$\nabla^2 p = \rho \frac{\partial}{\partial x_j} \left[\frac{\partial \left(u_i u_j \right)}{\partial x_i} \right] = f\left(u_i \right). \tag{2.7}$$

2.2.2 MRFSimpleFoam

`MRFSimpleFoam` is an extension of `simpleFoam` such that it enables rotating frames of reference by including centrifugal and Coriolis body-force components in Equations 2.2. This capability is important for the rotating impeller simulations. The body-force contribution to Equations 2.2 is

$$\boldsymbol{b} = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}) + 2\boldsymbol{\omega} \times \boldsymbol{v}, \tag{2.8}$$

where $\boldsymbol{\omega}$ is the rotation rate and $\boldsymbol{r}$ is the radial distance measured from the axis of rotation. The first term on the right-hand side is the centrifugal acceleration and the second term is the Coriolis acceleration.

2.3 Structural Solver

Traditionally, the finite element approach is employed for structural modeling while the finite volume (FV) approach is used for flow modeling. Development of the FE approach for structural modeling was underway in the early 1960's [8]. The ability to employ simple linear constitutive relationships for solids with wide applicability to real-world problems made feasible the early development of the structural FE approach. Fluid models, however, require the solution of non-linear equations and a much more complicated constitutive relationship to model turbulent flow and thus lagged the FE structural solver development because of computer speed and memory capacity limitations [30]. The FV approach for fluids evolved from the finite difference approach in the early 1970's, with the FV approach cited to offer some advantages over the FE method for convective-dominant equations and the FE approach offers some advantages over the FV approach for diffusion-dominant equations [65]. However, Idelsohn and Oñate [65] have shown that the FV and FE approaches have many commonalities and for some cases they are completely equivalent. Recently, there has been interest in FV structural modeling [30, 37, 71, 91, 125, 137, 147] because it would enable both flow and structural modelling to occur with a single discretization scheme [37]. Such an approach is not necessary for the partitioned approach to FSI, and many researchers still rely on FV fluid solvers and FE structural solvers. The author has chosen to implement a FE solver for the structural modeling in this work, based solely on individual preference.

The structural FE solver developed for this research uses the h-method (in contrast to the p-method or more recently the h/p-method) of finite element modeling, which means the element interpolations are of fixed polynomial order and only through mesh refinement can the solution accuracy be improved [8, 23]. This is the traditional approach to FE structural modeling and provides the most straight-forward implementation. The solver is implemented with a Lagrangian

frame-of-reference, which means the mesh velocity is equivalent to the material velocity, $\boldsymbol{v}^m = \boldsymbol{v}$ and hence the momentum equation (Equation 2.2) becomes:

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b}, \tag{2.9}$$

where $\boldsymbol{u}$ are the material displacements ($\partial \boldsymbol{u}/\partial t = \boldsymbol{v}$). As is described later in Section 4.3, the structural response can be treated as quasi-steady due to the long time scale associated with material stress relaxation and negligible inertial terms of Equation 2.9. Therefore, the momentum equation for the solid reduces to

$$\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} = 0, \tag{2.10}$$

with the boundary conditions:

$$\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{t}) = \boldsymbol{u}^p(\boldsymbol{x}, \boldsymbol{t}) \qquad \text{on surface} A_u \tag{2.11}$$

$$\boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{t}) \cdot \boldsymbol{n}(\boldsymbol{x}, \boldsymbol{t}) = \boldsymbol{t}^p(\boldsymbol{x}, \boldsymbol{t}) \qquad \text{on surface} A_t \tag{2.12}$$

where A_u is the constrained portion of the boundary surface, A_t is the portion of the boundary surface subject to an external traction, t^p is the applied traction, and $\boldsymbol{n}$ is the unit boundary surface normal.

The finite element procedure for structural mechanics is well-documented in a number of texts [8, 23, 24, 62], and is therefore only summarized herein. The basic approach is to rewrite the *strong form* of the stress equilibrium equation (Equation 2.10) in its *weak form*,[2] apply integration by parts, use the divergence

[2] Governing differential equations and boundary conditions represent the strong form of a problem, whereas the weak form is an integral expression of the problem that implicitly contains the differential equations.

theorem, and then incorporate the Bubnov-Galerkin (most often referred to as simply the *Galerkin*) method which implies the weak form's weighting function has the same form as the nodal interpolation scheme for the trial solution. The process is summarized in Figure 2.1, which shows the equivalence of the weak and strong forms, the use of approximation between the weak and Galerkin form, and the equivalence in the Galerkin and matrix forms.

Fig. 2.1. Finite element approach sequence

2.3.1 Small-Deformation Formulation

The resulting matrix equation for a single finite element of the small-deformation formulation is [8]:

$$\boldsymbol{R}(\boldsymbol{u}) = \sum_{V_{GP}} \left[\boldsymbol{B}^T \bar{\boldsymbol{\sigma}} - N^T \boldsymbol{b} \right] wJ - \sum_{A_{GP}} N^T t^p wj, \tag{2.13}$$

where $\boldsymbol{R}$ is the element force residual that is non-zero when the trial solution $\boldsymbol{u}$ differs from the exact solution (this represents the imbalance between the externally applied loads and the internal element reaction forces), $\boldsymbol{B}$ is the strain-displacement matrix, $\bar{\boldsymbol{\sigma}}$ is the stress tensor in vector form, w is the Gauss-point weight for the Gaussian quadrature, J is the volume-to-volume Jacobian relating reference and physical element volumes, and j is the face-to-face Jacobian. The strain-displacement matrix is determined by differentiation of the shape function

matrix (N) and relates strains ($\boldsymbol{\varepsilon}$) to nodal displacements ($\boldsymbol{u}$):

$$\bar{\boldsymbol{\varepsilon}} = \boldsymbol{B}\boldsymbol{u} \tag{2.14}$$

where $\bar{\boldsymbol{\varepsilon}}$ is the engineering strain:

$$\bar{\boldsymbol{\varepsilon}} = \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{31} \end{Bmatrix} \tag{2.15}$$

The strain-displacement matrix is computed by differentiating the shape function matrix. The shape functions enable interpolation of nodal values (e.g., $\boldsymbol{u}^k$) to a location within the element:

$$\boldsymbol{u} = \sum_k N^k \boldsymbol{u}^k, \tag{2.16}$$

where N^k is a function of the element reference coordinates $N^k(\xi, \eta, \zeta)$ shown in Figure 2.2 for a hexahedral element, and k is a counter over all nodes of the element. The present work uses only isoparametric elements, which means the same shape functions interpolate the solution variables and the element geometry. The shape

functions used for the eight-node hexahedron are:

$$\begin{aligned}
N^1 &= -\ (\xi-1)(\eta-1)(\zeta-1)/8 \\
N^2 &= \quad\ (\xi+1)(\eta-1)(\zeta-1)/8 \\
N^3 &= -\ (\xi+1)(\eta+1)(\zeta-1)/8 \\
N^4 &= \quad\ (\xi-1)(\eta+1)(\zeta-1)/8 \\
N^5 &= \quad\ (\xi-1)(\eta-1)(\zeta+1)/8 \\
N^6 &= -\ (\xi+1)(\eta-1)(\zeta+1)/8 \\
N^7 &= \quad\ (\xi+1)(\eta+1)(\zeta+1)/8 \\
N^8 &= -\ (\xi-1)(\eta+1)(\zeta+1)/8
\end{aligned} \tag{2.17}$$

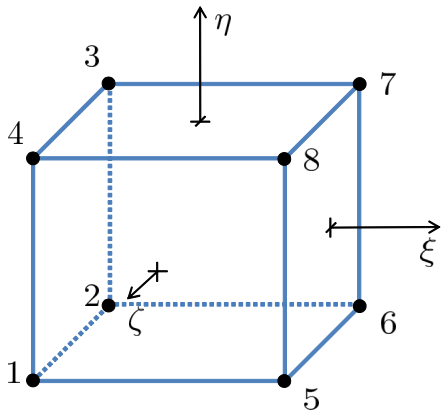

Fig. 2.2. Reference element for the eight node hexahedron

The stress tensor is computed from the material constitutive relationship using the fourth-order material stiffness tensor, $\mathcal{C}$, as follows:

$$\boldsymbol{\sigma} = \mathcal{C} : \boldsymbol{\varepsilon}. \tag{2.18}$$

The stress tensor in engineering format (using Voight notation) is

$$\bar{\boldsymbol{\sigma}} = \left\{ \begin{array}{c} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{array} \right\} \tag{2.19}$$

which is related to the engineering strain for linear elastic materials using $\bar{\boldsymbol{\sigma}} = C\bar{\boldsymbol{\varepsilon}}$, where C is the material stiffness matrix.

$$C = \frac{E}{(1+\nu)(1-2\nu)} \left[\begin{array}{cccccc} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-\nu \end{array} \right], \tag{2.20}$$

where E is the Young's modulus and ν is Poisson's ratio.

The Jacobian, J, is the determinant of the Jacobian matrix, $\boldsymbol{J}$, and represents a scale factor relating physical dimensions to reference-element (Figure 2.2) dimensions:

$$\boldsymbol{J} = \left[\begin{array}{ccc} x_{,\xi} & y_{,\xi} & z_{,\xi} \\ x_{,\eta} & y_{,\eta} & z_{,\eta} \\ x_{,\zeta} & y_{,\zeta} & z_{,\zeta} \end{array} \right], \tag{2.21}$$

where the subscript following the comma implies differentiation with respect to that variable.

The global residual, $\mathcal{R}$, is determined by summing the force residual for all elements (the *direct stiffness method* [8]):

$$\mathcal{R}(\mathcal{U}) = \sum_{elements} \boldsymbol{R}(\boldsymbol{u}), \tag{2.22}$$

where $\mathcal{U}$ is the global displacement vector.

A Newton-Raphson approach is employed to solve the system of equations, which requires an initial guess of the global solution vector, $\mathcal{U}^i$ for $i = 0$, and a tangent to compute the next guess of the solution vector:

$$\delta\mathcal{U} = -\left[\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}\mathcal{U}}\bigg|_{\mathcal{U}^i}\right]^{-1} \mathcal{R}(\mathcal{U}^i), \tag{2.23}$$

and then the next guess of the solution is:

$$\mathcal{U}^{i+1} = \mathcal{U}^i + \delta\mathcal{U}. \tag{2.24}$$

Typically the first guess of the global solution vector is zero displacement for all degrees of freedom.

The tangent stiffness is derived by differentiating Equation 2.22 by $\mathcal{U}$ or equivalently for all elements Equation 2.13 with respect to the solution vector $\boldsymbol{u}$:

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{u}} = \sum_{V_{GP}} \left[\boldsymbol{B}^T \frac{\mathrm{d}\bar{\boldsymbol{\sigma}}}{\mathrm{d}\bar{\boldsymbol{\varepsilon}}} \boldsymbol{B} - N^T \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{u}}\right] wJ - \sum_{A_{GP}} N^T \frac{\mathrm{d}t^p}{\mathrm{d}\boldsymbol{u}} wj, \tag{2.25}$$

The iterative procedure described by Equations 2.23 and 2.24 continues until the global residual is sufficiently small. The convergence criterion implemented for the present work is that the solution is considered converged when the Euclidean

norm of $\mathcal{R}$ is less than 0.001 times the net applied load magnitude. Linear analyses require a single solution step to satisfy this criterion.

The numerical integration over the element volumes in Equations 2.13 and 2.25 employ Gauss quadrature, which uses sampling points with assigned weights chosen to minimize integration error when the integrand is a general polynomial. By default, the hexahedron element used in this work employs points and weights such that the integration is considered *full*. Full integration, as defined by Cook [23], is a quadrature rule of sufficient accuracy to exactly integrate the stiffness coefficients of an *undistorted*, i.e., rectangular, element. For a distorted element the quadrature is not exact. The sampling points for the hexahedral element are $\xi, \eta, \zeta = \pm 1/\sqrt{3}$ for a total of eight points. The corresponding weights w are all unity.

Unfortunately, for the standard shape functions of Equations 2.17, full element integration, and a nearly incompressible material, the elements have a tendency to *lock* during a bending deformation causing them to appear overly stiff. This phenomenon is a result of spurious strains in the element that require much greater energy input than do the physically correct strains (see Cook [23] for additional information). Two approaches are available to address this issue. The first is to decompose the stress tensor into volumetric and deviatoric components and solve for the displacements (from the deviatoric stress) and the pressure (volumetric stress) separately (see [8]). This approach is referred to as a *mixed* formulation [62] or equivalently a pressure/displacement formulation [8]. An alternative approach is to employ selectively-reduced integration, where the volumetric and deviatoric stress components are integrated differently. Applying reduced integration, using a single quadrature point at parameterized location (0,0,0) and weight of 2 to the volumetric terms, and full integration to the deviatoric terms, the equivalence of the reduced integration and hybrid techniques can be shown [62]. The reduced

integration hexahedral elements are implemented in the FE solver and their performance is demonstrated during the solver validation study described later.

Note that the reduced-integration elements were pursued subsequent to initial FSI simulations using highly refined structural meshes. The dense meshes were required to reduce the locking effects caused by the nearly-incompressible material. After pursuing quadratic elements and a parallelized FE solver using the PetSc solver libraries [5, 6] as alternative avenues to reduce the structural solve times, the reduced order elements were implemented and enabled dramatic increases in mesh coarseness while still maintaining accurate structural response.

The equations for the small-deformation finite element formulation were implemented first into a solver with the intent to expand to a large-deformation formulation if later required. The need for a large-deformation formulation became apparent during the Hapflex beam and fin testing that is described later in Section 3.5.1. A large-deformation formulation is required when the deformation is large enough to alter the applied loads or the internal resisting forces and moments [23]. In practice, the solution is computed using both approaches to determine if the more costly large-deformation formulation is required. The large-deformation formulation implemented here follows the approach described by Bath [8] and Hibbett et al. [58]. The formulation is briefly summarized next.

2.3.2 Large-Deformation Formulation

The large-deformation FE formulation differs primarily from the small-deformation formulation due to geometric nonlinearity that arises from large deformations. The geometric nonlinearity must be accounted for when the deformations become large enough that the equilibrium equations must be written for the deformed configuration. Also, loads that *follow* the geometry will change direction as the structure deforms. With this formulation, the geometry for which the equilibrium equations are formed is not known a-priori and thus an incremental type

solution is used where the nonlinear solution is built up as a series of linear increments [58]. As described by Bathe [8], an effective incremental solution approach requires appropriate stress and strain measures. It is the purpose of this section to present the definitions of these measures. Note that for the present application as is shown later, the strain levels are small enough to use a linear constitutive relationship, thereby allowing the use here of the material model described for the small-deformation formulation.

The Total Lagrangian (TL) formulation is implemented for this work, which means all variables are referred to the initial configuration (occurring at time 0). In contrast to the TL formulation is the Updated Lagrangian (UL) formulation, wherein all variables are referred to the last-calculated configuration. The difference between the two methods lies only in their relative numerical efficiency [8]. The TL method generally requires more memory but less computation than the UL method because spatial derivatives are with respect to a fixed reference frame for TL and thus only need to be computed one time. The UL method requires updated derivatives each iteration because the reference configuration changes. Note that nearly all commercial solvers employ the UL method, but mostly because of historical reasons. When originally developed, memory was scarce and expensive and thus computational speed was sacrificed in favor of reduced memory requirements [98].

A notional schematic of the reference and current configurations is shown in Figure 2.3. This figure shows a general body undergoing a large deformation from a base configuration $\mathcal{B}_0$ to a deformed configuration $\mathcal{B}$. A position $\boldsymbol{X}$ in $\mathcal{B}_0$ changes to the position $\boldsymbol{x}$ in $\mathcal{B}$ leading to the following relationship:

$$\boldsymbol{x} = \boldsymbol{X} + \boldsymbol{u} \tag{2.26}$$

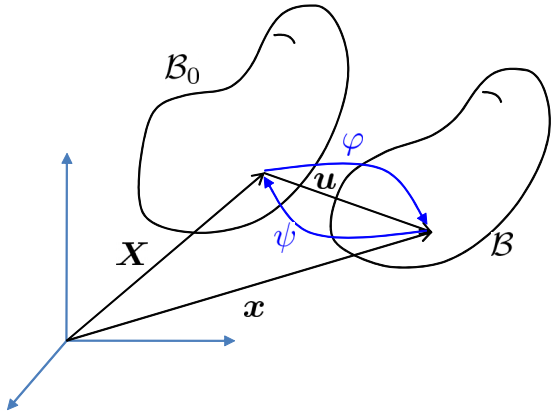

Fig. 2.3. Motion of a general solid body

The key difference in the large-deformation formulation over the linear finite element approach is the need for a strain measure that is invariant to rigid body rotations. One such strain measure is the Green-Lagrange strain, $\boldsymbol{\varepsilon}$. The definition of $\boldsymbol{\varepsilon}$ is usually stated in terms of the deformation gradient F:

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{bmatrix} \tag{2.27}$$

The Green-Lagrange strain, usually referred to as the Green strain, is defined as:

$$\boldsymbol{\varepsilon} = \left(\boldsymbol{F}^T \boldsymbol{F} - \boldsymbol{I} \right), \tag{2.28}$$

where $\boldsymbol{I}$ is the second rank unity tensor. Alternatively, the Green strain can be written

$$\boldsymbol{\varepsilon} = \frac{1}{2} \left[\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T + \nabla \boldsymbol{u} \cdot (\nabla \boldsymbol{u})^T \right], \tag{2.29}$$

to highlight its nonlinearity when compared to the small-strain relation used in linear analyses

$$\boldsymbol{\varepsilon} \approx \frac{1}{2}\left[\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right]. \tag{2.30}$$

Another important variable is the displacement gradient, $\boldsymbol{D}$

$$\boldsymbol{D} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial u_1}{\partial X_1} & \frac{\partial u_1}{\partial X_2} & \frac{\partial u_1}{\partial X_3} \\ \frac{\partial u_2}{\partial X_1} & \frac{\partial u_2}{\partial X_2} & \frac{\partial u_2}{\partial X_3} \\ \frac{\partial u_3}{\partial X_1} & \frac{\partial u_3}{\partial X_2} & \frac{\partial u_3}{\partial X_3} \end{bmatrix} \tag{2.31}$$

which can also be related to $\boldsymbol{F}$ as follows

$$\boldsymbol{F} = \boldsymbol{I} + \boldsymbol{D}.$$

A suitable stress measure must also be used for the large-deformation formulation. The requirement is that the stress measure must be a *work conjugate* of the strain measure. For the Green strain, the work-conjugate stress is the *second Piola-Kirchhoff* stress, $\boldsymbol{S}$. While this stress measure is used in the finite element equations below, it has little physical meaning [8] and must be transformed to the Cauchy stress $\boldsymbol{\sigma}$ when interpreting computed stresses:

$$\boldsymbol{\sigma} = J_{_F} \boldsymbol{F} \boldsymbol{S} \boldsymbol{F}^T, \tag{2.32}$$

where $J_{_F}$ is the Jacobian of the deformation gradient:

$$J_{_F} = \det\left(\boldsymbol{F}\right). \tag{2.33}$$

The element residual equation for the large-deformation formulation is similar to that for the small-deformation formulation (Equation 2.13), but uses a nonlinear strain-displacement matrix $\boldsymbol{B}_{nl}$ and the second-Piola-Kirchhoff stress $\boldsymbol{S}$ as follows

$$\boldsymbol{R}(\boldsymbol{u}) = \sum_{V_{GP}} \left[\boldsymbol{B}_{nl}{}^{T} \bar{\boldsymbol{S}} - N^{T} \boldsymbol{b} \right] wJ - \sum_{A_{GP}} N^{T} t^{p} wj, \tag{2.34}$$

where $\bar{\boldsymbol{S}}$ is the second-Piola-Kirchhoff stress written in vector form.

The non-linear strain-displacement is composed of linear and nonlinear components

$$\boldsymbol{B}_{nl} = \boldsymbol{B}_{l} + \boldsymbol{B}_{n}$$

such that the Green strain (in engineering form) is computed as

$$\bar{\boldsymbol{\varepsilon}} = \boldsymbol{B}_{l}\boldsymbol{u} + \frac{1}{2}\boldsymbol{B}_{n}\boldsymbol{u}.$$

The large-deformation formulation follows the same solution procedure as defined above for the small-deformation formulation (see Equations 2.23 and 2.24) and thus a tangent stiffness matrix is required. This is computed as follows:

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{u}} = \sum_{V_{GP}} \left[\boldsymbol{B}_{nl}{}^{T} \frac{\mathrm{d}\bar{\boldsymbol{S}}}{\mathrm{d}\bar{\boldsymbol{\varepsilon}}} \boldsymbol{B}_{nl} + \boldsymbol{G}^{T} \tilde{\boldsymbol{S}} \boldsymbol{G} - N^{T} \frac{\mathrm{d}\boldsymbol{b}}{\mathrm{d}\boldsymbol{u}} \right] wJ - \sum_{A_{GP}} N^{T} \frac{\mathrm{d}t^{p}}{\mathrm{d}\boldsymbol{u}} wj, \tag{2.35}$$

where $\boldsymbol{G}$ computes the displacement gradient $\boldsymbol{D}$ in vector form

$$\bar{\boldsymbol{D}} = \boldsymbol{G}u$$

and $\tilde{\boldsymbol{S}}$ is a reorganized form of the second-Piola-Kirchhoff stress

$$\tilde{\boldsymbol{S}} = \begin{bmatrix} \boldsymbol{S} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{S} \end{bmatrix},$$

where

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

2.3.3 Solver Implementation

The finite element formulation described above has been implemented using the C++ object-oriented programming language. While the primary classes created were not meant to use all aspects of the language, such as inheritance and polymorphism, the ability to create objects with custom methods (i.e., member functions) in a highly modular fashion provided a user-friendly development environment. Moreover, the use of a programming language consistent with that of the flow-solver simplifies the FSI solver integration. The current section describes some of the class's member functions and provides a general overview of the stand-alone structural solver called `feanl` for finite element analysis, non-linear. Later while describing the FSI solver implementation, the member functions described here will be referenced. The implementation of the class `Structure` required slightly over 18,000 lines of source code with over 3,000 lines of comments.

The member functions required for the FSI solver are listed below with a brief description. Additional information is subsequently provided where deemed necessary.

`loadModel` loads the finite element model from a file on disk. The file format mimics that of the Abaqus input file format for ease of communicating with existing pre- and post-processors. See Appendix A for a sample input file.

`init` initialize the solver to allocate all memory and perform node and element mappings for the element definitions, loads, and constraints. This is separated from the `loadModel` member function so not all compute nodes in a parallel processing paradigm need to allocate memory (this point is further discussed in Section 2.5.4).

`loadedFaceInformation` provides face location and face normal for all faces subject to a distributed load (e.g., the wetted faces of an FSI simulation). This allows the user to specify which faces reside on the fluid/structure interface, $\Gamma_{F/S}$, to facilitate the interface mapping algorithm discussed in Section 2.5.3.

`deleteCurrentLoad` deletes all existing externally applied concentrated forces and distributed (traction) loads. This is necessary to remove the distributed loads referenced by `loadedFaceInformation` when identifying $\Gamma_{F/S}$.

`createConcentratedLoad` creates a nodal force load set. All nodes comprising the faces on $\Gamma_{F/S}$ are assigned one or more force vectors that are later modified during the FSI simulation.

`setConcentratedLoadComponent` sets the force vectors (magnitude and direction) of a concentrated load set created using `createConcentratedLoad`.

`setEndTime` sets the simulation end time. This enables the flow solver to control the FSI simulation, with the ability to perform sub-iterations for tightly-coupled FSI simulations.

`solve` solves the structure from the previously solved state to the current end time as specified by `setEndTime`.

`elementParameterizedLocation` finds the location in the reference element space corresponding to physical space. This is necessary to interpolate displacement results to physical locations for the flow field's mesh motion solver.

`elementInterpolateDisplacement` interpolates displacement vectors to a physical location characterized by reference-element coordinates. Later it will be shown that the physical coordinates of interest here are those corresponding to the flow solver's vertices that reside on $\Gamma_{F/S}$.

`saveCurrentState` saves the current state of the structure (displacements, stresses, strains, output stream pointers, etc.). This is required for sub-iterations for a tightly-coupled FSI solver.

`resetState` resets the state of the structure (displacements, stresses, strains, output streams, etc.) to the state corresponding to the `saveCurrentState` function call. This is required for sub-iterations for a tightly-coupled FSI solver.

`setNodeDisplacements` sets the global displacement vector using values from a file. This is useful for the inverse analysis (described later) to move the fluid mesh to avoid manual mesh generation for the flow domain.

The member functions `solve` and `elementParameterizedLocation` warrant additional discussion. The `solve` member function performs the matrix and vector assembly for each iteration of the Newton-Raphson scheme (Equation 2.23). In general, the system matrix is very sparse and non-symmetric (but it is structurally symmetric). To facilitate the matrix assembly, a sparse matrix class was implemented that provides member functions for constructing and accessing the matrix elements by specifying either the row and column index or the sparse index location to enable fast matrix assembly. The reason for using a separate sparse matrix object was to simplify parallelization in the future, with for example

PetSc [7, 5, 6]. Once the tangent stiffness matrix and the global residual vector are assembled, a linear solver is called to compute the increment in the global displacements, $\delta\mathcal{U}$. The primary solver used for this work is the Intel®Math Kernel Library (MKL) Direct Sparse Solver (DSS) [66]. Validation of `feanl` is provided later in Section 5.2.

2.3.4 Inverse Formulation

The purpose of the inverse finite element analysis is to solve for the reference configuration (unknown) that would be required by the current configuration (known) to support prescribed load conditions (see Figure 2.3). Previous work, most recently by Fachinotti et al. [36], involved re-casting the finite element formulations in terms of the unknown configuration $\boldsymbol{X}$ instead of $\boldsymbol{x}$ (Figure 2.3). A similar approach was followed by other researchers but for different materials than required here [52, 53].

While starting to implement an extension of the work by Fachinotti et al. for hyper-elastic materials to perform the inverse analysis for a viscoelastic material, it was realized that the incremental formulation employed for large-deformation finite element analysis is based on a series of linear increments (see Section 2.3.2). Given the use of linear increments, it makes no difference whether the current configuration is changed during a conventional *forward* analysis or if the 'reference' configuration is changed by application of the displacements in the opposite sense. The only requirement is that the derivatives of the TL method be updated each increment. It turns out that this approach works very well for the problems considered here (see Section 5.3), and should be general for all such inverse problems. The inverse technique as described here was implemented into a second solver called `ifeanl` for inverse finite element analysis, non-linear. The inverse solver has all of the elements of the forward solver, but instead of only incrementing the displacement vector for the next guess at a solution (Equation 2.24), the reference

configuration is modified:

$$\boldsymbol{X}^{i+1} = \boldsymbol{X}_0 - \mathcal{U}^{i+1}, \tag{2.36}$$

where $\boldsymbol{X}_0$ is the original reference configuration, and all derivatives are updated each solution increment.

The limitations of this solver with respect to the approach employed by others should be evaluated in the future. For the present application, the inverse solver is shown to replicate well the inverse configuration. Validation of the inverse solver is provided later in Section 5.3.

2.3.5 Constitutive Relationships

A variety of constitutive relationships are commonly used for structural mechanics problems, especially when the models involve elastomeric materials. A linear-elastic material model with a large-deformation formulation was employed for this research. While both the small- and large-deformation formulations were implemented, only the large deformation formulation is described here in detail. The constitutive model used is linear-viscoelastic, and is implemented as a modification to a linear-elastic model. The Cauchy stress tensor for a linear-elastic solid is defined as follows:

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda\nabla\cdot\boldsymbol{u}\boldsymbol{I}, \tag{2.37}$$

where $\mu = E/2(1+\nu)$ and $\lambda = \nu E/(1+\nu)(1-2\nu)$ are Lamé's constants, $\boldsymbol{I}$ is the second rank unity tensor, and $\boldsymbol{\varepsilon}$ is the Green-Lagrange strain tensor (Equations 2.28 and 2.29).

The introduction of material viscoelasticity to the linear material model requires a nonlinear solution approach, as described below. The material viscoelasticity is modeled using a time-domain approach, similar to that used by the commercial software Abaqus (see the Abaqus User's Manual [57]) and derived in a similar manner by Kaliske and Rothert [72]. This model is for linear viscoelastic

materials, which does not mean the time response of the material is linear, but rather the stress is proportional to strain at any given time $\varepsilon\left[c\sigma(t)\right] = c\varepsilon\left[\sigma(t)\right]$, where c is a constant [44], and uses the approximation that shear and volumetric behavior are independent. Furthermore, the viscoelastic behavior is dominated by the deviatoric part of the material's deformation and therefore the shear and volumetric terms must be isolated [57, 72].

The underlying material model for this approach is the Generalized Maxwell Element, which consists of Maxwell elements (i.e., a spring and dashpot in series) in parallel with a Hooke element (i.e., a spring) as shown in Figure 2.4. The spring

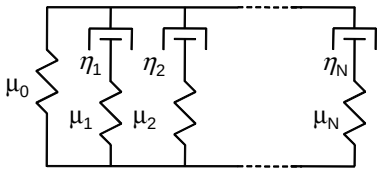

Fig. 2.4. Generalized Maxwell Element with N components

stiffness μ_0 shown in this figure represents the material stiffness at infinite time (i.e., after all of the viscoelastic forces have diminished to zero). Each Maxwell element, defined by a stiffness μ_i and a viscosity η_i, represents a different time scale of the material's response and comprises a term in a Prony series (a series of the form $\sum_{i=1}^{N} \alpha_i e^{-t/\tau_i}$) representation of the material.

Decomposition of the Cauchy stress is required for the viscoelastic model to effect only the shear component. The stress can be decomposed into hydrostatic and deviatoric components as follows:

$$\boldsymbol{\sigma}^{n+1} = \kappa \text{ tr } \boldsymbol{\varepsilon}^{n+1}\boldsymbol{I} + \text{dev } \boldsymbol{\sigma}^{n+1}, \tag{2.38}$$

where κ is the bulk modulus ($\kappa = \lambda + 2\mu/3$ for linear elastic materials). The deviatoric part of the material stress tensor for the current $(n+1)$ time step then takes on the following form:

$$\text{dev } \boldsymbol{\sigma}^{n+1} = \text{dev } \boldsymbol{\sigma}_0{}^{n+1} + \sum_{i=1}^{N} \boldsymbol{h}_i{}^{n+1}, \tag{2.39}$$

where $\boldsymbol{h}_i$ are the internal stress variables that come from the so called *heredity integral* [72]:

$$\boldsymbol{h}_i(t) = \int_o^t \gamma_i e^{-\frac{t-s}{\tau_i}} \frac{\partial \boldsymbol{\sigma}_0(s)}{\partial s} \text{ds}, \tag{2.40}$$

where γ_i are the normalized relaxation constants, τ_i are the relaxation times, and $\boldsymbol{\sigma}_0$ is the stress associated with the Hooke element of the Generalized Maxwell Element (Figure 2.4). The relaxation constants and times are defined by fitting a Prony series constructed with these parameters to empirical data, often using a least-square error approach. Splitting this integral into parts that are known (i.e., time period [0, t^n]) and unknown (i.e., time period (t^n, t^{n+1}]), and using the approximation $\frac{\partial \boldsymbol{\sigma}_0(t)}{\partial t} \approx \frac{\boldsymbol{\sigma}_0{}^{n+1} - \boldsymbol{\sigma}_0{}^{n}}{\Delta t}$, the internal stress variables at the next time step are approximated as follows:

$$\boldsymbol{h}_i{}^{n+1} \approx e^{-\frac{\Delta t}{\tau_i}} \boldsymbol{h}_i{}^{n} + \gamma_i \frac{1 - e^{\frac{-\Delta t}{\tau_i}}}{\frac{\Delta t}{\tau_i}} \left[\text{dev } \boldsymbol{\sigma}_0{}^{n+1} - \text{dev } \boldsymbol{\sigma}_0{}^{n} \right], \tag{2.41}$$

where $\Delta t = t^{n+1} - t^n$ is the time step.

The Prony series parameters required to implement this model are the focus of Chapter 3.

2.3.6 Structural Body Force

The body force vector that appears in the element residual equations (Equations 2.13 and 2.34) is only required in this work for two purposes: gravity and centrifugal force (no Coriolis force is required because the fin velocities are negligible for these quasi-steady analyses). Support for both of these body force loads is included in the structural solver. Gravity loads are most important for the material model and structural solver validation studies, while the centrifugal force is important for the rotating impeller simulations.

2.4 Fluid Mesh Motion

The partitioned approach to FSI has been described by several researchers as a three-field problem [10, 39, 82, 109, 114, 115, 116, 117] because it requires the solution of the flow, structure, and fluid mesh motion, none of which is trivial. The flow solver and the structural solver are described above with a brief introduction to the mesh motion for the ALE flow solver. Additional information for the mesh motion solver is provided in the current section.

The sole purpose of the mesh motion is to enable the flexible structure to deform in response to the fluid stresses acting on its wetted surface. The fluid mesh must track the motion of the structure and should deform such that it maintains the mesh quality near the fluid/structure interface. Such an approach is referred to as *interface tracking* [120, 119] because the interface between the flow and structural domains must be tracked as the structure deforms. The main objective with any mesh motion approach is to accommodate the structure's deformation while maintaining a quality mesh.

The mesh motion strategy can be as simple as prescribing a functional relationship between a boundary displacement at the fluid/solid interface and the fluid vertices [120, 127]. This approach can work for simple problems but becomes cumbersome, if not impossible, for complicated flow geometries. For complex flow

geometries, a more general approach that represents the fluid mesh as an artificial elastic continuum has been employed by several researchers [12, 61, 114, 117]. This approach is often referred to as the pseudo-solid method, with a stiffness matrix that can be derived by treating each cell edge as a spring with the spring stiffness based on the relative edge lengths [40]. Even more demanding geometries and deformation magnitudes might require re-meshing of the flow domain [119].

Unfortunately, the spring analogy approach to moving the fluid mesh has been shown to be non-robust for various problems and modifications to the basic approach have been made to improve the performance while adding complexity and computational cost to the approach [29, 38, 70].

In an effort to improve upon the mesh motion techniques for FV codes, Jasak and Tuković [70] introduced a vertex-based, unstructured mesh motion solver (operates on the cell vertices to avoid interpolation, which can lead to cell flipping and degeneration). The approach solves the Laplace equation with variable diffusion, as shown above in Equation 2.6. The OpenFOAM development versions (including versions 1.4.1-dev and 1.5-dev) features this mesh motion solver.

While the OpenFOAM Laplace face decomposition solver performs very well for many problems and is easy to implement because only motion on the tracked interface is required, it does have limits. In particular, the author has found that in its current form it does not function properly for parallel solutions, does not support cyclic boundaries, and does not support non-Cartesian boundary surfaces (e.g., for slip along a cylindrical surface).

While the use of a parallel solver is desired to improve turn-around time of the many simulations required for this research, it is not absolutely necessary. What is necessary, however, is to be able to perform mesh motion inside cylindrical tubes. A simple test case involving a deforming stator inside a tube is shown in Figure 2.5. The tube is specified as a slip boundary condition for the mesh motion solver. The large motion, coupled with the inability to constrain the boundary

mesh to a constant radius surface, causes the unwanted radial motion of the tube boundary shown in this figure. As such, other mesh motion techniques have been explored. These are summarized next.

The Radial Basis Function (RBF) mesh motion solver, implemented in OpenFOAM by Bos [14] has been explored for this research. This solver does support parallel execution and works well for large motion, as described by Bos. However, the author is unaware of a method to use this solver for motion near a boundary with a slip constraint, similar to that required in Figure 2.5.

Faced with limited alternatives, a custom mesh motion approach was implemented using a simpler approach of specifying field motion based on the boundary motion and weighting function (e.g., a half-wave sinusoid, Figure 2.6) to non-uniformly displace the fluid vertices. The function specifies a value of unity when the field point is located on $\Gamma_{F/S}$ and a value of zero when the field point is at a maximum distance from $\Gamma_{F/S}$ as prescribed by an input parameter to the solver. The advantage of this approach is that it is easily implemented for parallel solutions (a global listing of the boundary displacements are sent to each compute node and each processor moves the nodes in its domain). Processor boundary errors are avoided by first creating a global listing of all fluid vertices, each global vertex is mapped to a boundary point from which its motion is derived, and the mapping array is sent to all processors. This way, all global points are moved based on the same boundary point and the same motion function. This results in a computationally cheap motion solver because no matrix solution is required. While this approach works well for simple geometries, complex geometries and large structural deflections yield poor-quality cells thus violating one of the requirements of the motion solver. This motion solver is used for the FSI simulations described later and is referred to in this thesis as the Radial Motion Function (RMF) solver.

An alternative mesh motion solver was developed that makes use of the finite element solver `feanl` and its ability to interpolate displacements to arbitrary

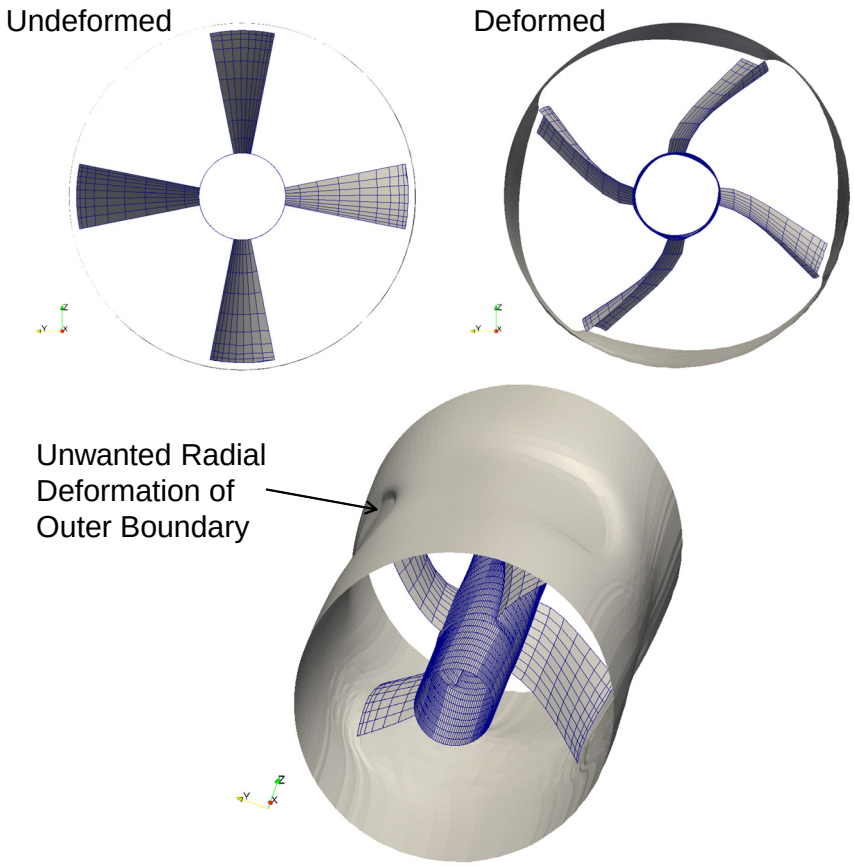

Fig. 2.5. Mesh motion for a cylindrical surface using the face decomposition solver; the initially cylindrical surface does not remain cylindrical

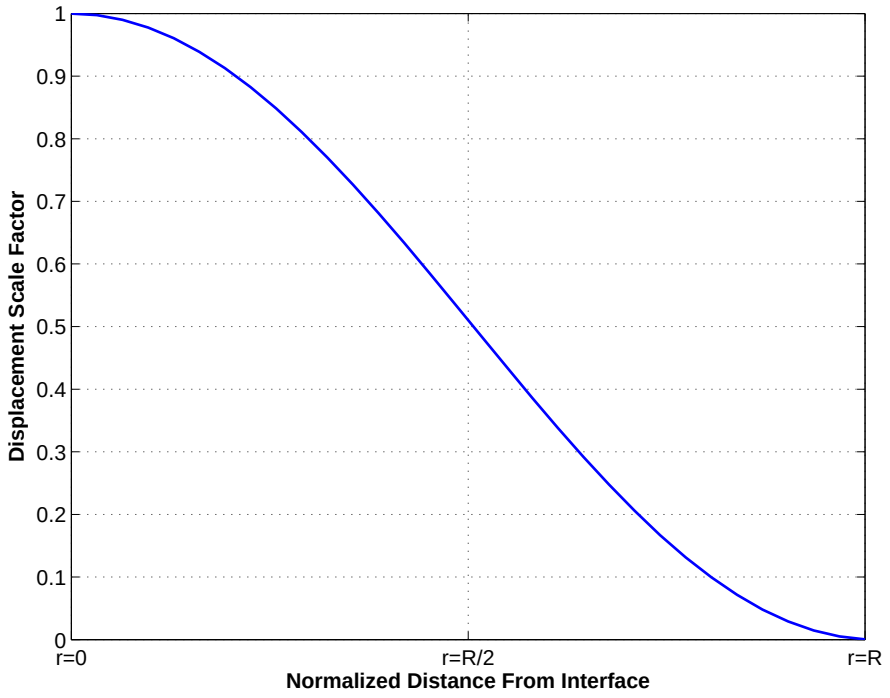

Fig. 2.6. Mesh motion weighting function as a simple alternative to the generalized mesh motion solvers in OpenFOAM

spatial locations. The approach is similar to the pseudo-solid technique described above, except that it uses an auxiliary mesh that intersects the flow mesh. Only the fluid vertices that fall within the intersected space are moved, which offers excellent control of the mesh motion. Moreover, the ability to vary element stiffness throughout the overlayed mesh allows the mesh quality near the moving surface to be maintained by causing the deformation to occur farther from the surface. Figure 2.7 shows an example fluid mesh overlayed by a coarser motion mesh. Figure 2.8 shows the same motion mesh, with the stiffened inner region colored differently than the softer outer region. This minimizes deformation of the fluid cells near the structure and forces most of the deformation to occur elsewhere. This is especially useful for rigid body rotations of the fin to change its angle of attack because the mesh quality near the fin does not change. Only when the fin deforms will the near-field mesh quality change. This is demonstrated by strain contours for the pitching fin in Figure 2.9. This mesh motion solver was implemented near the completion of this research and has been used only for large angle-of-attack fin simulations described later.

2.5 FSI Solver

As described above, the coupled problems to be solved for this research are being treated as incompressible, laminar, and steady. Typically for flow problems of this type an OpenFOAM user would employ the `simpleFoam` solver. Therefore, this solver is extended for the present work to include a dynamic mesh (i.e., the ALE formulation), the structure class `Structure`, and a method to interface the fluid and solid domains. To accomplish this task while maintaining a user-friendly program interface and reusable software that facilitates updates to the separate, stand-alone structural solver `feanl`, a separate C++ class has been created. The class is called `fsiInterface` and consists of header and source files

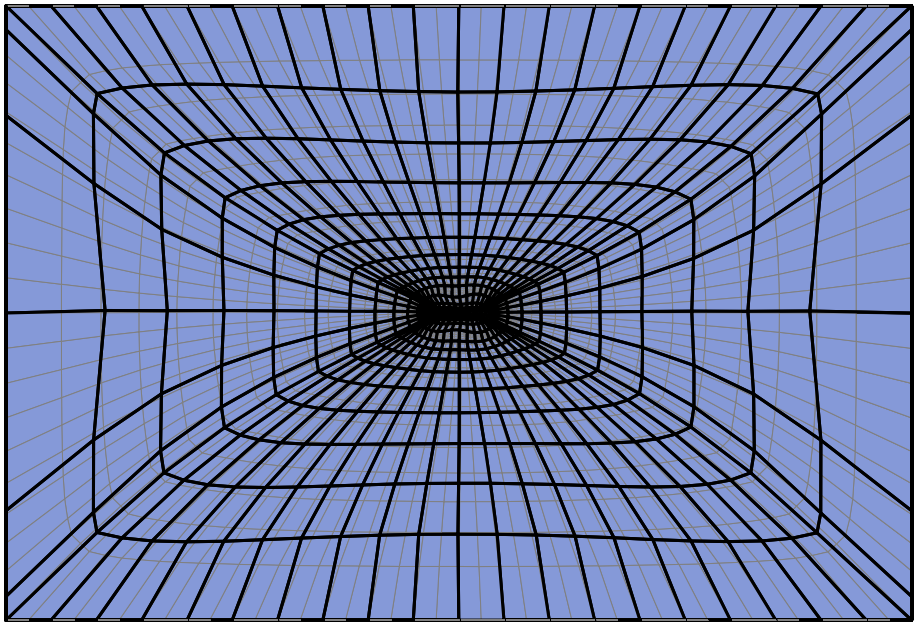

Fig. 2.7. Example of FE motion mesh (black element edges) overlaying a fluid mesh (gray element edges)

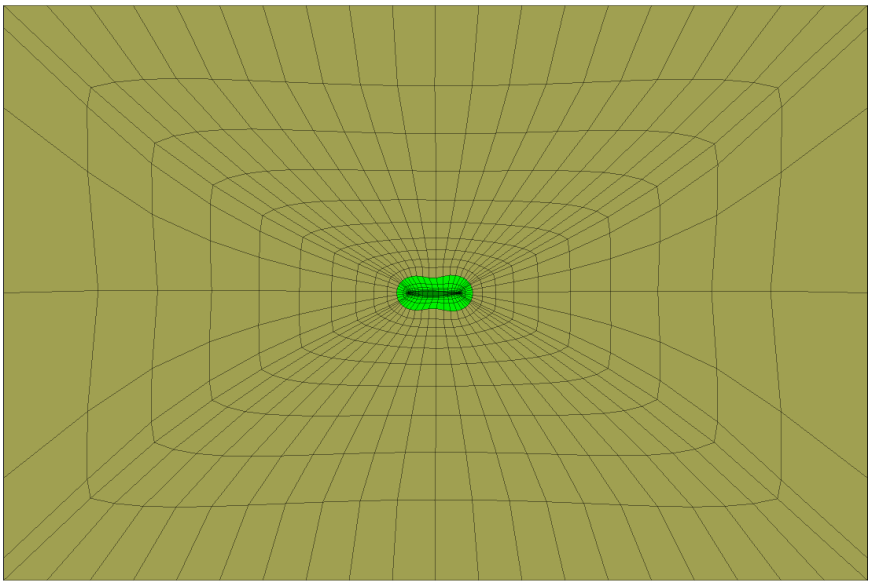

Fig. 2.8. Example motion mesh showing different regions of element stiffness to control deformation of the underlying fluid mesh elements

named `fsiInterface.h` and `fsiInterface.cpp`, respectively. The class is comprised of just under 5,000 lines of source code and about $1,500$ comment lines.

The use of classes for both the FSI interface (`fsiInterface`) and the structure (`Structure`) enables updates to both without changes to the actual FSI solver as long as the member function prototypes do not change. However, because the programming requires both the Standard Template Library (STL) [122] containers (`Structure` uses the STL) and OpenFOAM's custom containers, there is some redundancy in the data structures used to interface the OpenFOAM library with `Structure` which can be confusing and requires some additional memory. This situation could be avoided with some data type redefinitions, but it is not necessary for the present effort. The following sections provide an overview of the class, including a summary of the class member functions, and details on the fluid/solid interface communication and parallelization approach.

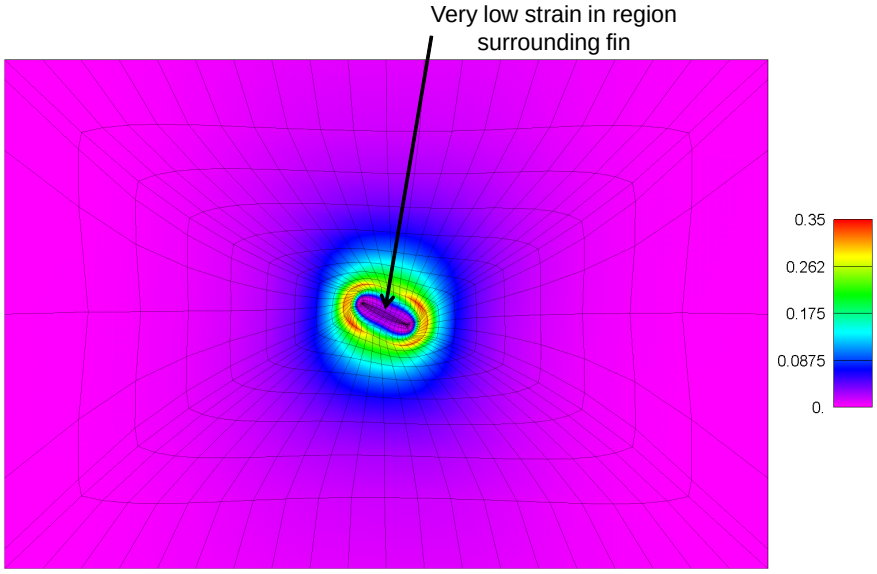

Fig. 2.9. Strain contours corresponding to the auxiliary mesh of Figure 2.8, showing very small strains and hence small mesh distortion near the fin for a rigid-body rotation of the fin

2.5.1 FSI Solver Overview

As described above, `fsiInterface` provides the interface between OpenFOAM and the structural solver but also handles the mesh motion for the ALE flow domain. Additionally, the class must support sub-iterations for fully-coupled simulations. Several of the necessary features to support these requirements, such as the sub-iterations, have been implemented in the structural solver class `Structure` and require only an interface to the correct member functions of `Structure`. Such cases are identified in the member functions listing of `fsiInterface` provided next.

`initialize` creates the fluid/structure interface mappings (further described in Section 2.5.3), creates interface-related output files, loads the finite element model of the structure, creates a generic load case that is modified during the simulation, and for parallel solutions assembles the interface mapping containers on the master node (further described in Section 2.5.4). This

step occurs automatically if the user instantiates the `fsiInterface` object by including the mesh object, the pressure field variable, and OpenFOAM's time object.

`saveStructureState` saves the current state of the structure and interface objects for use in sub-iterations for a tightly coupled solution.

`resetStructureState` resets the current state of the structure and interface objects to a previously saved state; required for sub-iterations.

`transferBoundaryStressToStructure` transfers the fluid stresses at $\Gamma_{F/S}$ to the structure (further described in Section 2.5.3).

`solveStructure` solves the structure for the current boundary conditions from the structure's current time to the prescribed end time (in an incremental fashion).

`moveFluidMesh` moves the fluid mesh (further described in Section 2.4).

`applyFluidAOA` rotates the structure about the z-axis to impart an angle-of-attack (AOA) change on the fin structure for the water tunnel/fin validation simulations described later. This reads a time-dependent listing of AOA's from a text file and uses the mesh motion solver to impart a rigid-body rotation of the structure on the fluid mesh. Note that the AOA time-dependency was implemented for general use because only fixed AOA's are required for the validation simulations.

`writeVTK` writes the structure results to a VTK file for post-processing. The file is only written if the current simulation time is one of the requested output times by the user in OpenFOAM's `system/controlDict` file.

A flowchart showing the flow of the `simpleFsiFoam` solver is provided in Figure 2.10. This high-level flowchart shows how the FSI solver begins by initializing

each of the three solvers (fluid, mesh motion, and structure) and their interfacing variables. After initialization the FSI solver begins a process of incrementing over all solution times, with each increment involving a fixed-point iteration to ensure the interaction of the fluid and solid domains is converged to within a specified tolerance (further described in Section 2.5.2). Prior to entering the fixed-point iteration, the current state of the structure is saved. This is necessary to correctly model time-dependent constitutive relationships, such as the viscoelastic material of concern here. Within each fixed-point iteration, the structure state is reset to this saved state, allowing the material to relax under the applied fluid stresses. Once converged, the fixed-point iteration is exited, the results are written to disk (if requested at this solution time), and the solution time is incremented. Iteration for the next solution time then begins. Details of the solver coupling by way of fixed-point iterations are provided next.

2.5.2 FSI Solver Coupling

As described earlier, the partitioned approach is being implemented for this work. This approach employs a *staggered*[3] solution procedure wherein each domain is solved sequentially and results are transferred between solvers. The most basic procedure, referred to as the Conventional Serial Staggered (CSS) approach by Piperno et al. [108], involves first a prediction of the structure's motion (${}^{p}\boldsymbol{u}^{n+1}$), solution of the fluid to get the stress acting on the structure (${\sigma_s}^{n+1}$), and then solution of the structure ($\boldsymbol{u}^{n+1}$), see Figure 2.11. It is evident from this figure that the CSS approach does not guarantee convergence of the fluid–structure interaction during a solution step because there is no check that the predicted structural displacements match the displacements computed at the end of the step. This

[3] Also known as a *segregated* or *time-lagged* approach.

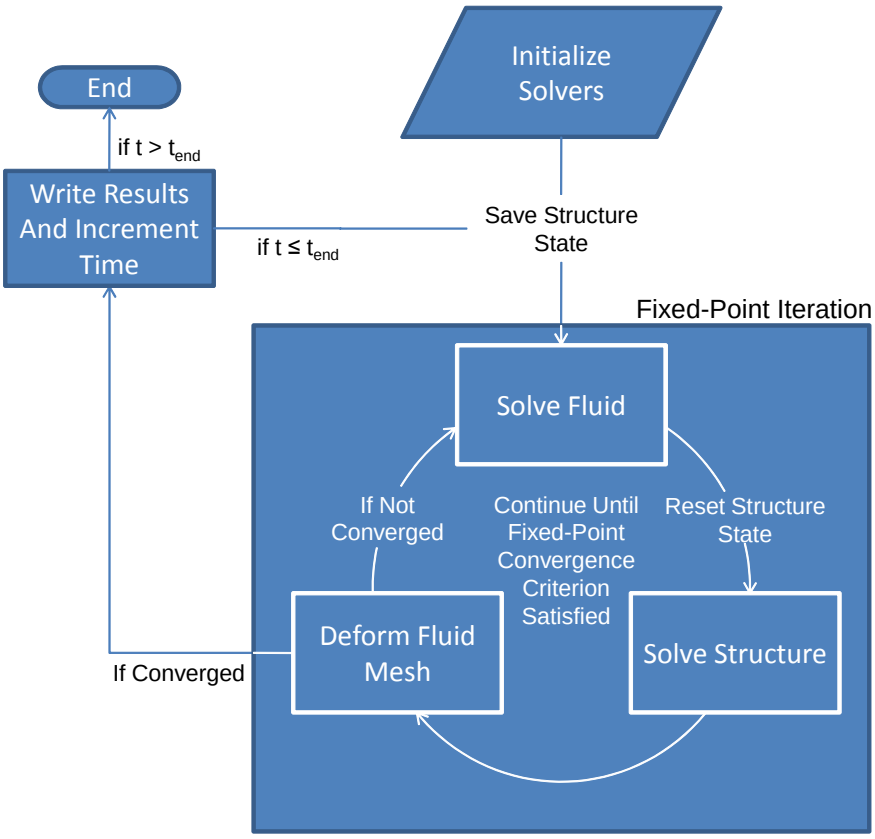

Fig. 2.10. FSI solver flow chart

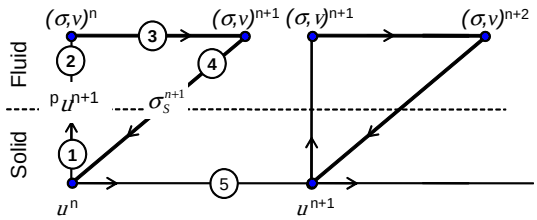

Fig. 2.11. Conventional Serial Staggered solution approach

algorithm is therefore said to provide a *loose* (or *weak*)[4] coupling [75]. The "tightness" (or "strength") of the coupling refers to the degree of convergence of the fluid and solid domains at any time during the solution. A loose coupling is akin to an explicit time integration even though the integration of the individual components may be implicit [95]. Similarly, a *tight* (or *strong*) coupling is akin to an implicit time integration. It is for this reason that "loose" and "explicit" are used synonymously and "tight" and "implicit" are used synonymously both in this work and in the literature [92, 93, 94].

A loosely coupled algorithm is converted to tightly coupled with the introduction of a corrector step that requires iteration to convergence at each solution step [93]. This modification ensures the computed structural displacements match the predicted displacements in the CSS algorithm at each step of the solution. The required iteration for a tightly coupled approach is generally either a fixed-point iteration [31, 93] or a block-Newton root finding formulation [34]. The most popular approach seems to be the fixed-point iteration because of the compatibility with black-box solvers (it does not require the calculation of a tangent, which is problematic for closed solvers). However, the fixed-point iterations tend to converge slowly unless certain improvements (e.g., the Aitken extrapolation as described later, Equation 2.49) are employed [56, 33, 141]. The solution procedure for a tightly coupled approach using a fixed-point is shown in Figure 2.10, and is repeated here in Figure 2.12 in a slightly different form having more details within the fixed-point iteration loop (see [2, 89] for additional information).

The details of how and when information is transfered across the fluid/structure interface for a loosely coupled segregated approach has received much attention over the last decade because of solution instabilities inherent to the staggered solution approach. The issue is related to energy conservation at the interface and

[4]The term "weak coupling" is also heavily used instead of "loose coupling". The terms "loose" and "tight" are used herein because they are antonyms.

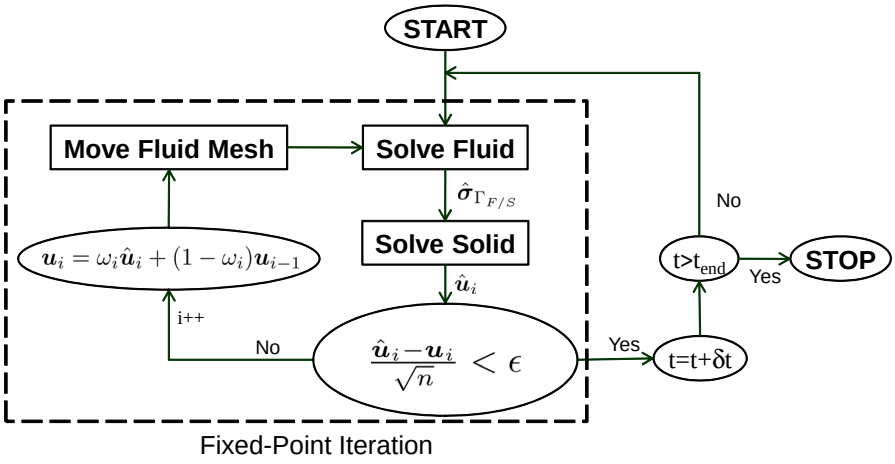

Fig. 2.12. Partitioned approach to FSI showing a fixed-point iteration with under-relaxation for tightly coupled solutions

is referred to as the "added-mass effect" [19, 47, 48]. The effect is most pronounced when the fluid and solid have similar densities [19, 101]. Several researchers have, with limited success, devised improvements to the loosely coupled schemes, such as the use of non-collocated time stations, to address the added-mass effect [108]. Fortunately, a tightly coupled approach does not suffer from the added-mass effects, as shown by Abouri et al. [2], Deparis et al. [32], and others.

Regardless of the tightness of the coupling, fluid stress information must be transferred from the flow solver to the structural solver and displacement information transferred from the structural solver to the flow solver.[5] In terms of equations, the requirements for compatibility and the no-slip condition require the following

$$\begin{aligned} \boldsymbol{v}^m &= \tfrac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} \\ \boldsymbol{\sigma}^S \cdot \boldsymbol{n} &= \boldsymbol{\sigma}^F \cdot \boldsymbol{n} \end{aligned} \qquad \text{on} \quad \Gamma_{F/S}, \tag{2.42}$$

[5] As described by Bathe and Zhang [9], for stiff structures it is important for stability to impose the fluid stresses on the structure and the structure's displacements on the fluid. Reversing this (i.e., imposing a stress boundary condition on the fluid and a velocity on the solid) might lead to instability because small errors in displacements imposed on the structure would lead to large errors in tractions imposed on the fluid because of the sensitivity of structural stresses to displacements.

where $\boldsymbol{n}$ is the unit normal on the interface and the superscripts on $\boldsymbol{\sigma}$ denote stress for either the fluid or solid domain.

Note that the FSI implementation developed here is for the *permanent type*, which implies there is no fluid detachment (i.e., the fluid stays in contact with the structure) [17, 18]. This restriction helps simplify the problem allowing the software to be more general because the interface information need not be updated during the simulation. The fluid points will then always be in the same relative position with respect to the structural nodes and vice versa.

The flowchart defining the solution procedure in Figures 2.10 and 2.12 include a fixed-point iteration that is performed to ensure the fluid pressures and solid displacements are tightly converged before moving on to the next time step. The use of under-relaxation in this iteration (ω in Figure 2.12) controls the amount of the structure's displacement that is imparted on the flow domain and is solely used to improve convergence. The relaxation coefficient can be constant, or, for improved convergence characteristics a dynamic coefficient is often used. One option for dynamically changing the coefficient is the Aitken Δ^2 method [76]. This method employs a recursion formula that originates with the secant root finding method:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n), \tag{2.43}$$

where x_n in this case is the *applied* interface structural displacement, represented by the variable d, at the current time step d_i. (The derivation is provided for one-dimension and then is later expanded to three dimensions.) Here, the function being ciphered is the difference between the *computed* displacement $\hat{d}$ to be used in the next iteration of the loop $i+1$ and the *applied* displacement d for the current iteration i with the goal of finding the *applied* displacement for the next iteration d_{i+1}:

$$f\left(d_i\right) = \hat{d}_{i+1} - d_i. \tag{2.44}$$

Rewriting Equation 2.43 in terms of d_{i+1} and substituting for the function with Equation 2.44 gives the applied displacement for the next iteration

$$d_{i+1} = \frac{d_{i-1}\hat{d}_{i+1} - \hat{d}_i d_i}{\left(\hat{d}_{i+1} - d_i\right) - \left(\hat{d}_i - d_{i-1}\right)}. \tag{2.45}$$

A relaxation factor ω_i is sought such that

$$d_{i+1} = d_i + \omega_i \left(\hat{d}_{i+1} - d_i\right). \tag{2.46}$$

Combining Equations 2.45 and 2.46 and factoring the numerator yields

$$\omega_i = -\omega_{i-1} \frac{\hat{d}_i - u_{i-1}}{\left(\hat{d}_{i+1} - d_i\right) - \left(\hat{d}_i - d_{i-1}\right)}. \tag{2.47}$$

or

$$\omega_i = -\omega_{i-1} \frac{r_{i-1}}{r_i - r_{i-1}}. \tag{2.48}$$

where $r_i = \hat{d}_{i+1} - d_i$. This is the same formula presented by Küttler and Wall [76].

For the case of vectors instead of scalars, the quotient of Equation 2.48 cannot be evaluated and thus the following approximation is used [76]

$$\omega_i = -\omega_{i-1} \frac{\boldsymbol{r}_i^{\,T}(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)}{|\boldsymbol{r}_{i+1} - \boldsymbol{r}_i|^2}, \tag{2.49}$$

As described by Küttler and Wall [76], this is a projection of the participating vectors in the $\boldsymbol{r}_i - \boldsymbol{r}_{i-1}$ direction. The relaxation equation, Equation 2.46, is shown in the coupling algorithm of Figure 2.12. Convergence of the system is

based on the residual norm as follows:

$$\frac{|\boldsymbol{r}_i|}{\sqrt{n}} < \epsilon, \tag{2.50}$$

where n is the length of $\boldsymbol{r}_i$ and ϵ is the error tolerance that can be set based on an allowable error in the displacements, which is interpreted as a small fraction of the size of the structure's domain. The FSI solver allows the user to set this error tolerance via OpenFOAM's `system/controlDict` file.

It should be noted that the relaxation parameter for the current iteration depends upon information from two previous iterations. Therefore, FSI simulations using dynamic under-relaxation as defined here require at least two fixed-point iterations and require a prescribed relaxation parameter for the first two iterations. The choice of this parameter for the first two iterations is found to be problem-dependent and in general should not be set from previous time increments [76].

The intent of the present research is to first employ a constant relaxation coefficient and proceed to a dynamic coefficient only if required. The details of the information communication between the solvers appearing in Figure 2.12 is described next.

2.5.3 FSI Interface Communication

The communication of stress and displacement information between the flow and structural solvers is a subject that has recently attracted several researchers (see for example [40, 49, 124]) and has warranted at least one focused PhD dissertation [123]. The difficulty in the coupling involves the use of non-conforming meshes at the fluid/solid interface $\Gamma_{F/S}$ and how to transfer fluid stresses to the structure and structural displacements to the fluid in an energy-conserving manner. The simplest coupling scheme is to have matching interfaces and exchange fluid stress and displacements in a one-to-one fashion. However, it is generally advantageous

to employ different mesh resolutions for the fluid and solid domains because the fluid domain often requires much more refinement to accurately model the flow field and using an over-refined structural mesh can lead to excessive computation times. Moreover, the use of non-conforming meshes at the interface simplifies mesh generation, allowing the fluid and solid domains to be independently discretized.

The techniques applied to couple the fluid and solid at the interface for non-conforming meshes range from simply interpolating the fluid stresses onto the structure's element faces and integrating by the structural solver to get force [9], to a fully implicit approach using Lagrange multipliers wherein the interface traction and displacements are solved simultaneously [123, 124] (but currently only demonstrated for one- or two-dimensional problems using finite element formulations for both the fluid and solid domains). In any case, the objective of the coupling scheme is to satisfy the requirements of Equations 2.42.

The present application employs a *consistent* approach similar to that described by Farhat et al. [40] and used by others [49, 118] (including the MpCCI commercial coupling software [49, 99]). The consistent approach, as introduced by Farhat et al., states that the solver that owns the information to be exchanged interpolates that information to the necessary location required by the receiving solver, thereby using an interpolation scheme that is consistent with the scheme used to compute the information. For the case of pressure or viscous stress transfer from the fluid to the structural solver, the approach of Farhat et al. calls for the stress to be interpolated to the Gauss points of an appropriate structural face and the structural solver then integrates the stress to compute a force. This approach, however, does not guarantee the same discrete loads as computed by the flow and structure solvers because in general they will employ different interpolation schemes and have different discretizations, resulting in a non-conservative coupling (force times displacement on the fluid side does not equal force times displacement on the structural side). Note that Farhat et al. [40] show that even

though the coupling is strictly non-conservative, numerically conservative results can be achieved even for highly disparate meshes.

The approach employed here uses a consistent approach in that the solver that owns the variable interpolates that variable, but with the modification to the method in [40] that the same solver also integrates the field if an integration is required. This means that the flow solver integrates over all cell faces on $\Gamma_{F/S}$ to compute a resultant fluid force centered on the cell face. This force is then applied to the structural solver by weighting the force magnitude based on distance to the structural nodes to approximately preserve the moment. This approach, which is similar to the approach used by the MpCCI software as described by Glück et al. [49], guarantees the resultant force on the fluid and structural solvers are identical. However, as stated in [49], the disadvantage is that for a coarse source grid and a fine receiving grid the forces are distributed in a non-physical way. Fortunately, the fluid domain usually requires a much finer mesh than the structural mesh thereby alleviating this issue.

Transfer of displacement information from the structure to the fluid does not require integration, and thus only interpolation is required by the structural solver to approximate the computed nodal displacements at prescribed locations on the fluid/structure interface $\Gamma_{F/S}$.

The process of coupling the solvers begins by identifying for each fluid face center[6] on $\Gamma_{F/S}$ an owner structural element. The structural element must have a face with an outward facing normal that is nearly opposed to the outward facing normal assigned to the fluid face. The face normals, in general, will not be of exact opposite sense because of the disparate discretization of non-planar surfaces, as demonstrated by the two-dimensional example in Figure 2.13. The location of the fluid vertex must also fall within the bounds of the structural face. This is achieved for each fluid vertex by searching all structural faces until the best match

[6] OpenFOAM is a cell-centered solver, so the integrated fluid forces reside at the face centers

is found, with the match criteria weighted heavily on proximity and a go/no-go criterion on face normal directions to be greater than 135° of each other.

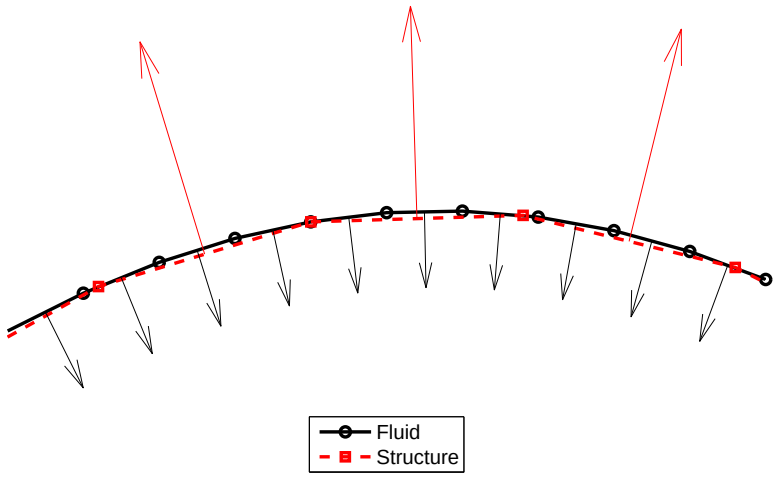

Fig. 2.13. Example of disparate fluid and structure meshes at the interface $\Gamma_{F/S}$; the fluid vertices do not lie on the structural element faces and the surface normal directions between the fluid and structure are not exactly opposite

Because the structural solver requires forces be applied to the structural nodes, the fluid force vector computed at the face centers is applied to the structural nodes by a load-weighting factor that depends on the distance from the fluid face location projected onto the structure's face to each node of that structural element's face. This scheme results in a unity scale factor if the fluid location coincides with a structural node and yields a scale factor that equally divides the force if the fluid location is centered on the structural face. An example showing the load scaling factors for a fluid face location that is not quite centered on a structural face is provided in Figure 2.14. During each loop of the fixed-point iteration, the fluid force vector is computed by the flow solver using its inherent

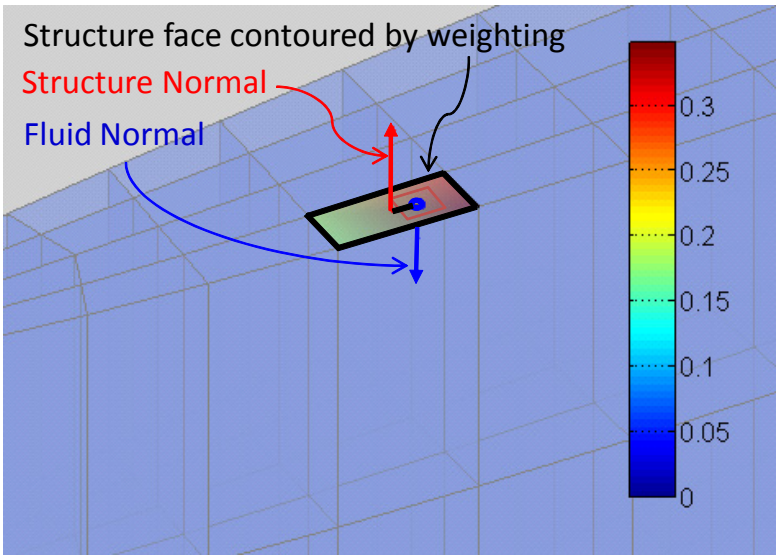

Fig. 2.14. Example of interface force mapping showing the load scaling factors used to apply the fluid load to the structural nodes

interpolation scheme and applied to the structural nodes according to the scaling defined at the start of the analysis. (The fluid and solid move together during the analysis such that the parameterized location of any given fluid face on a structural element remains constant for the duration of the simulation.)

All structural nodes that are subject to fluid loading are identified in the structural model with a distributed load boundary condition, i.e., using the `*dload` keyword to be consistent with the Abaqus input file format [57]. This load is subsequently removed from the structure's load set.

Communication requirements of the structural displacements to the fluid mesh varies depending on the mesh motion scheme as defined in Section 2.4. Regardless of the scheme, the computed structural displacement field must be interpolated to a prescribed location on the interface $\Gamma_{F/S}$ (e.g., a fluid vertex for the Laplace face decomposition and RMF mesh motion, or an enforced displacement location for an overlayed finite element mesh). The prescribed spatial location on $\Gamma_{F/S}$ changes during the simulation, but its location relative to the finite element

reference coordinates does not change if the point resides on $\Gamma_{F/S}$ (because the fluid and solid move in unity on $\Gamma_{F/S}$). The process involves first locating the owner structural element for each mesh motion vertex, and then finding the location on the owner element in terms of the element's parameterized coordinates using the `feanl` member function `elementParameterizedLocation`. This member function uses a three dimensional Newton-Raphson iteration scheme with a numerical Jacobian to iterate within the reference element to find the prescribed location to within a specified tolerance of $1.0\text{x}10^{-10}$ (the iteration stops when the Euclidean norm of the point location difference is less than this tolerance). In practice, the point locations are generally found to within machine tolerance. An example reference element showing a point inside the element for which the parameterized location is sought is shown in Figure 2.15.

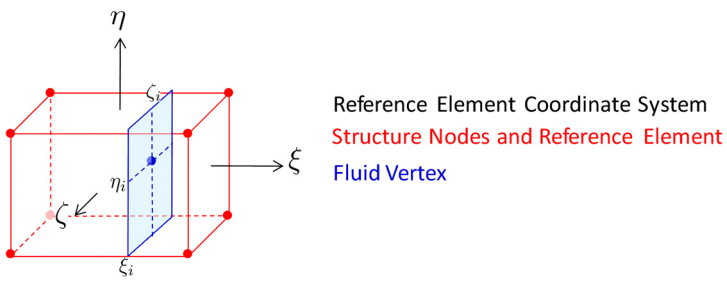

Fig. 2.15. Example of a finite element reference element, its reference coordinates, and a point within the element for which the parameterized location (ξ_i, η_i, ζ_i) is sought

2.5.4 FSI Parallel Processing

Parallel processing is important for this research to provide reasonably short simulation (wall-clock) times for the numerous required simulations. The approach

taken here is to maintain OpenFOAM's existing parallelized flow solver[7] and implement parallel support for the FSI interface class `fsiInterface`, which involves parallel mesh motion as described previously for the RMF solver and communication with the structure solver. The structural solver operates in serial model for all FSI simulations and is always solved on the master node. Interface displacements computed and interpolated by the structural solver are then sent to each computed node based on interface mapping containers created at the start of the simulation. Likewise, fluid forces computed by each processor are sent to the master node for inclusion in the structural solver's solution. Solution times for this scheme of parallel processing are still dominated by the flow solver as shown later in Section 5.6.1.

[7] OpenFOAM uses a domain decomposition approach for parallel solves. This approach sub-divides the domain and assigns each division to a single processor.

Chapter 3

Structural Material Model

The target application for this research is an expandable impeller pump that is fabricated from a polymeric material to satisfy large strain requirements of the device. The downside to using a polymeric material for the pump is that the material exhibits stress relaxation, which is time-dependent deformation that occurs even for a constant load. Aside from the material's viscoelasticity, the need to collapse the impeller during implantation [96] requires the material to be relatively soft. As a result, large deformations of the impeller blades occur in response to the fluid stresses imparted on the impeller blades. An accurate model of the impeller material is paramount to an accurate FSI simulation of this device. As such, an entire chapter is devoted to the characterization of the material.

The impeller material is an industrial-grade polymer called Hapflex 598, purchased from Hapco, Incorporated of Hanover, Massachusetts, USA. The Hapflex 598 material exhibits a stress-strain relationship depicted by the tensile test results shown in Figure 3.1. These results are generated by the cyclic loading of a tensile specimen from no load to a strain of approximately 5% for several cycles and then followed by cyclic loading to approximately 10% for several cycles. The load rate for all cases is 1% strain per second. It is evident from this figure that the material does not instantaneously return to zero strain upon load removal. The non-zero strain is comprised of nonrecoverable and recoverable strains. The recoverable strain comes from the material's viscoelasticity. The fraction of recoverable to nonrecoverable strain could be determined, for instance, by subjecting the tensile test specimen of Figure 3.1 to one cycle of loading and then monitor the specimen's strain at zero load until a steady state condition is achieved. Experience

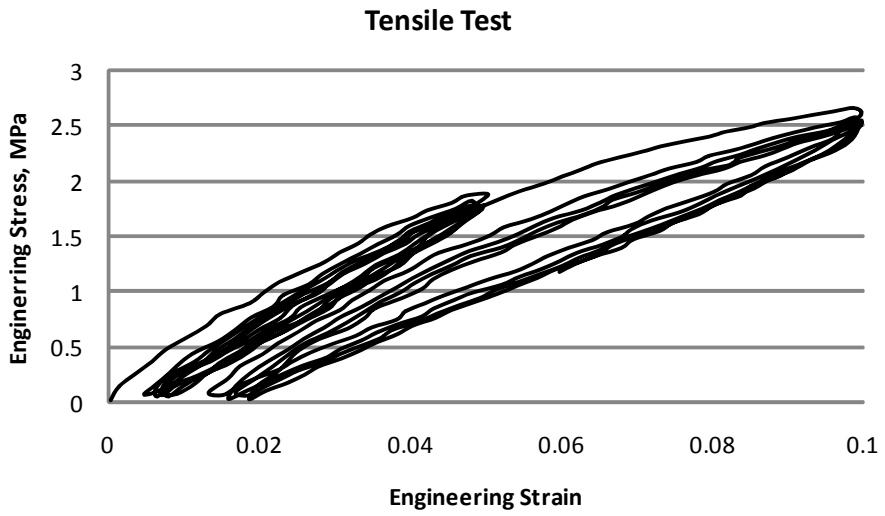

Fig. 3.1. Hapflex 598 tensile test results with a load rate of 1% strain per second

with this material (for strain levels of $\lesssim 5\%$) has shown that nearly all of the strain is recovered after several hours in a relaxed state.

The current application requires only the loading portion of the first cycle of the curve in Figure 3.1 and does not require any unloading or cyclic behavior. Therefore, the material model is comprised of two components. The first is a constitutive representation of the material that is independent from effects of stress relaxation. This is accomplished by loading the material either over a very short duration where there is negligible time for the material to relax or over a very long duration to provide sufficient time for the material to fully relax. A second component is necessary to model the stress relaxation. This is accomplished by treating the material as viscoelastic. The constitutive models employed for this material are described in Section 2.3.5. The material parameters required for these constitutive relationships are defined in this section.

It is shown in this section that the Hapflex material exhibits different deformation response based on lot-to-lot variations. Material lot-to-lot variation is known to cause significant property variation as described in [57]. The approach

employed here is to evaluate an initial material model, derived from material characterization tests performed on samples of the material in 2007, by comparing FE simulations to experimental results. The material model is then tuned such that the FE model simulations match closely the empirical results. The initial material model is derived from material testing performed by Axel Products, Incorporated (API).[1] It has been found that modifications to only the viscoelastic properties are necessary to get good agreement between the simulations and experiments. Also, the use of a linear elastic material model is found to be sufficient and the elastic modulus and Poisson's ratio are determined from the test results of API. The final viscoelastic property parameters are determined through a trial-and-error process and confirmed by beam and fin material testing as described below.

3.1 Elastomer Elastic Modulus

The underlying model used to represent the elastomeric material (in one dimension) is the Generalized Maxwell Element, which consists of Maxwell elements (i.e., a spring and dashpot in series) in parallel with a Hooke element (i.e., a spring) as shown in Figure 2.4. The spring stiffness μ_0 shown in this figure represents the material stiffness at infinite time (i.e., after all of the viscoelastic forces have diminished to zero). The Maxwell elements capture the stress relaxation characteristics of the material through their dashpots. Each of the Maxwell elements constitute a term in a Prony series (a series of the form $\sum_{i=1}^{N} \alpha_i e^{-t/\tau_i}$) representation of the material, which is further described below.

The spring with stiffness μ_0 of Figure 2.4 (which is observed at infinite time after all of the dashpots have relaxed) corresponds to the *long term modulus* for a three-dimensional, linear-elastic material model. The *instantaneous modulus* is obtained through the step application of a load to avoid any viscoelastic effects

[1] Testing performed in October 2007 by Axel Products, Incorporated of Ann Arbor, Michigan `http://www.axelproducts.com/`.

because the dashpots have no time to react. The long term and instantaneous moduli are related through the viscoelastic Prony parameters, γ_i, as follows [57]:

$$E_{\text{inst}} = E_\infty + \sum E_i \Rightarrow E_\infty = E_{\text{inst}} - \sum E_i = E_{\text{inst}} \left(1 - \sum \gamma_i\right), \tag{3.1}$$

where E_∞ is the long term modulus ($E_\infty = \mu_0$ of Figure 2.4), and E_{inst} is the instantaneous elastic modulus. Ideally, the instantaneous modulus is obtained empirically by a step application of a load, but limitations of test equipment preclude a truly instantaneous load application. Instead, the approach recommended for determining the instantaneous modulus is to measure the initial elastic response under several increasingly high strain rates, until the measured modulus converges on a value [57].

Tensile test results for Hapflex 598 specimens are provided in Figures 3.2 and 3.3, which were acquired from simple tension tests employing "dog-bone" specimens (shown in the inset of Figure 3.3). The objective of the simple tension test is to achieve a state of pure tensile strain, and thus the specimen must be much longer in the direction of stretching than in the width and thickness dimensions. Ideally, there will be no lateral constraint to specimen thinning.

The data of Figure 3.2 do not indicate a substantial change in tensile performance of the material for the evaluated strain rates, and thus this test approximates the idealized stepped load test described above. Therefore, API employed a strain rate of 1 mm/mm/s for the tensile tests of Figure 3.3 and for all of their subsequent testing. The Young's modulus for the material is approximately 60. MPa as estimated from the simple tension data and is shown by the solid curve in Figure 3.3, which considers strains up to 1.2%. Later it will be shown that the computed material strain levels for the FSI simulations are less than this maximum value.

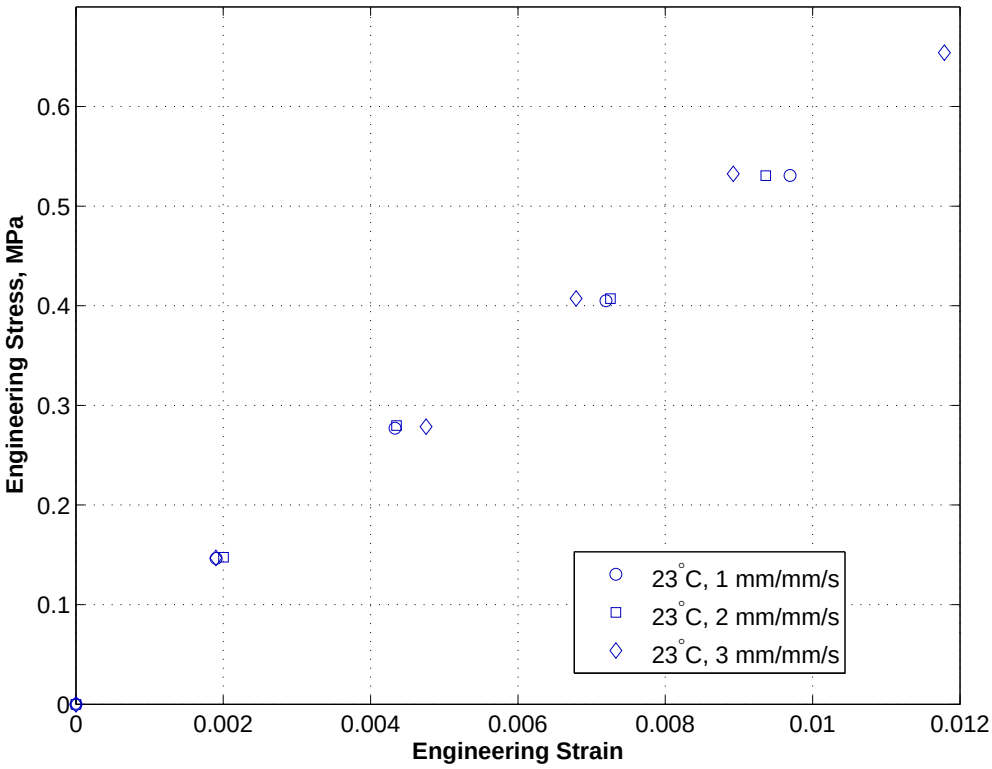

Fig. 3.2. Simple tension results for Hapflex 598 at 23 °C and three strain rates

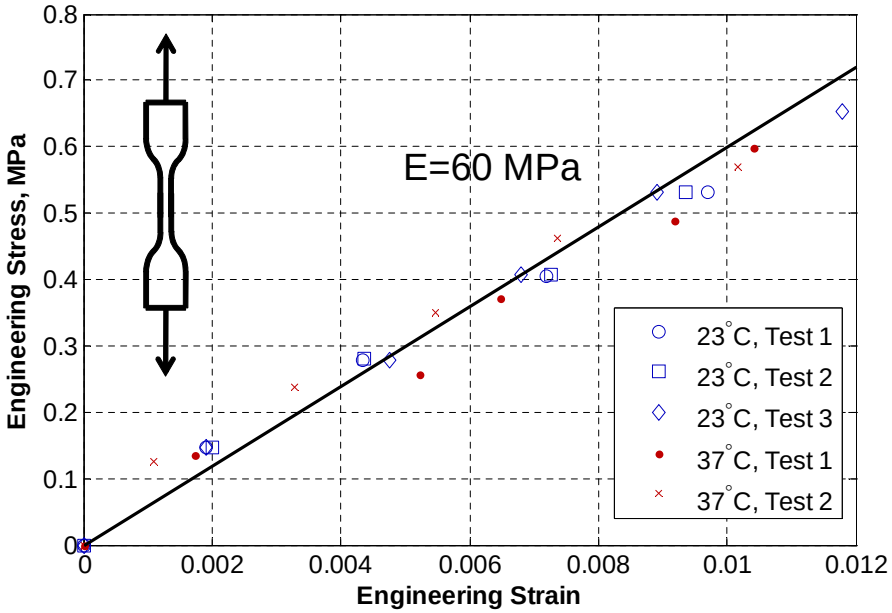

Fig. 3.3. Simple tension results for Hapflex 598 at two temperatures and curve showing stress-strain values for a Young's modulus of 60 MPa

3.2 Viscoelastic Material Definition

As previously described in Section 2.3.5, the material viscoelasticity is modeled using a time-domain approach for linear viscoelastic materials ($\varepsilon[c\sigma(t)] = c\varepsilon[\sigma(t)]$, where c is a constant) and uses the approximations that shear and volumetric behavior are independent and the viscoelastic behavior is dominated by the deviatoric part of the material's deformation.

Viscoelasticity is often characterized by stress relaxation data. These data are generated by straining a uni-axial tensile stress specimen to a prescribed strain level and then measuring the stress as it relaxes with time while holding the strain constant. The stress relaxation data measured by API is provided in Figure 3.4. These data were collected for a strain of 5%.

The curve fit shown in Figure 3.4 is obtained by fitting a Prony series using a least-square error approach to the 23 °C Test 1 data. Three components of a Prony series are required to obtain a fit with a root-mean-square error of 0.49%, which are provided in Table 3.1. These parameters are then used by the finite element

Table 3.1. Prony series stress relaxation parameters

Component, i	γ_i	τ_i
1	0.1484	4.130
2	0.2115	$8.195\text{x}10^{1}$
3	0.1993	$1.610\text{x}10^{3}$

material model to represent the material's viscoelasticity. This viscoelastic model provides a starting point for the material model and is updated below based on beam simulation and test results.

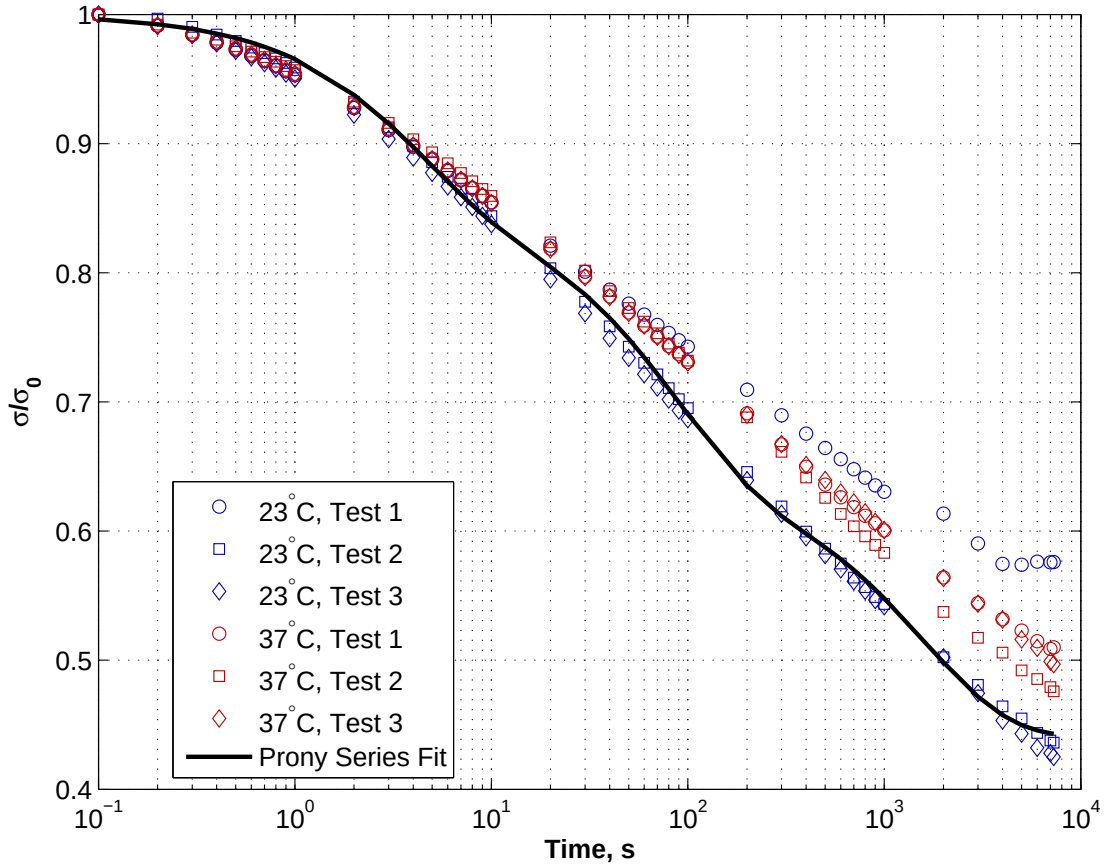

Fig. 3.4. Stress relaxation data with measured stress normalized by initial stress; Prony series fit using three Prony series parameters of Table 3.1

3.3 Poisson's Ratio and Nearly Incompressible Effects

The current material exhibits what is considered "nearly incompressible effects" because the bulk modulus is much larger than the shear modulus, yielding near zero volumetric strains when deformed. Because strains are determined via derivatives of displacements in the displacement-based finite element approach, errors in predicting the near zero volumetric strains yield large variations in the computed stresses, which in turn affect the computed displacements when trying to balance element forces with externally applied loads [8]. This behavior can be addressed by decomposing the stress into deviatoric and volumetric components and then solving for the volumetric component separately using a displacement/ pressure approach instead of the more common displacement-based finite element approach. The Abaqus software has implemented the displacement/pressure approach via their "hybrid" elements. The benefit of using a hybrid approach over a pure displacement approach with and without reduced integration elements is evaluated using Abaqus. But first, an estimate of Poisson's ratio is required.

Poisson's ratio for a linear elastic material can be determined from the Young's and the bulk moduli as follows:

$$\nu = \frac{3\kappa - E}{6\kappa}, \tag{3.2}$$

where κ is the bulk modulus. The bulk modulus is estimated from volumetric compression data acquired by API for the Hapflex 598 material. The volumetric compression results provided in Figure 3.5 show a bulk modulus of 2.3 GPa is a good approximation for the material.

The Young's modulus for the material is estimated as 60. MPa from the simple tension data. Substituting these values into Equation 3.2 yields a Poisson's ratio of $\nu = 0.496$.

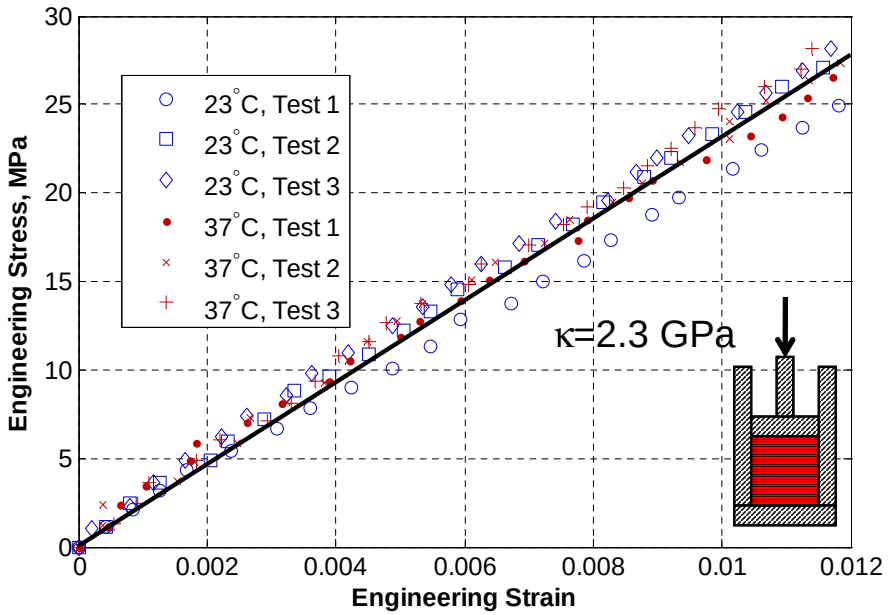

Fig. 3.5. Volumetric compression results for Hapflex 598 samples at two temperatures and a curve showing stress-strain values for a bulk modulus of 2.3 GPa

To evaluate the effects of this nearly incompressible material on the use of pure displacement-based finite elements, two finite element models are evaluated both with and without hybrid elements. The first FE model is of a Hapflex 598 beam undergoing a bending load (Figure 3.6) and the second is a fin structure also undergoing a similar bending load (Figure 3.7). Bending deformations are considered here because they are the most representative of the load types pertinent to this research (impeller blade bending) and this mode of deformation is much more susceptible to the shear locking effects of the nearly-incompressible material [22]. Both models employ fixed-free boundary conditions and details of the models are provided in the following sections of this chapter. The bending load for both cases is achieved by a concentrated force near the tip of the free end. The concentrated force in the FE simulations is linearly increased from zero load to its maximum value in one second (the step application of this load for the quasi-static analysis is acceptable, but is difficult to achieve with the large displacement, non-linear analysis because the deformations are large and the FE solvers are based on an incremental formulation).

Comparisons of tip deflections for the cases with full integration displacement elements, selectively reduced integration displacement elements, and hybrid elements for both models are provided in Figure 3.8 for the beam and Figure 3.9 for the fin. The results in these figures suggest there is no need to employ pressure/displacement finite elements for these models, because selectively reduced-order integration elements provide equivalent results as expected from Hughes [62] and Bathe [8]. Note that these models employ the final material model described in Section 3.5.1.

3.4 Temperature Effects

Two temperatures were used in the material evaluations performed by API to ascertain the impact of temperature on the material stiffness and relaxation. The

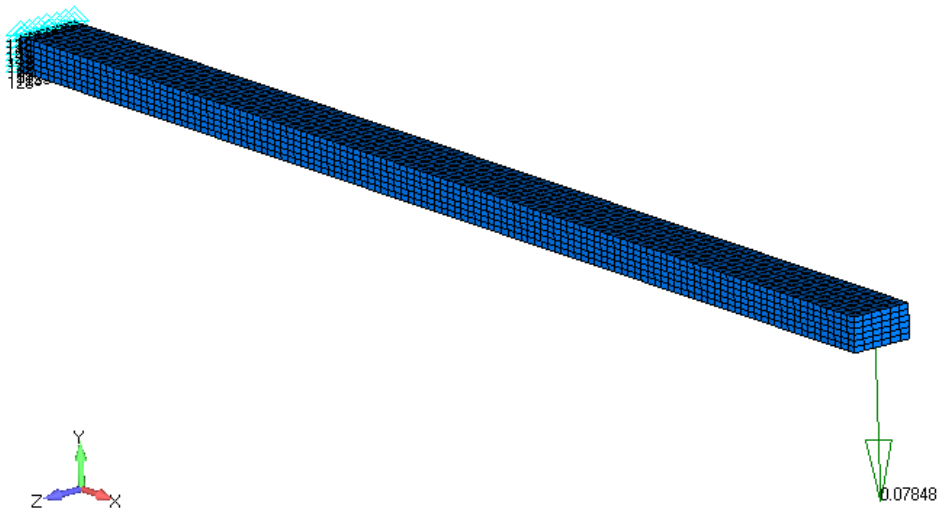

Fig. 3.6. Finite element model of Hapflex 598 beam test; two models used, coarser model shown here

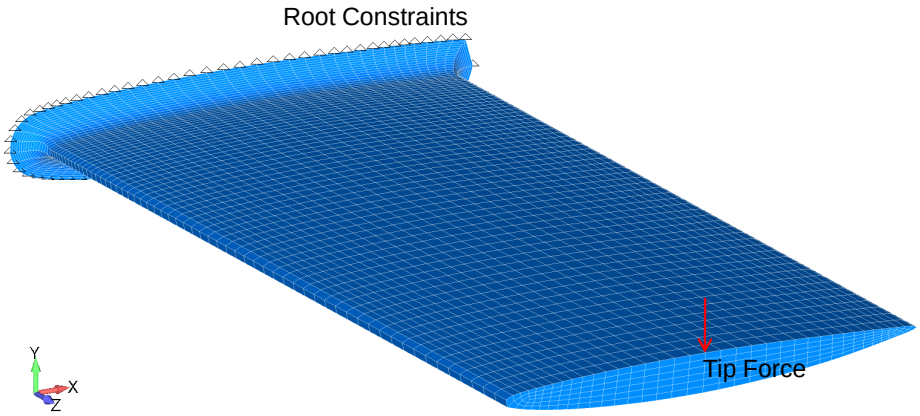

Fig. 3.7. Fin bending model for evaluation of incompressible effects

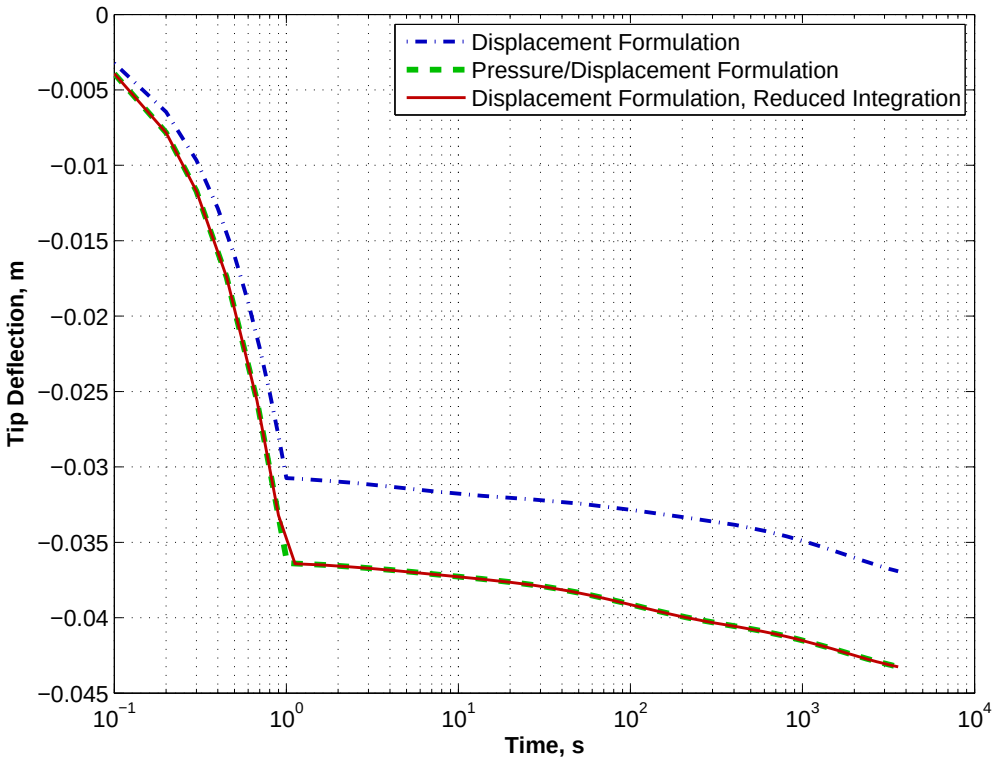

Fig. 3.8. Beam bending model results for displacement-based and pressure/displacement elements; the displacement-based elements with reduced integration agree well with the pressure/displacement hybrid elements

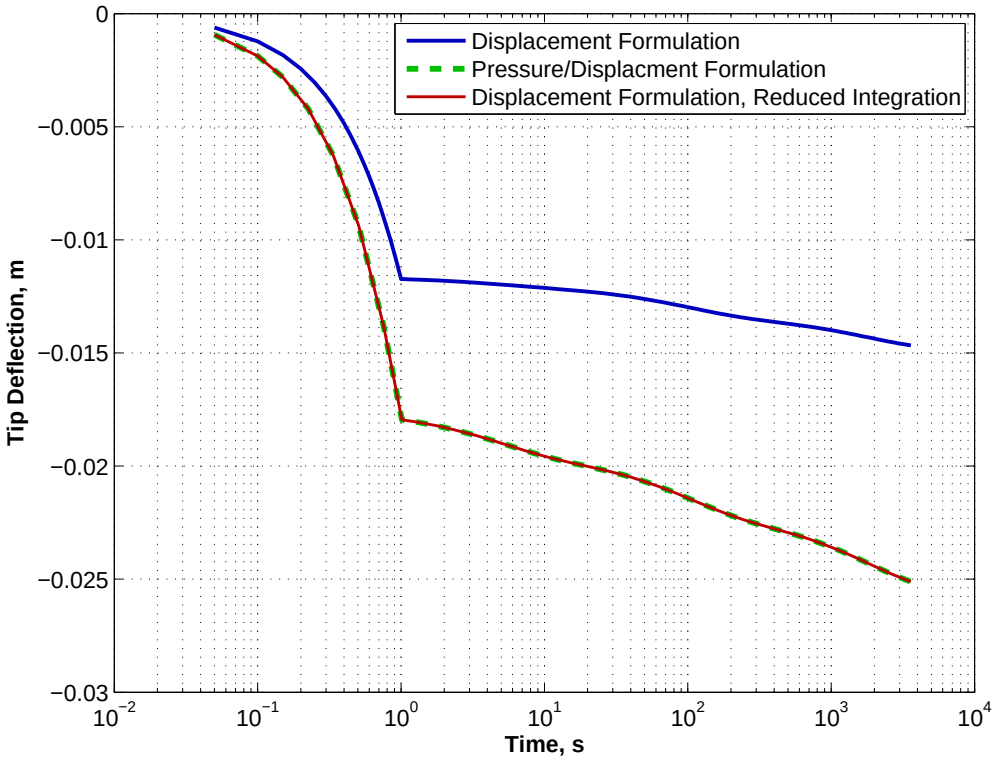

Fig. 3.9. Fin bending model results for both displacement-based and pressure/displacement elements

temperatures were chosen as 23 °C and 37 °C to correspond to room temperature and body temperature, respectively, thereby bounding the anticipated temperature variation for use in the expandable impeller pump.

The data shown above in Figures 3.3–3.5 do not indicate a significant impact of this temperature variation on the material characteristics of interest. The material model developed here is used to model a fin subjected to quasi-steady flow with a nearly constant fluid temperature of 22 °C and also to a pump impeller subject to fluid at 37 °C. Therefore, temperature effects are not included in the present material model.

3.5 Parameter Estimation

3.5.1 Beam Model

The material model is evaluated against large-displacement[2], time-dependent bending tests of sample beams made of Hapflex 598. The beams are intended to have rectangular cross sections of 3.5 mm thick, 6.5 mm wide, and the tested length (from edge of constraint to tip) is 125 mm (see Figure 3.10). Three beams have been tested. The beam width and length are consistent between the samples, but the beam thickness ranges from 3.41 mm to 3.62 mm along the length of the beams. This causes scatter in the empirical results presented below and is a source of discrepancy between the predicted and empirical results.

The test conditions for the beams consist of fixed-free constraints and a constant load applied to the beam tip, Figure 3.11. The beams are clamped by a mounting bracket such that the active length of the beams from the edge of the clamp to the tip is 125 mm. The beam deformation is quantified through pattern

[2] "Large-displacements" as described herein imply displacements large enough to violate the assumption of infinitesimal displacements as used in the derivation of the linear finite element approach. This means, for a linear-elastic material, the displacement response of an object is not a linear function of an applied load.

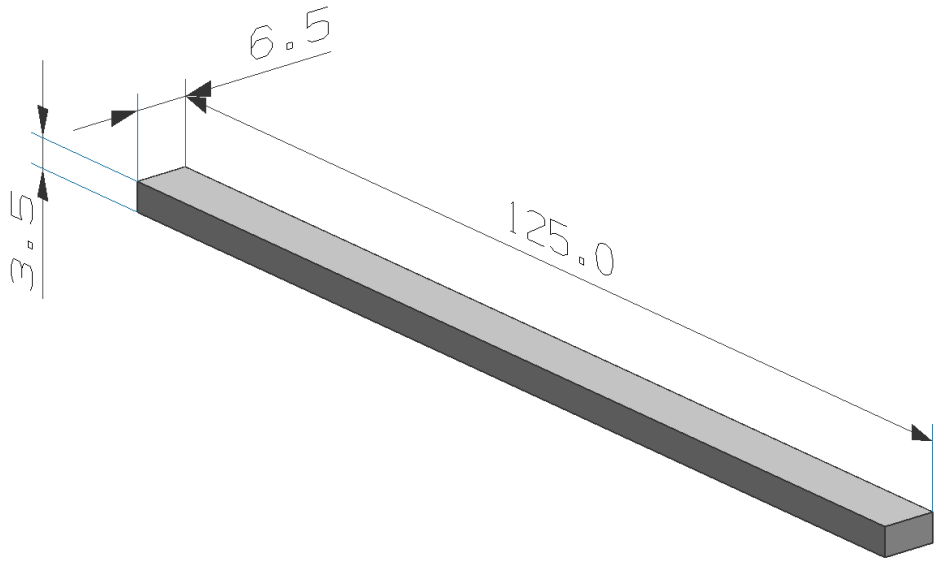

Fig. 3.10. Dimensions of Hapflex 598 beams tested for material model validation (dimensions in mm); Note that 125 mm is the active length of the beams during testing, the actual beam length is 150 mm

recognition of the beam edges using a video camera setup as shown in Figure 3.12.

The procedure for testing the beams is as follows:

1. clamp beam into fixture ensuring the active length is 125 mm,
2. start the data acquisition,
3. apply known weight to tip of beam, and
4. acquire images for one hour.

An edge detection routine with pixel scaling to scale the imaged displacements was implemented in Matlab to post-process the videos. A listing of the source code, called `imageEdges`, for this software is provided in Appendix B. Sample output from the program is shown in Figure 3.13. From results similar to these for all

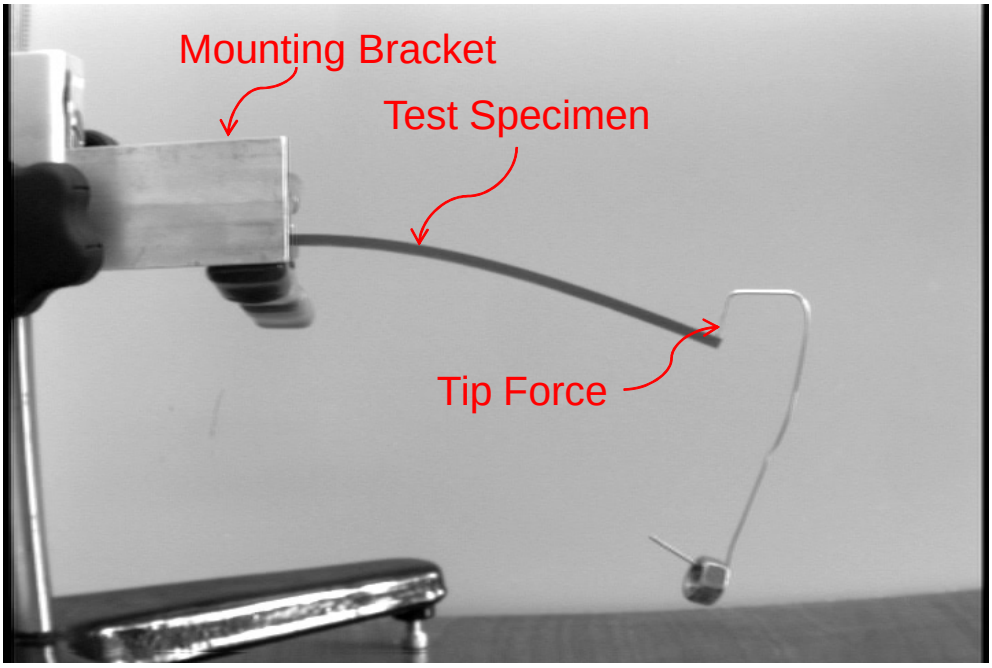

Fig. 3.11. Boundary conditions of the Hapflex 598 beam tests showing the fixed-free constraints and the tip load

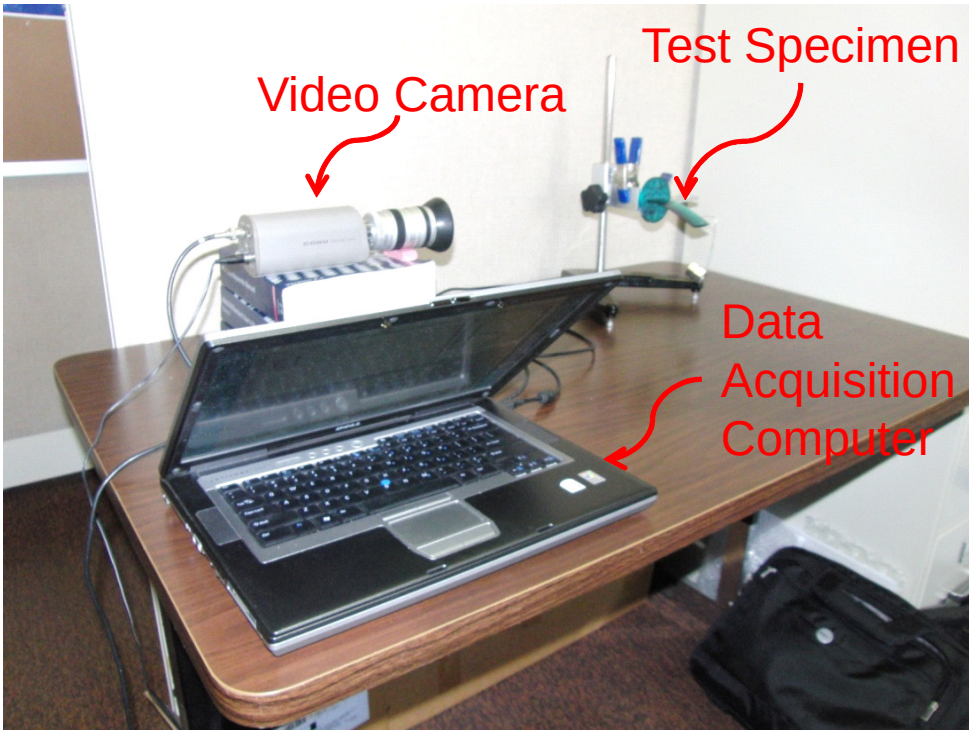

Fig. 3.12. Test setup for beam and fin bending tests; a video camera is used to capture images of the deformed specimen; the setup is shown here with a Hapflex 598 fin (described below), but the same setup is also employed for the Hapflex beams

Fig. 3.13. Sample image of edge detection for the Hapflex 598 beam test; the top and bottom edges are identified separately and shown here with different colors

acquired frames, the beam tip deflection is quantified and plotted versus time in Figure 3.14. Note that the resolution of the results in this figure is limited by the pixel size of the processed images. In all cases, the load application began at approximately 5 seconds and was completed by approximately 10 seconds. However, the effect of gravity on the beam (which is significant as shown below) began during setup of the beam. The effect of gravity on the beam deformation between when the beam is exposed to gravity and when the video recording began has been estimated by measurements from a stationary ruler near the beam tip (not shown in Figure 3.11 or 3.12). The magnitude of this deflection is evident in the non-zero deflection at time = 1 second in Figure 3.14.

The validity of the material model is then determined by comparing finite element model results of the beam model to the empirical results shown in Figure 3.14 and by comparison of deformation shapes. The beam deformation shapes are compared after about one hour of relaxation under the applied load.

The finite element model is constructed in the commercial software Femap [139] using hexahedral elements, constrained at the root, and loaded with a concentrated load of 0.0785 N (corresponding to the 66.5 g mass shown in Figure 3.11

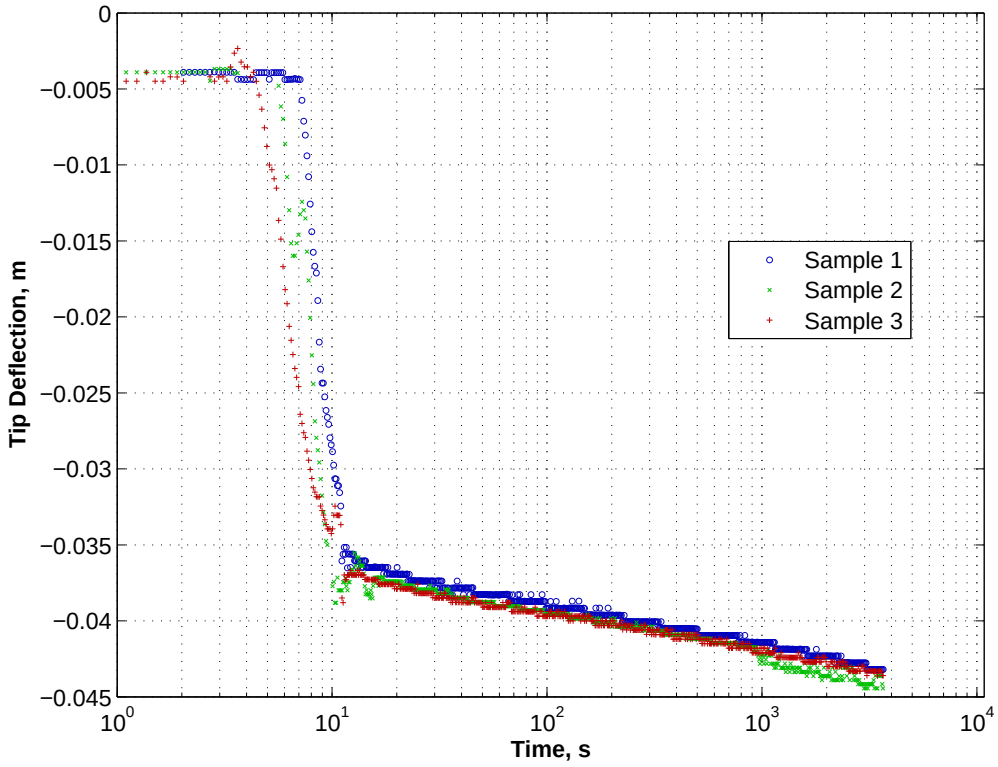

Fig. 3.14. Beam tip deflection for three Hapflex 598 beam samples subjected to a tip force of 0.0785 N (8 g mass)

at the free end) and a body force due to gravitational effects (the beam deformation due to gravity only is shown in Figure 3.15, which uses the final material model described below and corresponds to time of 1 hour). The concentrated load is linearly increased from zero magnitude to its maximum value in one second (the load application for the experiments is somewhat longer than this, but the numerical results are shifted in time such that the time at which the load is fully applied agrees between all results). A quasi-static analysis is used for the simulations because time only affects the viscoelastic behavior. The inertial terms are negligible in the experiment because the load was applied nearly uniformly over several seconds, which is much lower in frequency than the first flexural mode of the beam, approximately 3.5 Hz as estimated experimentally with the tip mass attached.

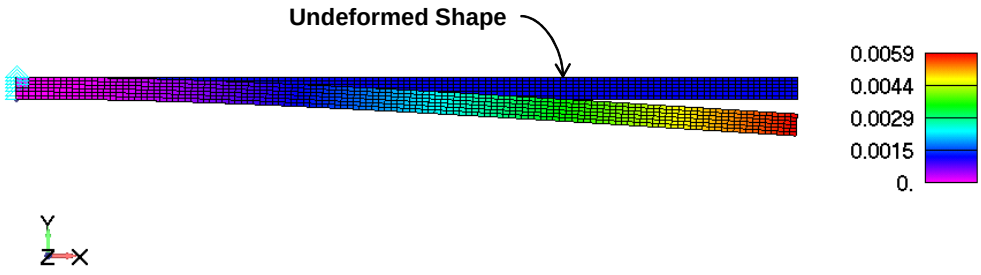

Fig. 3.15. Beam deformation after subjected to gravity for 1 hr; the final material model is used for this simulation; deformed beam contoured by displacement magnitude in meters

Two finite element models with varying mesh resolution (Figure 3.6 shows the coarse-mesh model) are constructed to ensure mesh convergence. The results of the convergence study are provided in Figure 3.16, which shows less than a 2% difference in the displacement results between the coarse and refined mesh.

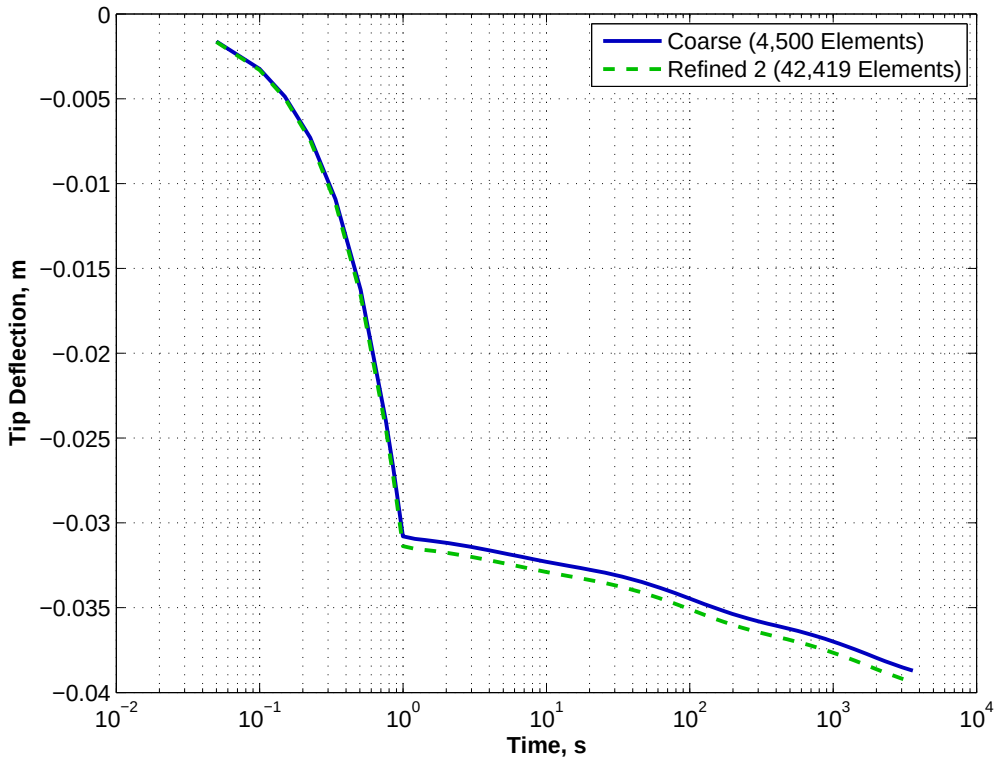

Fig. 3.16. Beam tip deflection for coarse and refined finite element meshes

Finite element model results are compared to the empirical results in Figure 3.17 using the API-derived material model and for a revised material model. As indicated by the steeper slope of the simulated response for the original API model in this figure, the material model exhibits too much stress relaxation (i.e., the beam deflection is larger than the measured deflection after viscoelastic effects have diminished), but the response to the initial "instantaneous" loading represents well the material. The initial modulus obtained via the tensile testing is accurate, and only the viscoelastic parameters require tuning (recall, the infinite modulus depends on the instantaneous modulus and the viscoelastic relaxation parameters, Equation 3.1).

The final material model compares well with the empirical results, as shown in Figure 3.17. The revised viscoelastic Prony series parameters are shown in Table 3.2. The revised parameters yield a material model that exhibits much less relaxation, as indicated by the synthesized stress relaxation data of Figure 3.18.

Table 3.2. Revised Prony series stress relaxation parameters based on beam bending test results

Component, i	γ_i	τ_i
1	0.05	4.130
2	0.1	8.195×10^{1}
3	0.1	1.610×10^{3}

The measured and computed (using the revised material model) beam deformations in Figure 3.19 show the computed deformations to be within the scatter of the empirically-derived shapes. It is therefore concluded that the viscoelastic material model with an underlying linear-elastic constitutive relationship is satisfactory for these Hapflex beams undergoing large deformations.

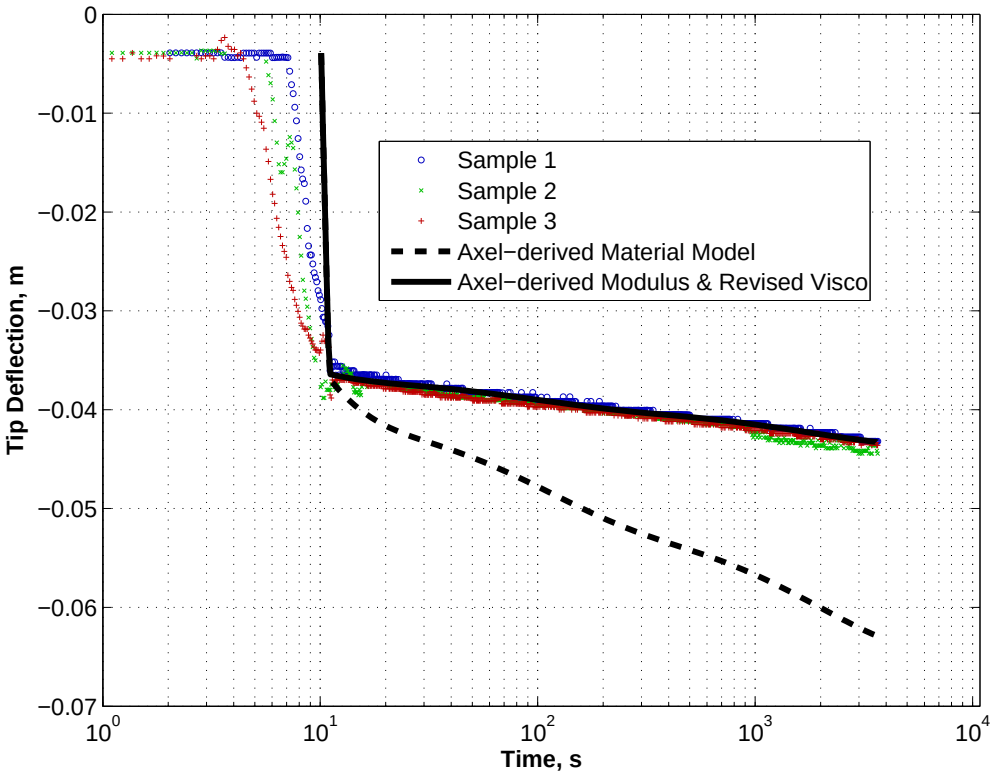

Fig. 3.17. Beam finite element model comparisons to experimental results; model results are shown for the API-derived material model for both the constitutive and viscoelastic parameters and also for the revised material model

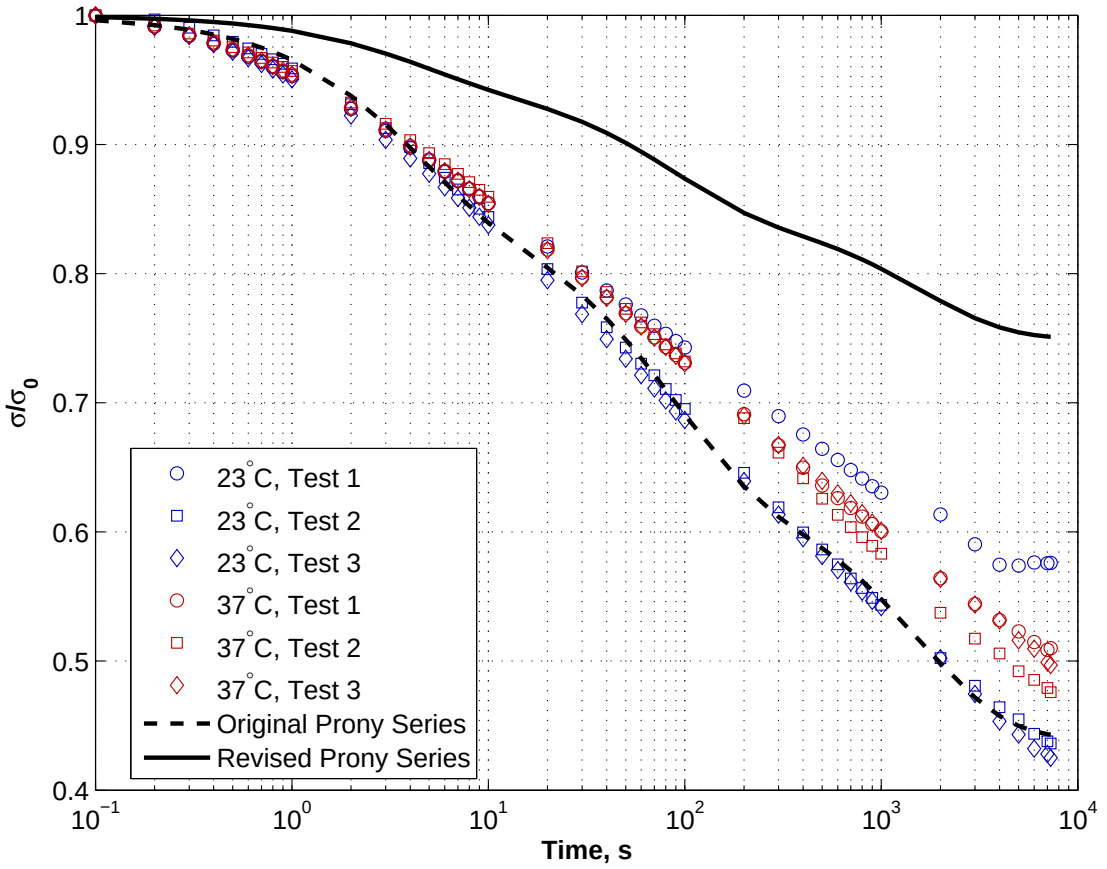

Fig. 3.18. Stress relaxation data with measured stress normalized by initial stress; synthesized relaxation curves generated using both the original and revised viscoelastic models

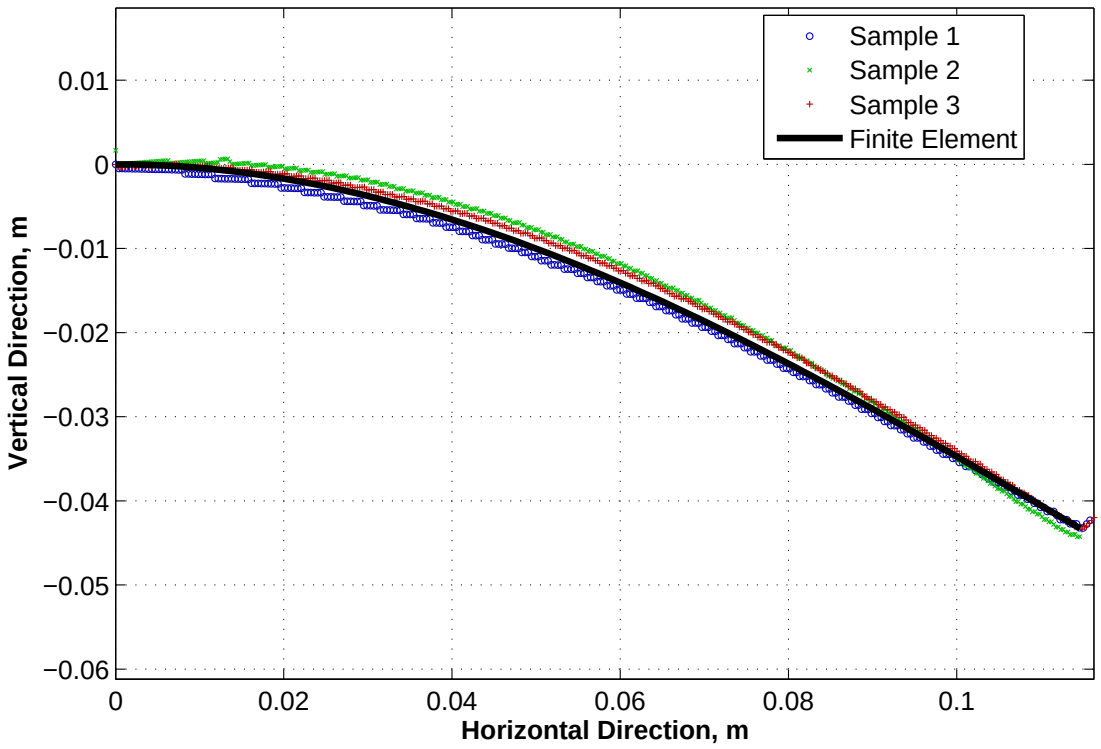

Fig. 3.19. Beam deformation shape comparisons between revised FE model and empirical results

The maximum computed strain levels from the numerical model are less than 1.2% as shown in Figure 3.20. The maximum strains considered when deriving the elastic modulus and Poisson's ratio in Figures 3.3 and 3.5 was 1.2%, which is in good agreement. Had the computed strain levels been much larger and the results disagreed, then the API data would need to be revisited. Note that the equivalent strain metric used in Figure 3.20 is similar to von Mises stress in that it enables a single strain level to be computed for a complex stress state within a structure [22, 59]. Note also that the material test results from API were used

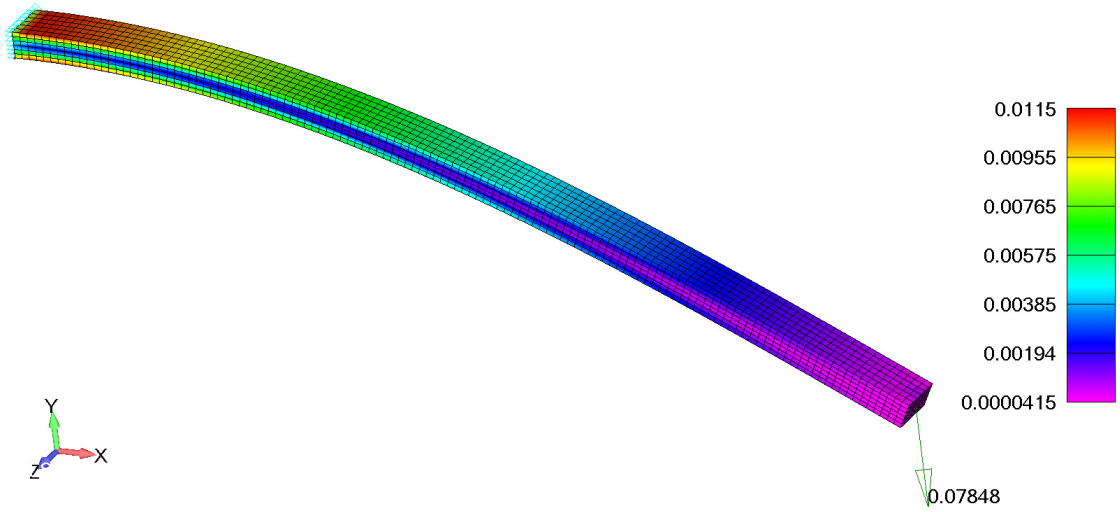

Fig. 3.20. Deformed beam after 1 hr of relaxation, contoured by equivalent strain

as the starting point for the development of the material model. It is shown herein that the material parameter estimates from these data do not represent well the Hapflex beams tested and reported herein. The underlying reason for this is unknown, but it is most likely due to lot-to-lot variations in the material. Material lot-to-lot variation is known to cause significant property variation [57].

In summary, the material modeling results have thus far shown the following:

1. A linear-elastic constitutive relationship is sufficient to model this material given a uni-directional loading and strains of less than approximately 1.2%.

2. The material is nearly incompressible with a Poisson's ratio of 0.496, but a displacement-based finite element approach with selectively reduced integration provides nearly identical results to a pressure/displacement formulation.

3. Temperature effects are not significant over the temperature range of 23 °C to 37 °C which is comparable to the desired operating range of this material for the cases to be considered during this research, 22 °C and 37 °C.

The final step in the material model development is to evaluate an FE model of the single fin that will be modeled and tested. The results of this study are provided next.

3.5.2 Modified NACA 66 Fin Model

Ultimately, the finite element solver and material model must represent well the structure to be used in the FSI simulations. Therefore, `feanl` is evaluated here against quasi-static load tests for the modified NACA 66 fin made of Hapflex 598 (the modified NACA 66 fin is described in Chapter 4). An experimental characterization of the fin deformation has been performed using a similar approach to the characterization of the Hapflex beams in the preceding section. The FE model results are compared to the empirical results below. First, however, similar to the mesh development for the Hapflex beam tests a mesh resolution study was performed to ensure the FE model is converged. Three mesh refinements were considered: coarse (4,692 elements), refined x 1.5 (15,300 elements), and refined x 2 (37976 elements). The refined x 1.5 model is shown in Figure 3.7. (The blade models were created automatically from the blade section data using the custom `bladeGen` software described in Section 5.1.2.)

The results of the convergence study, Figure 3.21, show a change in tip deflection at 3600 s between the coarse and first refinement is approximately 4%, whereas the change between the first and second refined meshes is approximately 2%.

Comparisons between the FE results and the experimental results using the previously tuned material model are shown in Figure 3.22 for the tip deflection and in Figure 3.23 the foil deformation shape after 1 hour of relaxation. The FE results in Figure 3.22 were shifted in time so that the starting time corresponds to the beginning of load application in the experiments. Note that the effect of gravity is included in the FE simulations, similar to the approach used for the beam models. Gravity alone accounts for the deformation shown in Figure 3.24.

Note that the fin tests in these figures used the same fins multiple times with a minimum of three hours between tests. The objective of this evaluation was to show the material response is not significantly influenced by multiple tests. This observation enables multiple water tunnel tests to occur with the fin, thereby eliminating the need to reinstall a new fin for each tunnel test.

Note also that Fin 08 was fabricated from a different lot of material than that used by the other fins. It was observed that the original material was becoming difficult to degas, resulting in unwanted voids in the cast parts, so a second batch was purchased from Hapco. After characterizing the fins during bending testing, and observing the substantial change in character, it was decided to return to the original material for sake of consistency. After discussing the degassing and air entrainment issue with material experts at ARL/Penn State, the original material batch (part A of the two-part material) was exposed to an elevated temperature of 66 $^{\circ}$C for 24 hours to remove the unwanted polymer chain linking that was causing issues degassing the material (the chain linking occurs in the material over time due to exposure to humid air). The ability to degas the material enabled all fin specimens created beyond Fin 08 to use the original material.

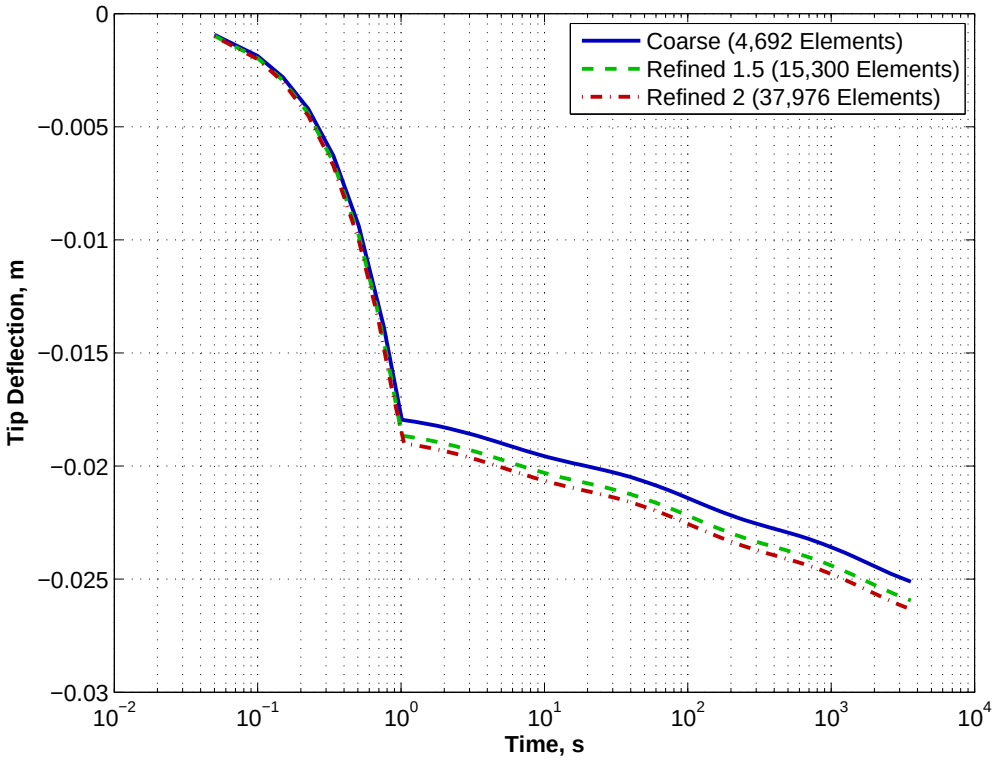

Fig. 3.21. Fin tip deformations for various mesh refinements; simulations performed using `feanl`

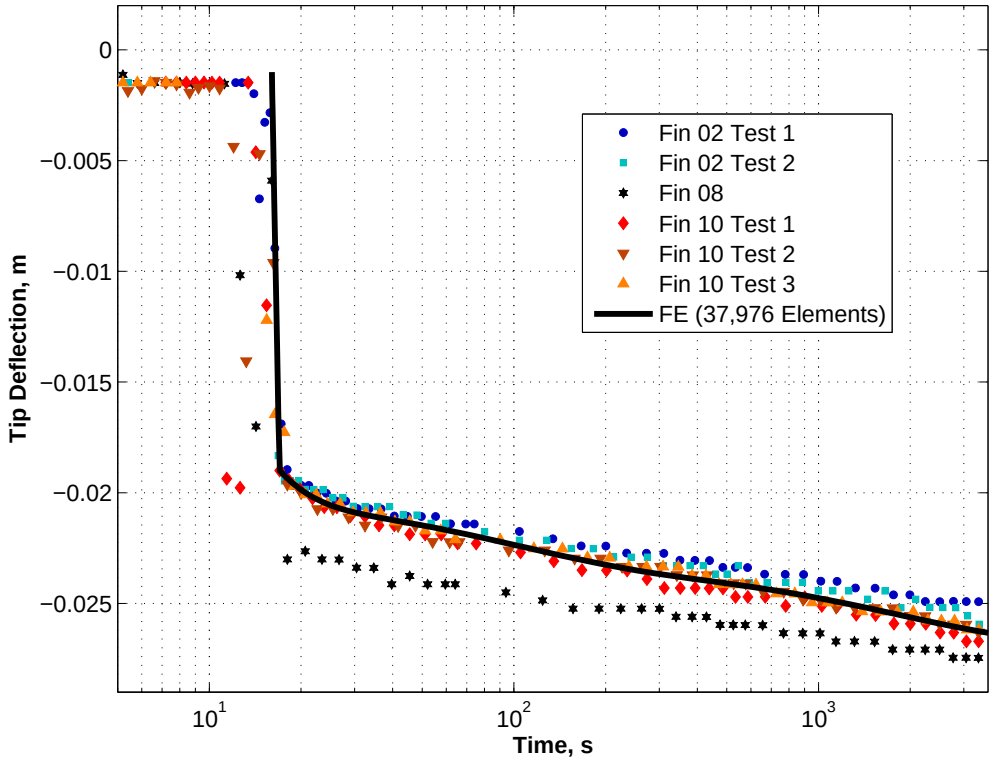

Fig. 3.22. Comparison of finite element fin model (using beam-tuned material model) to fin test results

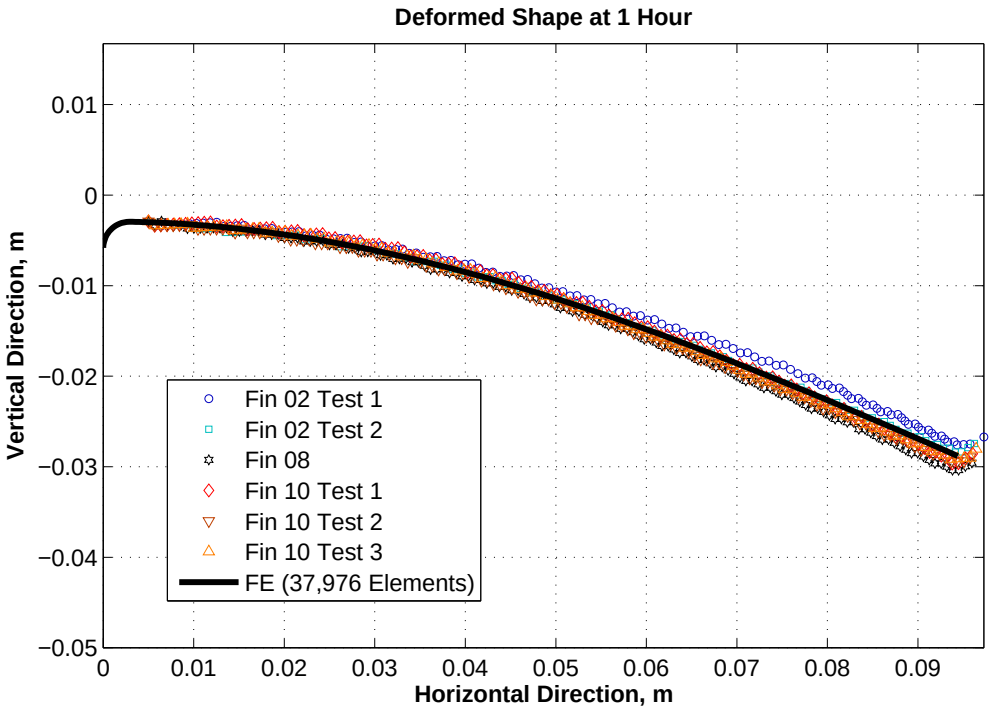

Fig. 3.23. Comparison of deformation shapes from the finite element fin model (using beam-tuned material model) to fin test results after 1 hour of relaxation

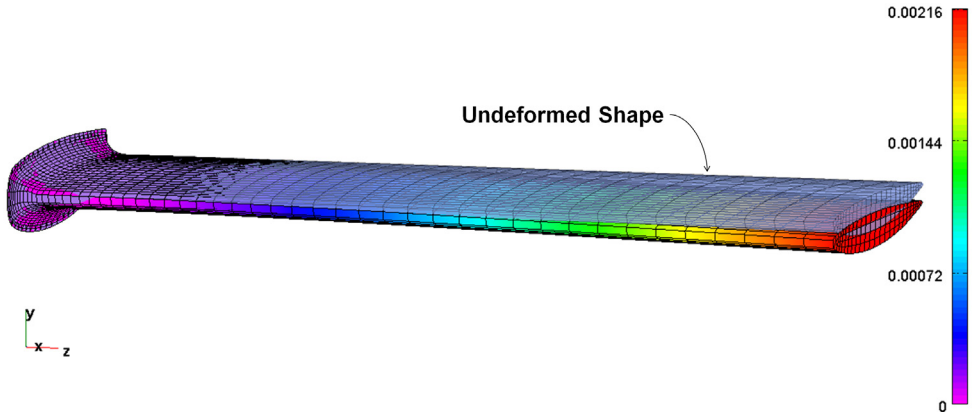

Fig. 3.24. Fin deformation after subjected to gravity for 1 hr; the final material model is used for this simulation; deformed fin contoured by displacement magnitude in meters

Chapter 4

Water Tunnel Test

A water tunnel test of a single hydrodynamic fin (Figure 4.1) is used to verify and validate the FSI solver. The fin is a modified NACA 66 with $a = 0.8$[1] camber as described by Brockett [15] and summarized in Section 4.2. This chapter first describes the experimental facility, followed by details of the foil fabrication, and concludes with results from the water tunnel tests.

4.1 Experimental Facility

The 0.3048 m (12 in.) test section diameter water tunnel at ARL/Penn State was used to create validation data for the FSI solver. The tunnel is a closed-circuit, closed-jet system with test section maximum speed of 20 m/s and absolute pressure range of 20.7 kPa to 413.7 kPa. The velocity is controlled by a mixed-flow peerless pump that is powered by a 111.8 kW (150 hp) electric motor. The water's air content is regulated by a 0.2 m^3/s bypass system. Free air in the tunnel can be observed through two transparent domes at the tunnel high points, and excess air can be vented through these domes by solenoid-operated valves. These vents must also be used during the fill and drain process of the tunnel. The water tunnel free stream turbulence level is controlled using a 0.152 m deep section of honeycomb with a 0.025 m core size positioned in the plenum upstream of the nozzle and an 11.3:1 contraction ratio through the inlet nozzle. The free stream turbulence intensity is roughly 0.3% to 0.5% over the tunnel's velocity range. [106]

[1] The parameter a represents the fraction of the chord from the leading edge over which the loading is uniform at the ideal angle of attack [1].

Fig. 4.1. Modified NACA 66 fin for water tunnel testing

While there exist two test sections for the tunnel, one circular and the other rectangular, only the rectangular section was used for this research. The test section measures 0.508 m wide by 0.114 m high by 0.762 m long. A schematic of the water tunnel is provided in Figure 4.2 and photographs of the tunnel (from the opposite side and with markings to identify key features) are provided in Figures 4.3 and 4.4. The water tunnel test section with the Hapflex fin in place is shown in Figure 4.5. Additional information about the water tunnel facility can be found in the lecture notes by Lauchle et al. [78].

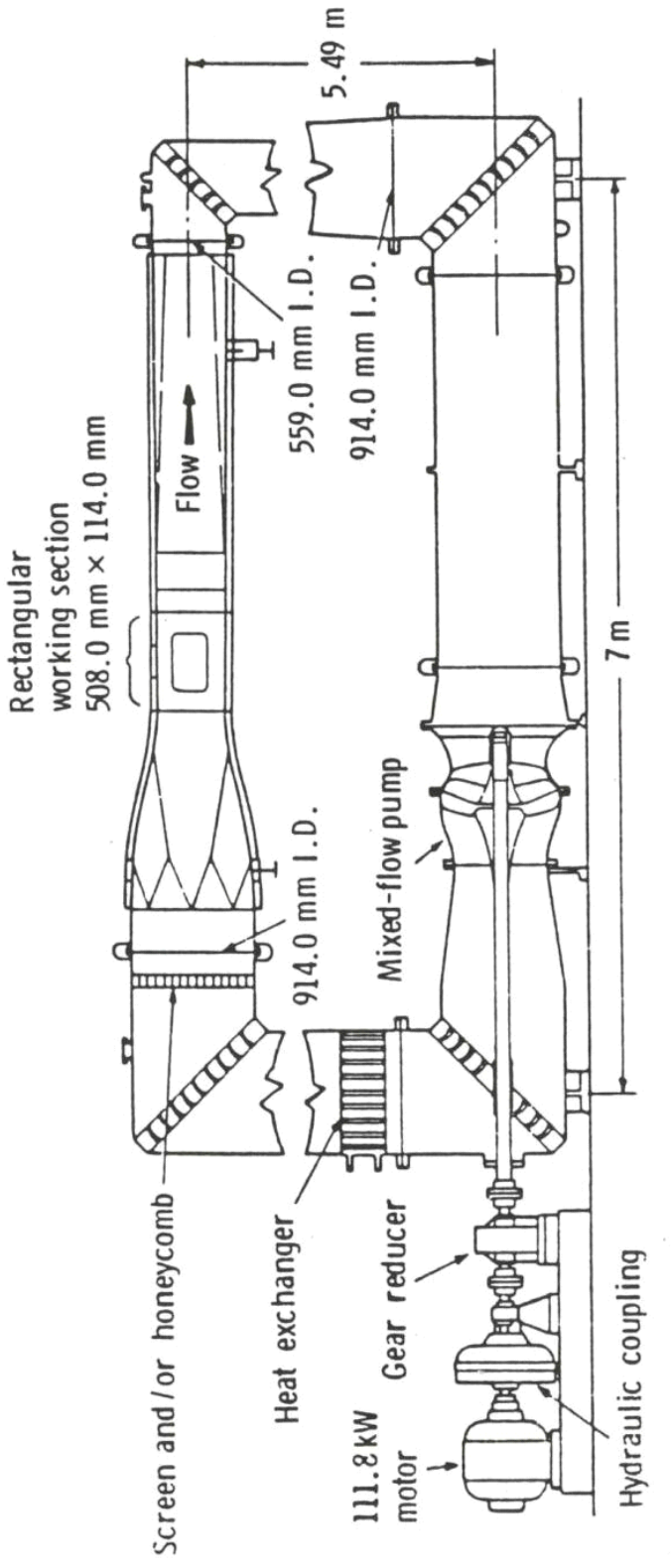

Fig. 4.2. Water tunnel schematic

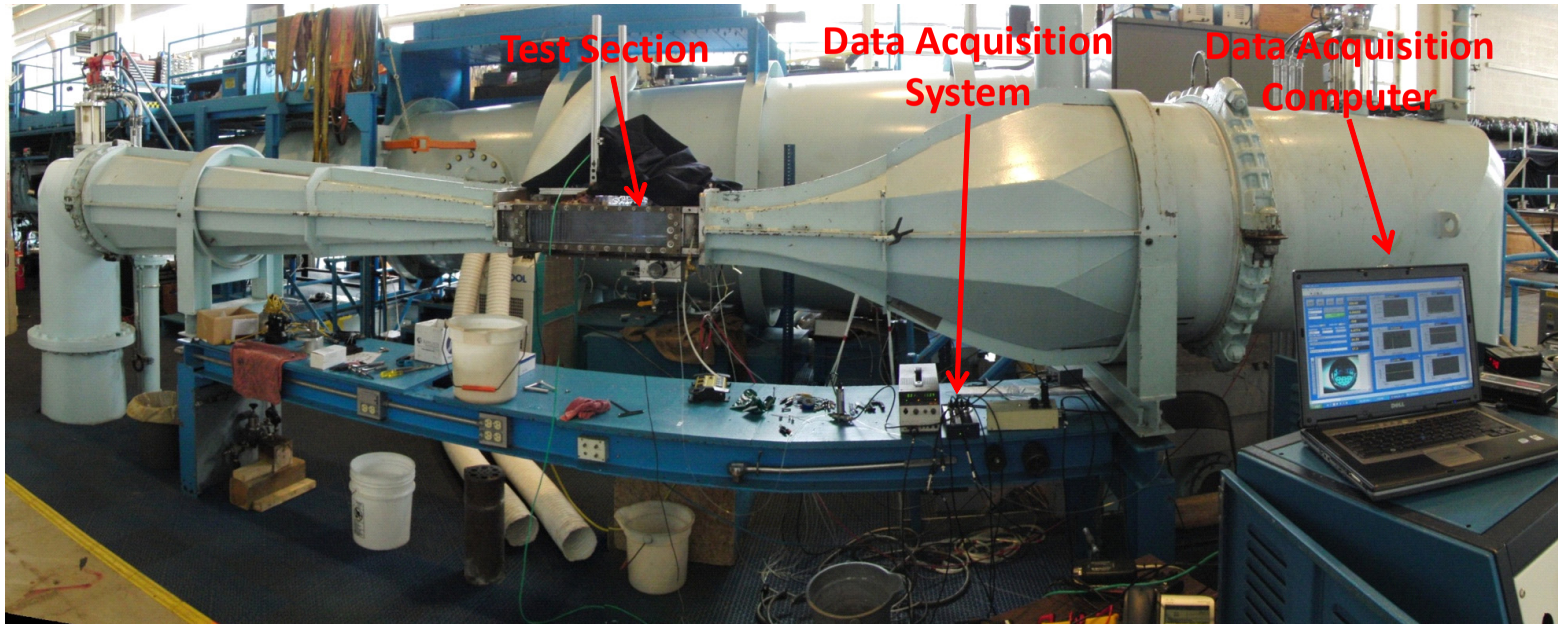

Fig. 4.3. Water tunnel and some key data acquisition system components (note that the curvature in the image is not real, but an artifact of stitching multiple photographs together into a panoramic view)

The angle of attack is estimated by matching an alignment mark on the fin with an alignment mark in the tunnel test section, Figure 4.6. This provides an approximate setting for zero AOA. The actual angle of attack is not known. Instead, this orientation is considered a baseline AOA and the fin is rotated a known amount to give baseline plus or minus a change in the AOA. The FSI simulations will first determine what simulation AOA corresponds to the baseline AOA and then also simulate changes in AOA about the baseline for model validation.

The mechanism that enables a change of the fin's angle of attack without disassembling the water tunnel is shown in Figure 4.7. This was augmented with an extension arm and a grounding bracket to accurately and quantitatively control changes in the fin AOA. The modified mechanism is shown in Figure 4.8, which is a view from below the water tunnel test section. As shown in this figure, an extension arm is affixed to the lever arm of the rotator mechanism (Figure 4.7) and constrained to a grounding bracket. The distance from the fin's rotation axis to the grounding point is 266.3 mm. By the addition of standard machinist's shim stock at this point, the angle of attack can be changed very accurately. A bar clamp is used to constrain the extended lever arm to the grounding bracket. Three different AOA variations were used during the experiments: -0.25°, -0.375°, and -0.5°. The required shim thickness for each of these AOA variations, computed using the lever arm length and the AOA change (shimThickness = radius x angle), is shown in Table 4.1.

Table 4.1. Angle of attack and required shim thickness

AOA, degrees	Shim Thickness, inches
-0.25	0.046
-0.375	0.069
-0.5	0.091

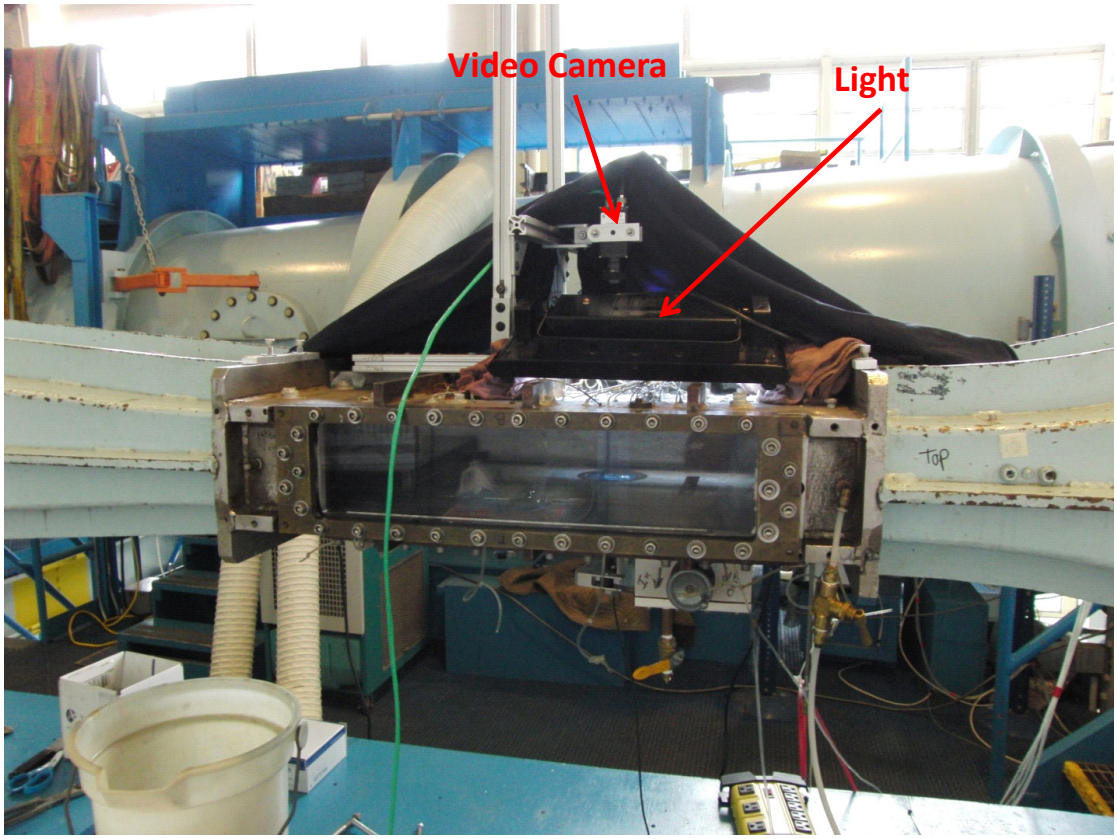

Fig. 4.4. Water tunnel test section showing video camera and light

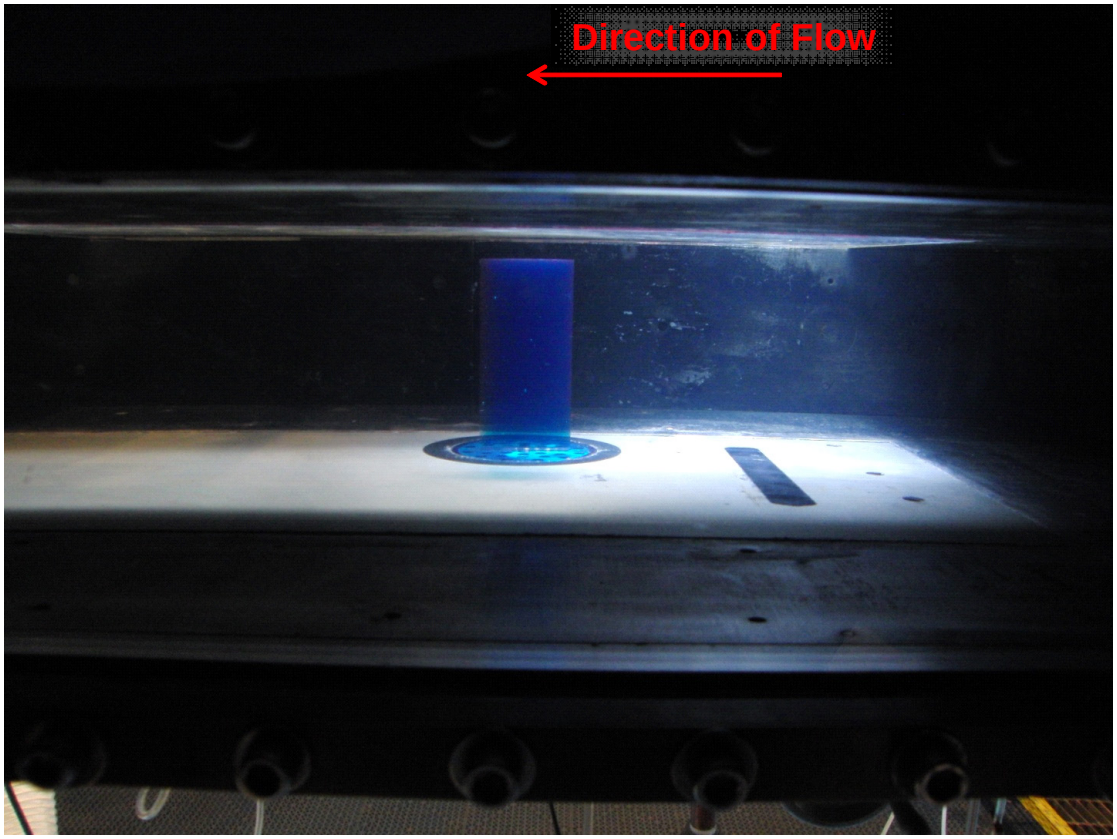

Fig. 4.5. Water tunnel test section and the Hapflex fin prior to a test

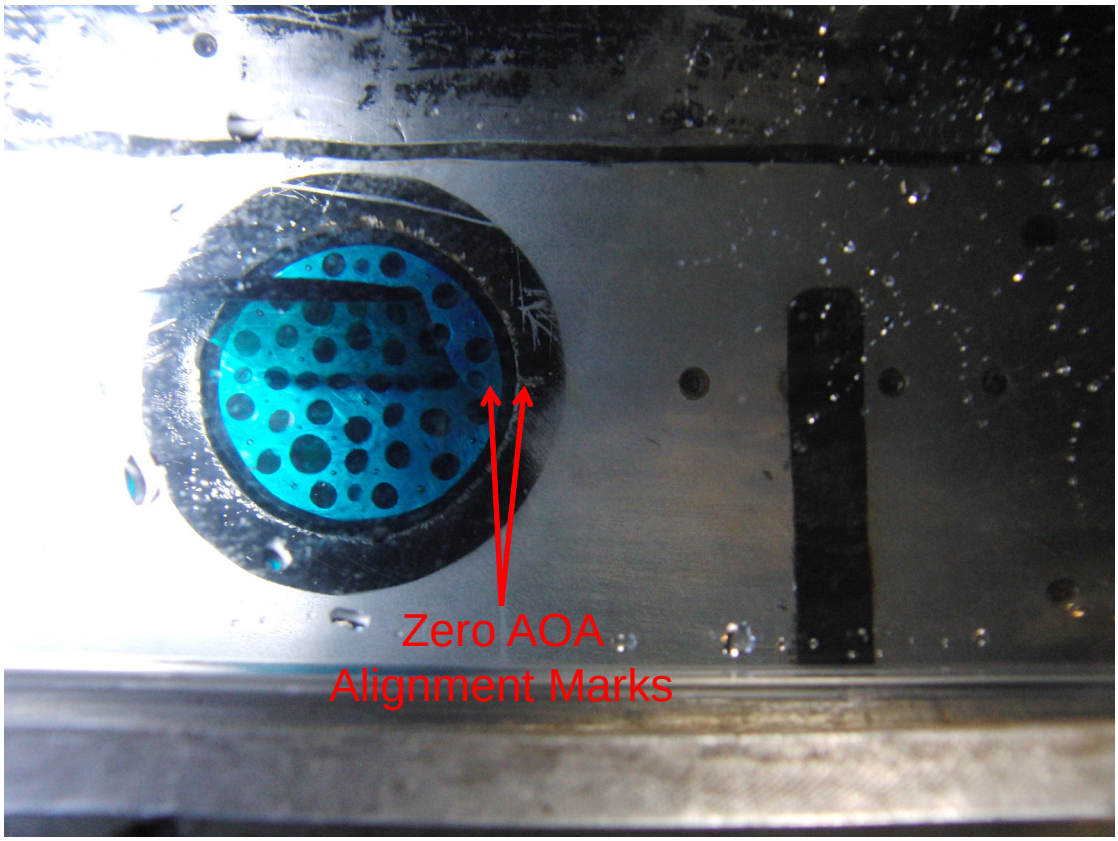

Fig. 4.6. Setting the fin angle of attack; view is from above test section beside video camera shown in Figure 4.4

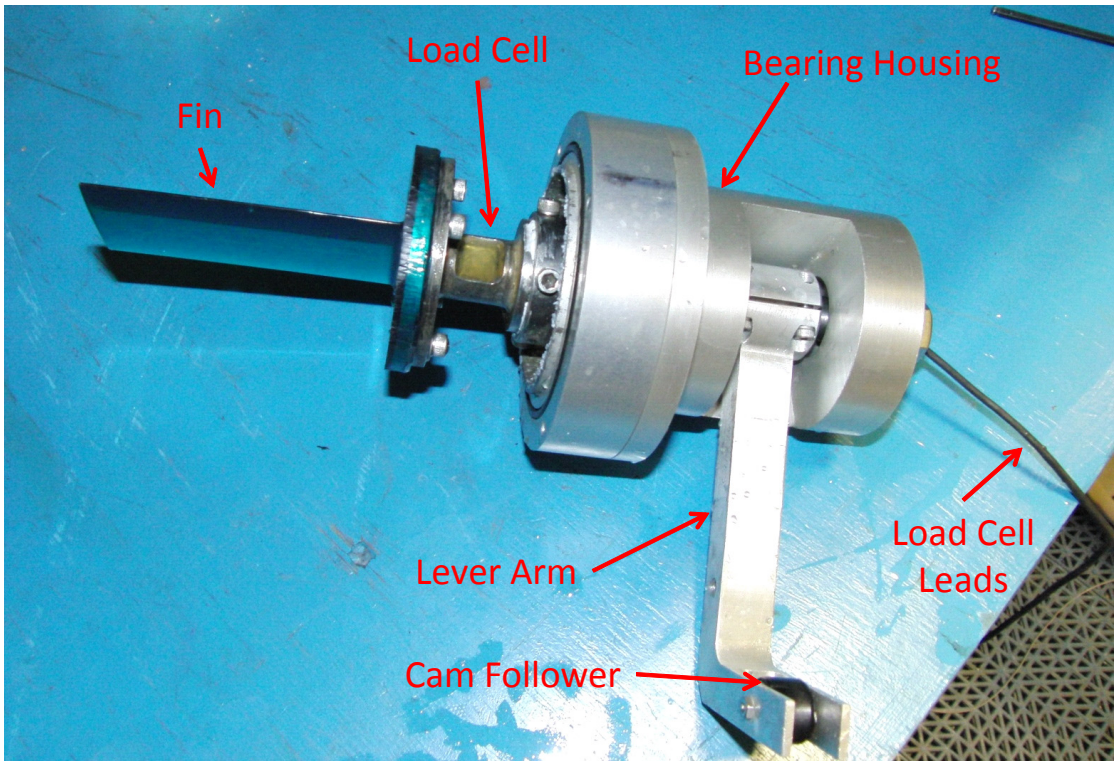

Fig. 4.7. Mechanism to rotate fin to change its angle of attack

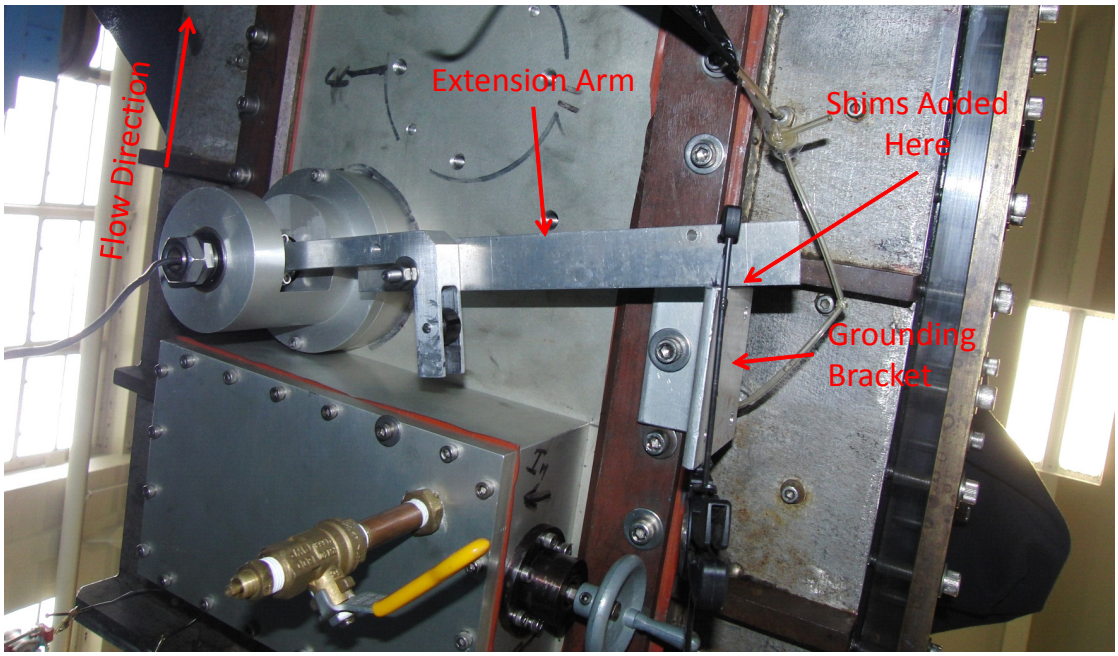

Fig. 4.8. Augmented rotator mechanism to accurately and quantitatively control changes in the fin's angle of attack; the view is looking up from beneath the test section with the rotator mechanism fully installed

The AOA settings and the tunnel speed were chosen to provide a sufficiently large fin deflection while maintaining a Reynolds number low enough to avoid transition to turbulent flow (described below). The speed chosen for the tunnel was 1.2 m/s.

The water tunnel mean water temperature ranged from 21.4 °C to 22.1 °C for the three tests reported here. The temperature, measured in the same plane as the Kiel probe, but near the tunnel wall (Figure 4.9) varied the most during startup of the tunnel due to stratification of the stagnant water, but tended toward a steady temperature as the water was mixed inside the tunnel, see Figure 4.10 and Table 4.2. These results give an average temperature for all three tests of approximately 21.7 °C. The water static pressure in the test section is maintained at approximately 1.25 bar (18.1 lbf/in^2) by the tunnel's pressure control system. This pressure was chosen primarily because it simplifies the tunnel operation (for bleeding pressure transducers and draining). Cavitation was not a concern for these tests. The water density and viscosity at these conditions is approximately 998 kg/m^3 and 1.003x10^{-3} Pa $\cdot$ s, respectively [148]. The Reynolds number based on chord length (0.05 m, described below) is therefore

$$Re_c = \frac{\rho v c}{\mu} \approx 60\text{x}10^3.$$

The critical Reynolds number for transition to turbulence is approximately 2x10^5, and therefore the flow can be treated as laminar in the FSI simulations [1].

Table 4.2. Water tunnel water temperatures

Test Number	Mean Temperature, °C	Standard Deviation, C°
47	21.4	0.15
48	21.7	0.23
44	22.1	0.16

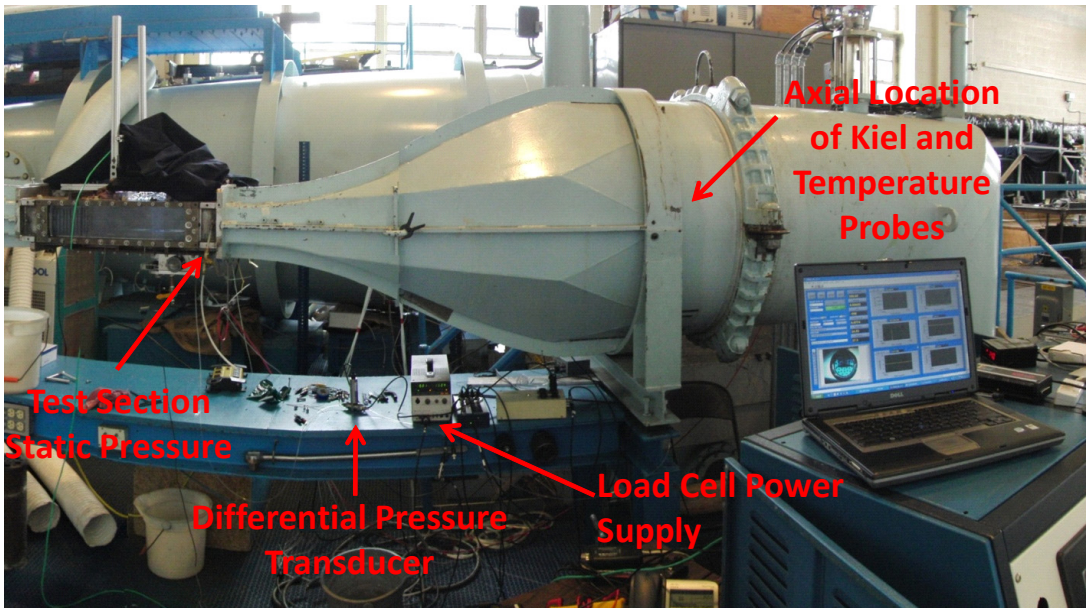

Fig. 4.9. Photograph of water tunnel annotated with pressure transducer locations; the Kiel probe is located in the central plane of the tunnel at the same elevation as the static probe

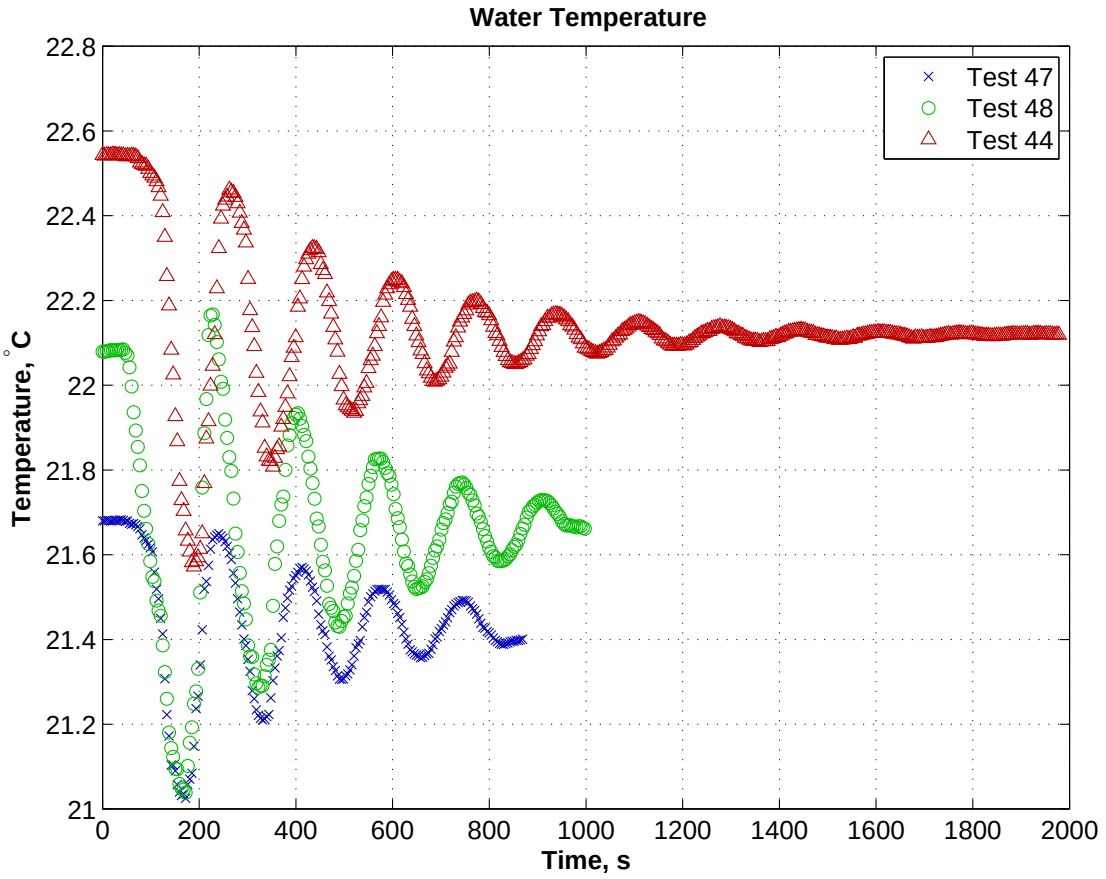

Fig. 4.10. Water tunnel water temperature during three tests; statistics of these temperatures are provided in Table 4.2

A stagnation pressure of the flow is measured just upstream of the test section nozzle using a Kiel probe (Figure 4.9). A static pressure measurement occurred just inside the test section (Figure 4.9). With the approximation of steady, incompressible, and frictionless flow along a stream line, Bernoulli's equation is applied to estimate the tunnel flow speed in the test section as follows:

$$p_k + \rho\frac{v_k^{\,2}}{2} + \rho g z_k = p_t + \rho\frac{v_t^{\,2}}{2} + \rho g z_t,$$

where the subscripts k and t represent the axial location of the Kiel probe and the test section, respectively, z is the elevation at which the pressure measurement is made, p is the static pressure, v the flow velocity, and g is the acceleration due to gravity. Because the Kiel probe is at the same elevation as the test section's static probe, the elevation effects are removed and the relation becomes:

$$p_k + \rho\frac{v_k^{\,2}}{2} = p_{sk} = p_t + \rho\frac{v_t^{\,2}}{2}.$$

With the stagnation pressure at the Kiel probe location, p_{sk}, measured directly with the Kiel probe and the static pressure in the test section measured by the static probe, the flow velocity is estimated as follows:

$$v_t = \sqrt{\left(p_{sk} - p_t\right)\frac{2}{\rho}}$$

Note that the pressure differential for a flow speed of 1.2 m/s is approximately 720 Pa (0.10 psi). The magnitude of this differential is too small to separately measure the stagnation and static pressures with the standard 100 psi transducers used for the tunnel. Instead, a Honeywell model FDW 2 psid differential pressure transducer was used to measure the pressure difference directly.

4.2 Fin Fabrication

The objective of the fin testing is to provide validation data for the FSI solver, which is to ultimately be applied to the expandable impeller pump. The fins are therefore fabricated from the same material as the pump impeller, namely Hapflex 598. A casting process is employed to fabricate the fins, which requires a mold master from which a mold can be made.

The fin considered here is constructed from blade section geometry that is commonly used for propeller design. These sections are similar to the NACA 66-006 with a=0.8 camber, but thickened near the trailing edge for ease of manufacture as described by Brockett [15]. A plot of the profile geometry is provided in Figure 4.11. The profile geometry was used to loft a blade with a chord length of 0.05 m and a span of 0.10 m. A fillet was then added to the root of the blade solid model to provide additional structural stability (see root fillet in Figures 4.12 and 4.13).

The mold master is fabricated using a rapid prototype manufacturing technique called stereolithograpy (SLA). Protogenic, a division of Spectrum Plastics Group of Westminster, CO, fabricated the part using their Accura 60 resin. The part, shown in Figure 4.12 is hand-polished to provide smooth outer surfaces. Alignment holes and a leading edge indicator have been added to the mold master for registration of the parts during the casting process and alignment of the part in the water tunnel test section, respectively. These features are identifed in the figure.

The material used in the SLA process is advertised as being similar to the more commonly known polycarbonate. It has a tensile modulus of about 3 GPa,[2] which is sufficiently stiff to use in the mold-making process described herein. Protogenic advertises dimensional tolerance of ± 0.005 in. (± 0.13 mm). The actual

[2]Information obtained from `http://www.icm-mouldmakers.co.uk` on 18 November 2009.

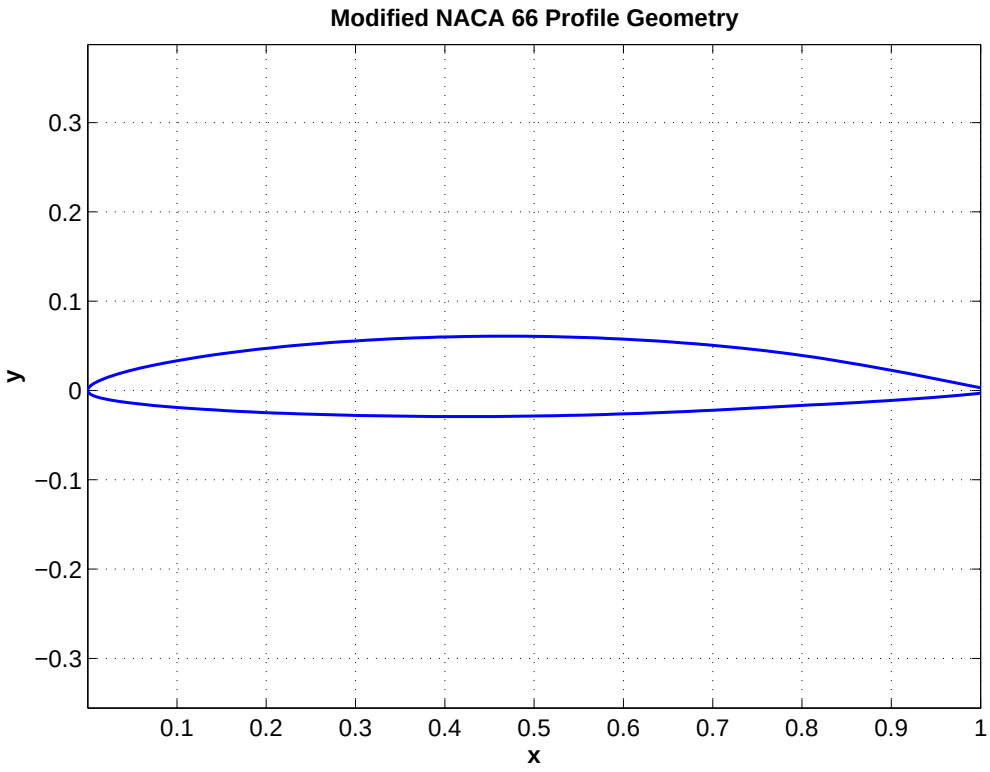

Fig. 4.11. Modified NACA 66 profile geometry

part dimensions have been measured using calipers and are summarized in Table 4.3 with percent changes relative to the design dimensions shown in Table 4.4. The results in these tables indicate excellent agreement between the SLA master and the design intent and also indicate the presence of some shrinkage of the of the cast parts relative to the mold master. Finite element models of the fin account for this shrinkage.

Table 4.3. Fin dimensions as measured with calipers for mold master and cast parts versus fin design dimensions where T is the thickness, C is the chord, and S is the span dimension (see Figure 4.13); the mean and standard deviation results are for the cast parts only

		Measured Dimensions, mm							
	Design	Master	Fin 1	Fin 2	Fin 3	Fin 4	Fin 5	Mean	Std. Dev.
T	4.37	4.37	4.27	4.29	4.22	4.34	4.29	4.28	0.0461
C	48.4	48.5	48.3	48.2	48.2	48.4	48.3	48.3	0.0585
S	100	100	100	99.5	99.6	99.5	99.6	99.6	0.216

The mold is fabricated by placing the fin master into a cavity and pouring the cavity full of mold making silicone rubber (Rhodorsil V-3040, purchased from Freeman Manufacturing & Supply Company of Avon, OH) as shown in Figure 4.14. The alignment plate and pins shown in this figure are used to register the mold to the fin master. The outer alignment pins produce a cavity in the mold for use in aligning and centering the fin base plate when casting the fins. It should be noted that the alignment plate and pins were necessary only to orient the base plate (described below) so that it provided part-to-part consistency within approximately 15 degrees so the rotary fixture of the tunnel did not need to be adjusted when changing fins. The more important alignment mark for zero AOA shown in Figure 4.12 does not require special fixturing because it is transferred to the mold and each cast part.

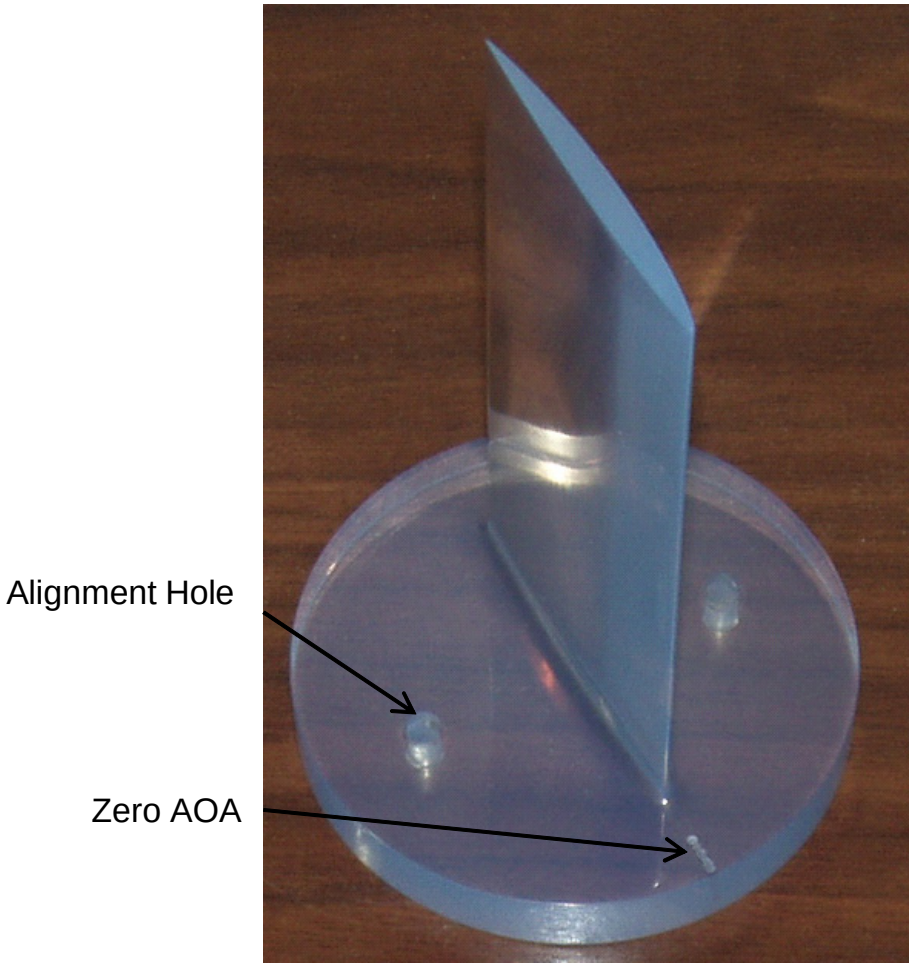

Fig. 4.12. Fin master for fin casting; fabricated from stereolithography process; surface markings added to identify the zero angle-of-attack location when mounted in water tunnel

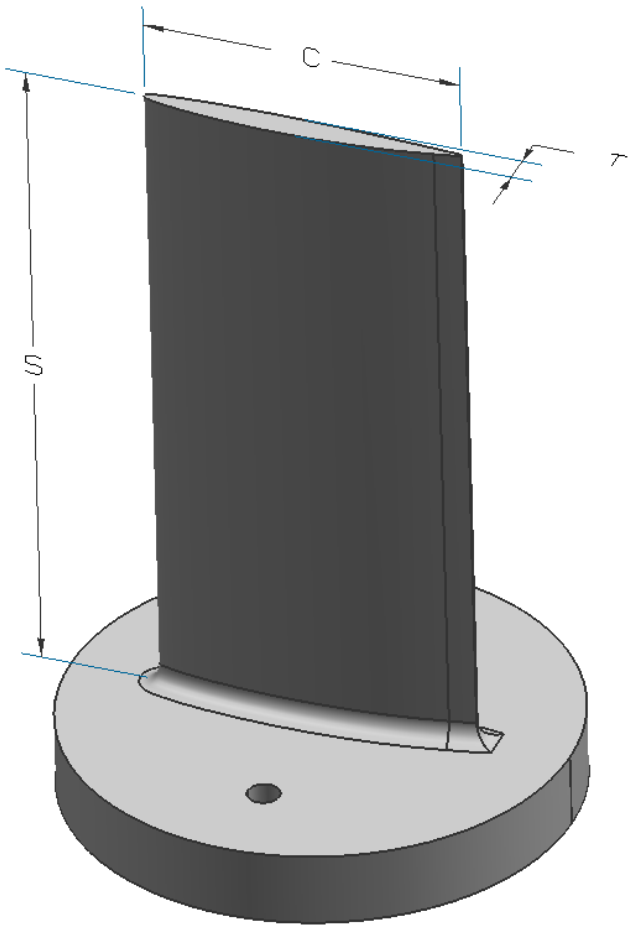

Fig. 4.13. Fin model dimensions with values provided in Table 4.3; S - span, C - chord, and T - thickness

Table 4.4. Fin dimensions as measured with calipers for mold master and cast parts versus fin design dimensions where T is the thickness, C is the chord, and S is the span dimension (see Figure 4.13); the mean and standard deviation results are for the cast parts only

		Percentage Change from Design Dimensions							
	Design	Master	Fin 1	Fin 2	Fin 3	Fin 4	Fin 5	Mean	Std. Dev.
T	0.00	0.00	-2.33	-1.74	-3.49	-0.58	-1.74	-1.98	1.06
C	0.00	0.21	-0.26	-0.42	-0.42	-0.16	-0.21	-0.29	0.12
S	0.00	0.00	0.00	-0.51	-0.43	-0.53	-0.38	-0.37	0.22

Mold pullers are added to the mold for use when removing the cast parts. These allow the mold to be pulled radially to separate the bond between the silicone rubber and the cast part.

Figure 4.15 shows a mold and the necessary hardware for casting a fin. The aluminum base plate is fully encapsulated in the cast part to provide a rigid base for mounting of the fin in the water tunnel test section via metal cap screws and threaded holes in the base plate. Details of the base plate geometry are provided in Figure 4.16. The various through-holes are used to allow the material to flow through and provide a mechanical bond. Multiple copies of this insert were fabricated on a computer-numeric controlled (CNC) milling machine to enable multiple cast parts at any given time.

The fin insert is affixed to the silicone mold using the alignment plate and pins and the Hapflex 598 is poured into the mold cavity to form the fin. The final poured mold is shown in Figure 4.17. Prior to pouring the Hapflex material into the mold cavity, the material is subjected to vacuum for approximately five minutes to remove entrained air to minimize air pockets in the cast parts. The assembly as shown in Figure 4.17 is also subjected to a vacuum immediately after pouring the Hapflex to again remove entrained air to ensure there are no air pockets in the final part. This time, the vacuum is maintained for up to 15 minutes (prior to the material "curing" too much to freely flow), but no longer or any air remaining in

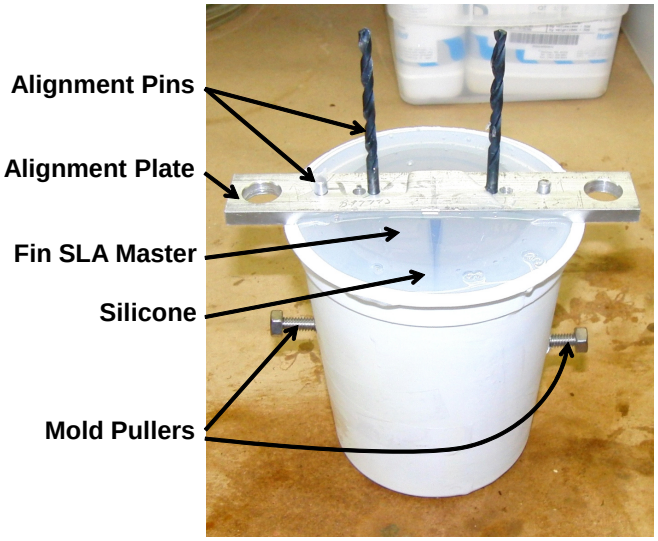

Fig. 4.14. Fin mold fabrication with alignment fixtures in place; the mold silicone is visible in this photograph

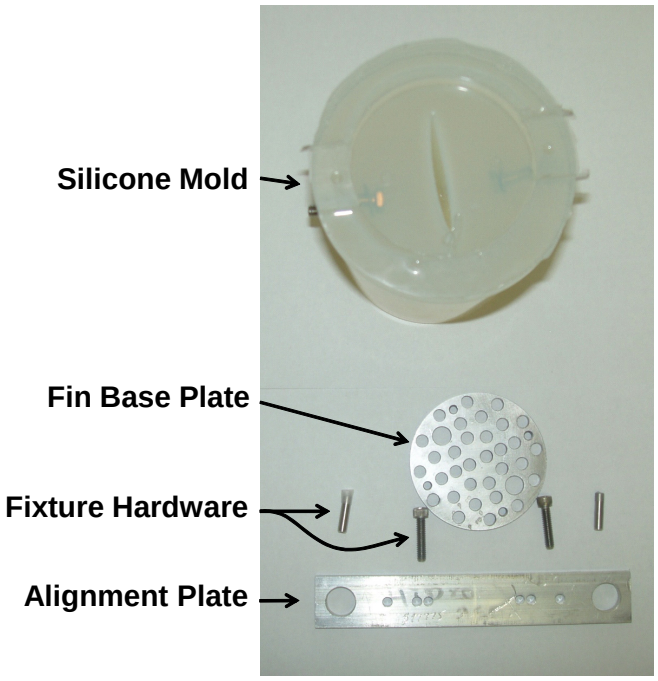

Fig. 4.15. Fin mold and casting hardware; the base plate provides threaded holes to mount the fin in the water tunnel test section

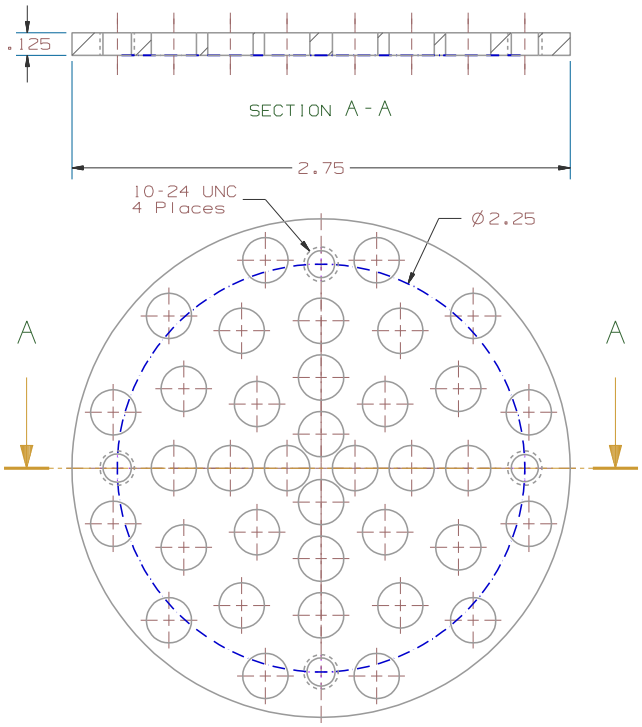

Fig. 4.16. Relevant dimensions of fin insert; through-holes added to allow material to form a mechanical lock to the insert; fabricated from aluminum

the material will produce a much larger void than if left at atmospheric pressure. The vacuum chamber and vacuum pump are shown in Figure 4.18.

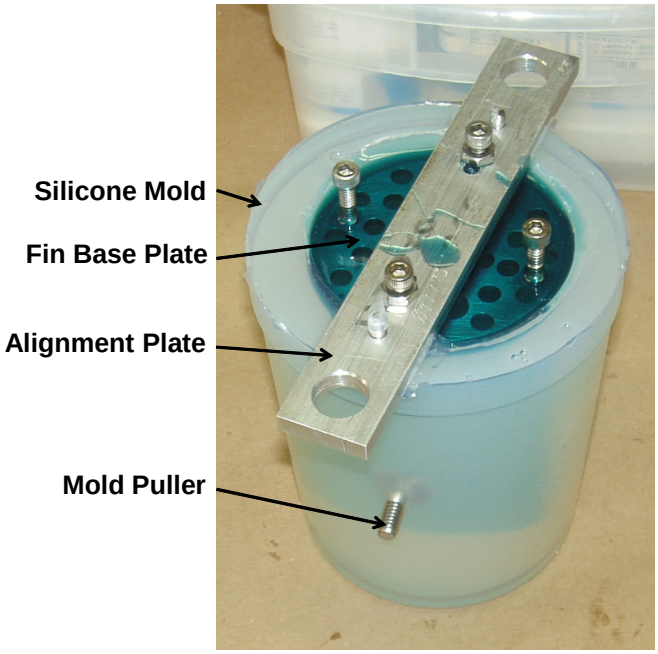

Fig. 4.17. Hapflex 598 is poured into mold cavity to for a fin for testing; screws are placed into tapped holes of fin insert to keep material from filling the holes

The cast part remains in the mold for at least 24 hours to allow the Hapflex time to cure. The part is then extracted from the mold by first removing the alignment plate, submerging the mold assembly in water, pulling on the mold puller pins (see Figures 4.14 and 4.17) to de-bond the fin from the mold, and then pulling the fin out of the mold (see Figure 4.19). Submerging the assembly in water lubricates the interface between the fin and the mold and keeps the fin from re-bonding to the mold once the bond is broken.

Several fins have been cast and measured for dimensional stability. Results are provided in Tables 4.3 and 4.4. A total of three molds and ten fins were cast using Hapflex material from two different lots.

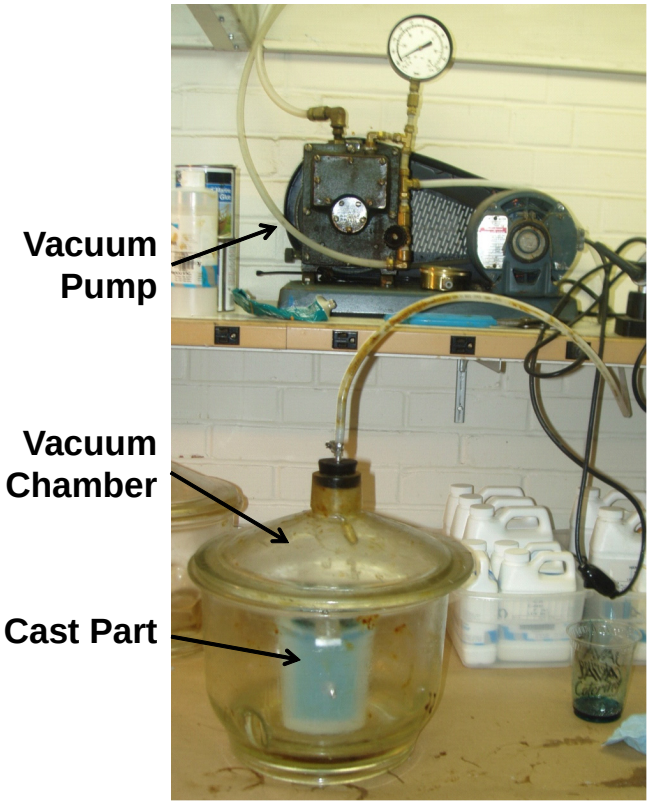

Fig. 4.18. Mold assembly shown inside a vacuum chamber to remove entrained air from the Hapflex 598 while still in the liquid state

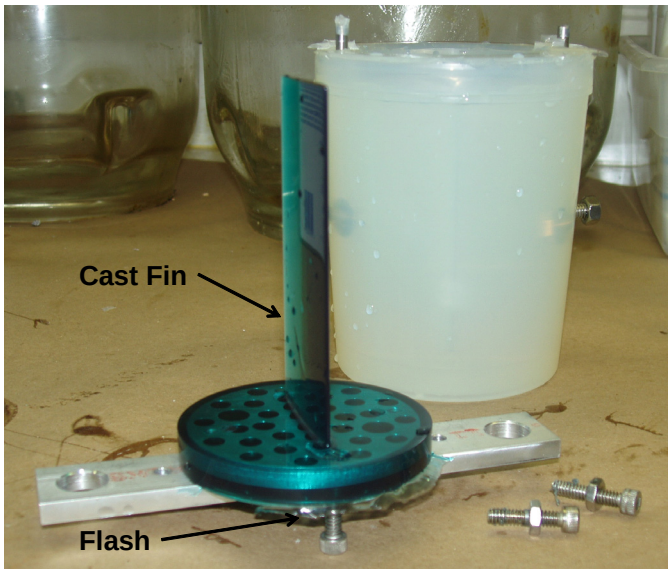

Fig. 4.19. The cast fin is removed from the mold by submerging in water (see water droplets on outer surface of mold and fin); some flash exists on cast part which is removed using a sharp knife

4.3 Test Results

The results of the water tunnel test for the modified NACA 66 fin are provided next. These data consist of flow speed and blade deformation. Note that a force cell was installed to measure fin lift force (Figure 4.7), but it proved too noisy to obtain useful information. The cell is designed for maximum lift of 50 lbf and the current application has a maximum lift of approximately 0.1 lbf. The low signal level did not provide sufficient signal-to-noise to reliably extract lift force during the testing. However, because the fin material model and structural solver are validated separately, the only need for a measured force cell would be to identify the source of error if large discrepancies exist between the predicted and measured results.

The blade deflections are quantified from the video data recorded from a location above the test section as shown in Figure 4.4. For quantitative comparisons to the simulation results, tip deflections at the leading and trailing edges are required. Because of the poor contrast between the fin tips and the background (i.e., the bottom of the tunnel test section), the video frames were processed manually. A sample frame from one test video is shown in Figure 4.20, which shows a line connecting the LE and TE tip points that are manually selected.

To avoid large perspective distortion as the blade deforms and changes its distance from the camera, the flow speed and fin's AOA were limited to control the blade tip deflection to less than about 20% of the blade chord, which results in a blade tip elevation decrease of approximately less than 0.5 mm based on the solver simulation results presented below. The measured fin deflections are scaled based on the known chord length of approximately 0.05 m at the fin tip. Results for three test cases are shown in Figure 4.21 for the leading edge tip deflection and in Figure 4.22 for the trailing edge tip deflection. The results are also plotted in terms of pitch and heave in Figures 4.23 and 4.24, respectively. A plot of the flow speed for each test is shown in Figure 4.25. The tip deflection excursions shown in the figures are a direct consequence of water tunnel flow rate excursions. This is demonstrated by the plots in Figure 4.26.

One approximation that is made in the FSI modeling for this effort is that the system is quasi-steady. Ignoring the small-amplitude deflections of the fin attributed to large-scale flow structures that are visible in the recorded videos, the response of the fin observed here is quasi-steady and can be quantified using the reduced frequency parameter as follows. The reduced frequency, k, is often used to characterize the degree of unsteadiness for a system and is defined based on the hydrofoil semi-chord:

$$k = \frac{\omega c}{2U}, \tag{4.1}$$

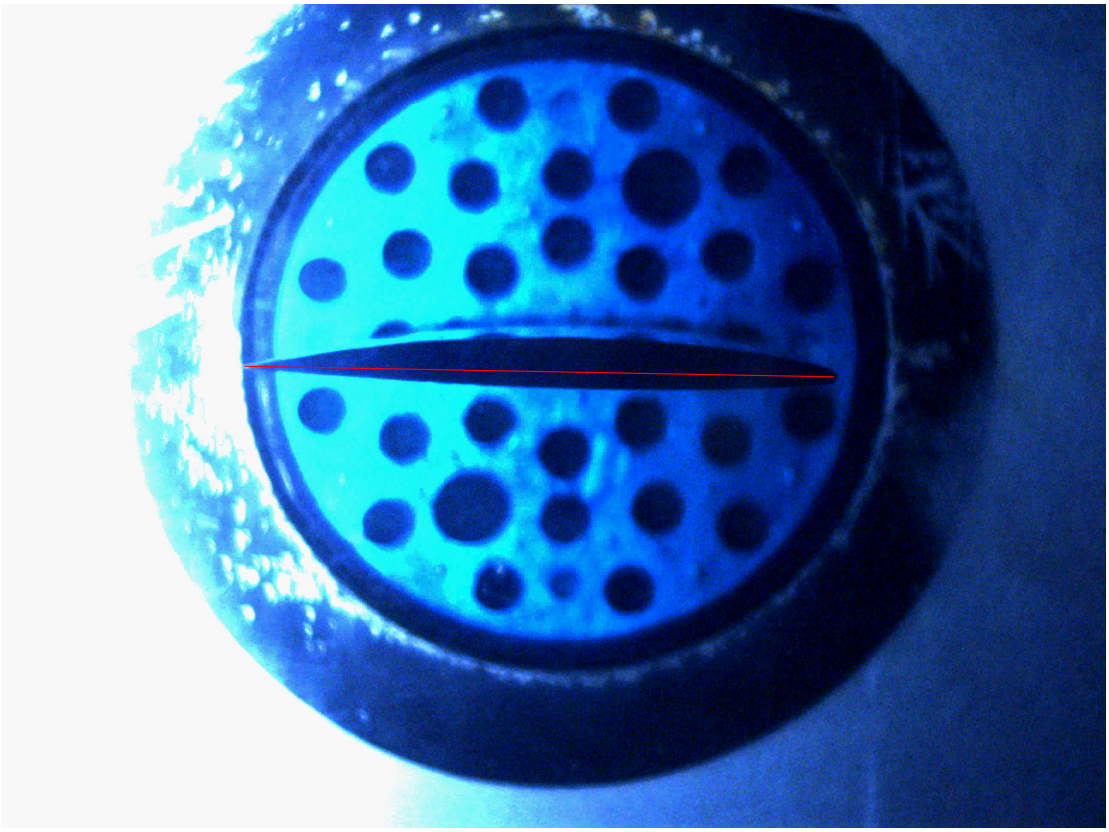

Fig. 4.20. Sample video frame from water tunnel testing; red line connects the manually picked points at the tips of the leading and trailing edges

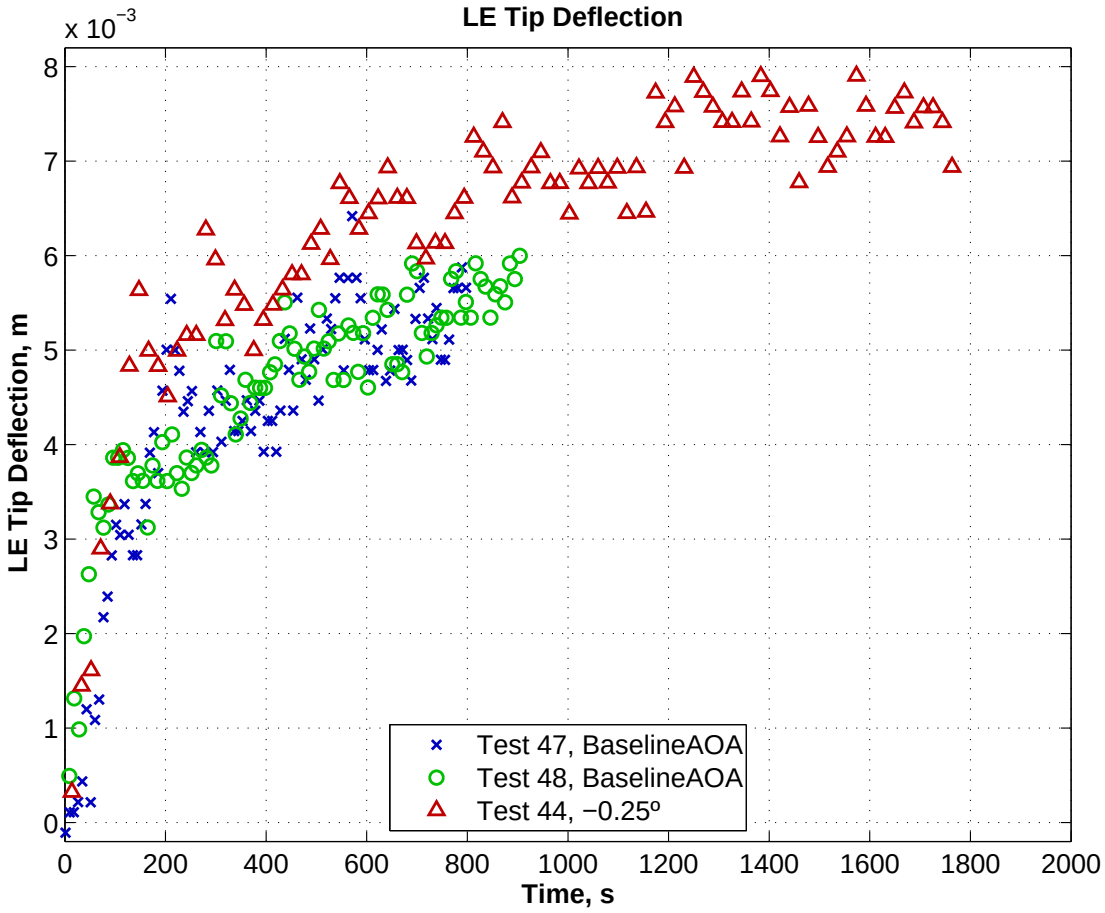

Fig. 4.21. Hydrofoil leading edge tip deflection

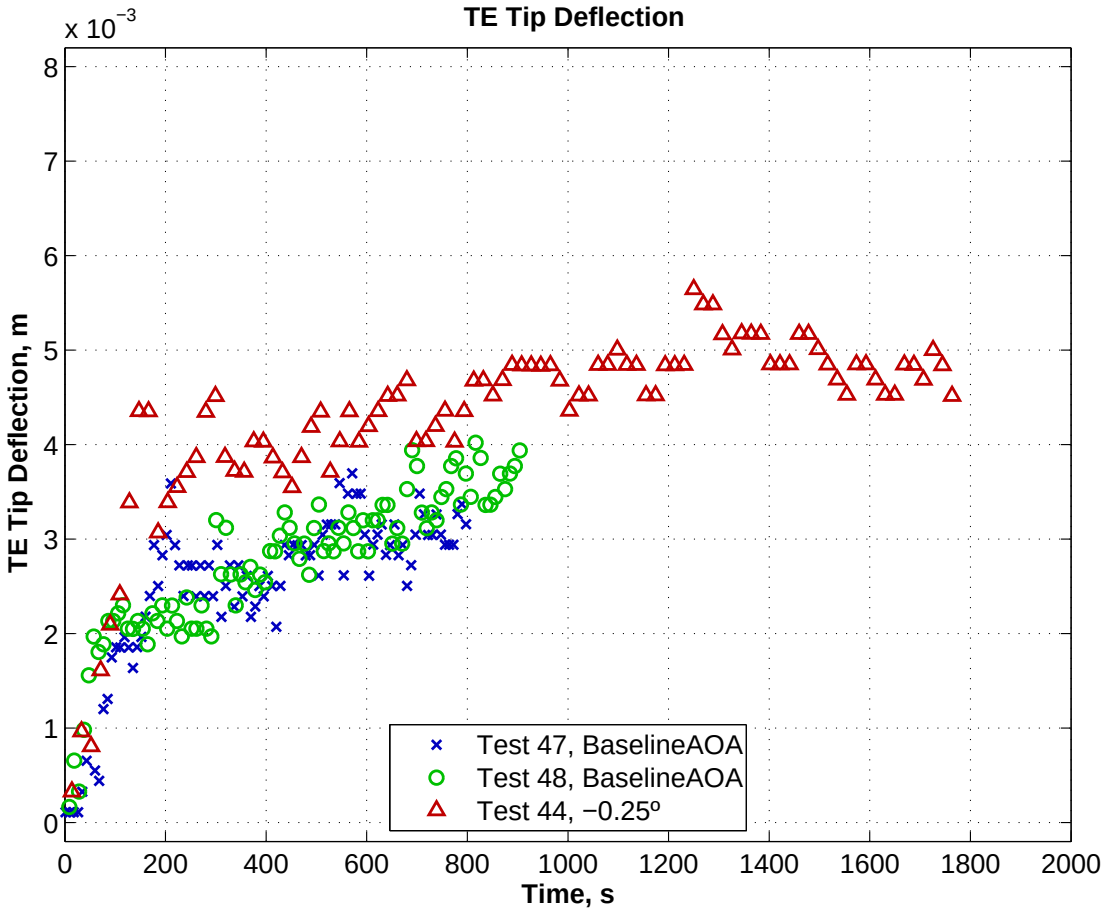

Fig. 4.22. Hydrofoil trailing edge tip deflection

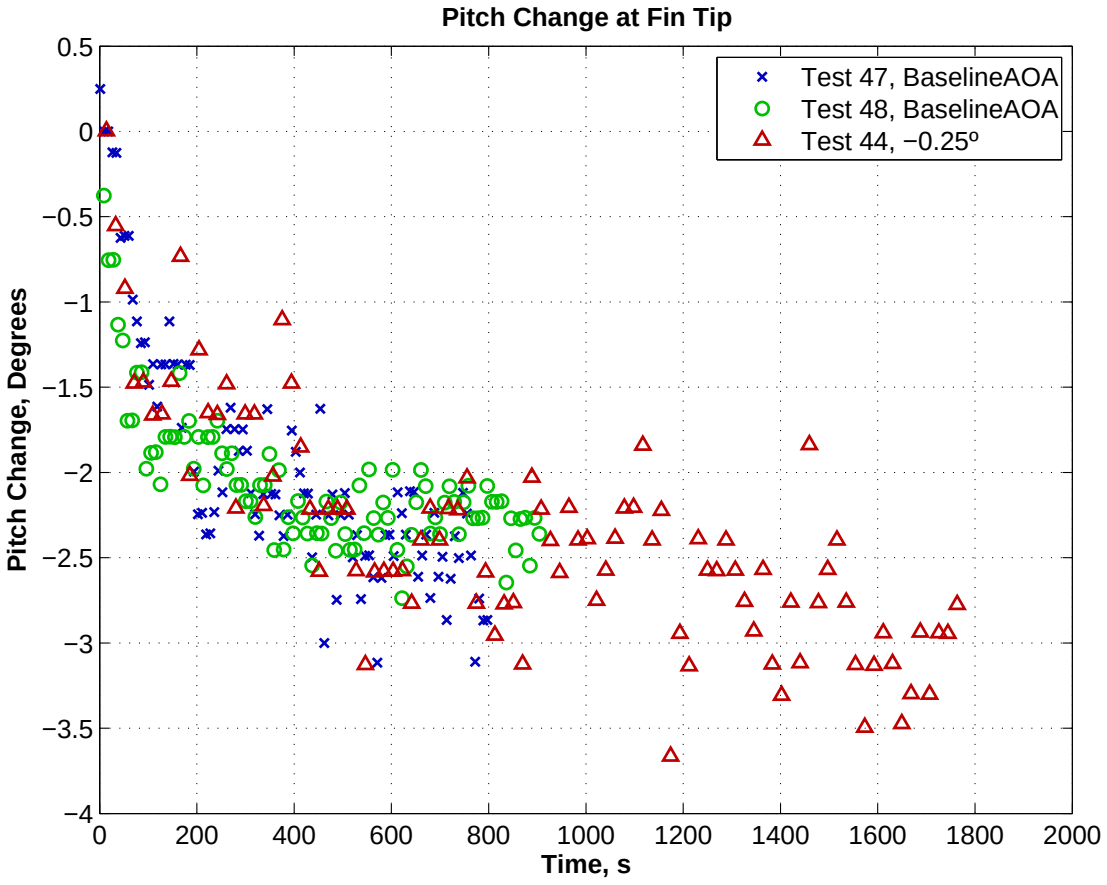

Fig. 4.23. Hydrofoil pitch angle change at the fin tip

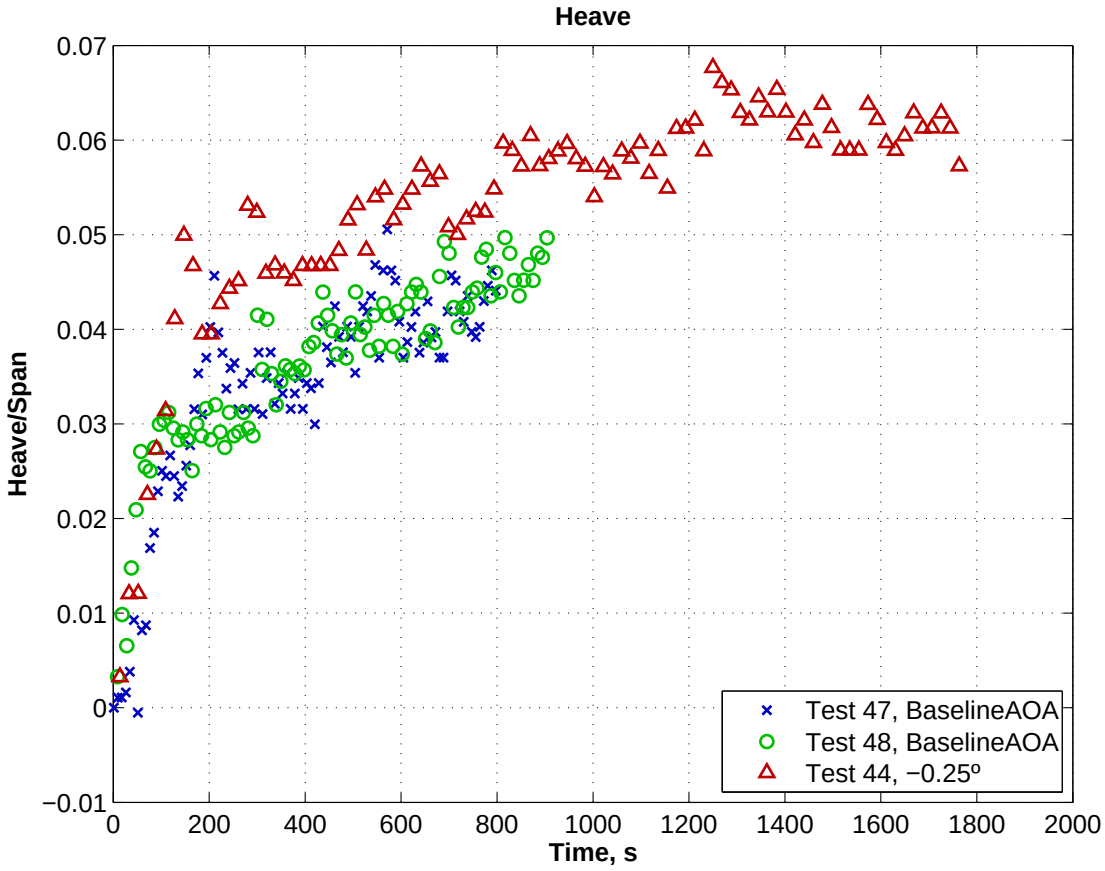

Fig. 4.24. Hydrofoil heave at the fin tip, normalized by the fin span

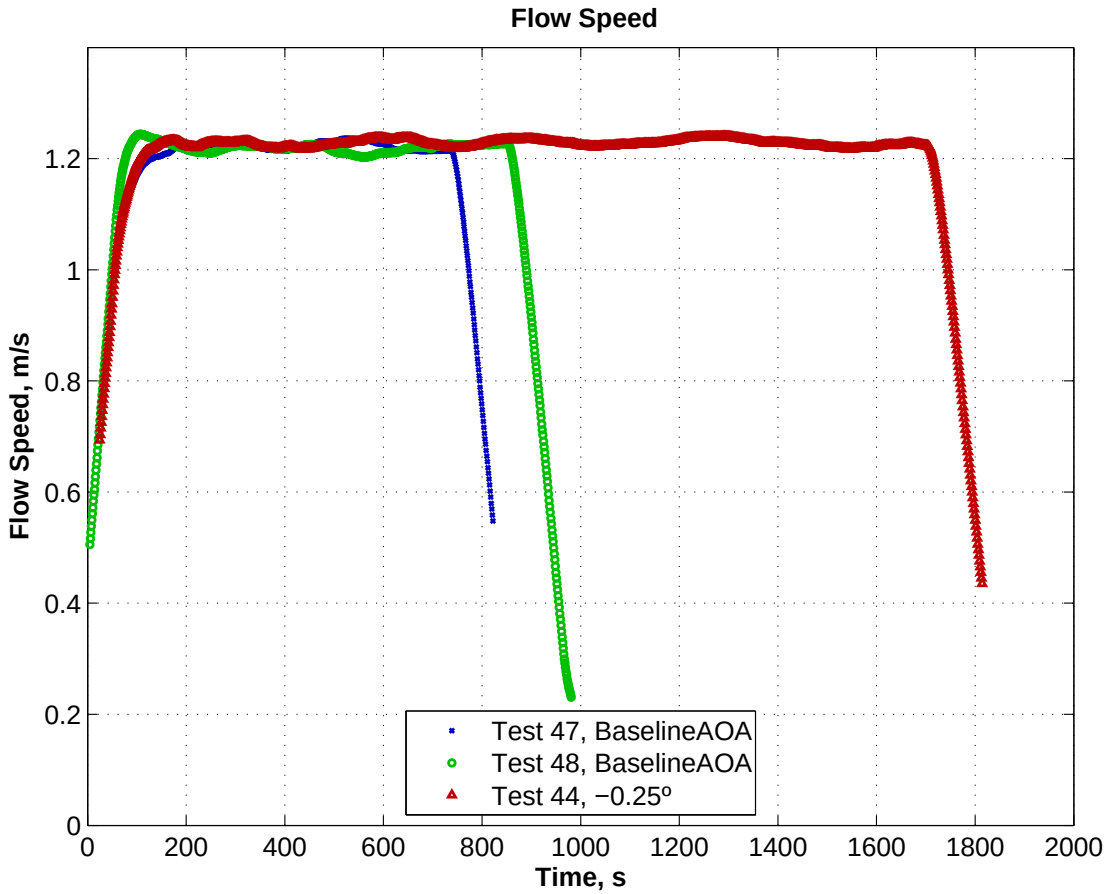

Fig. 4.25. Water tunnel flow speed in test section, corresponding to fin tip deflections in Figures 4.21 through 4.24

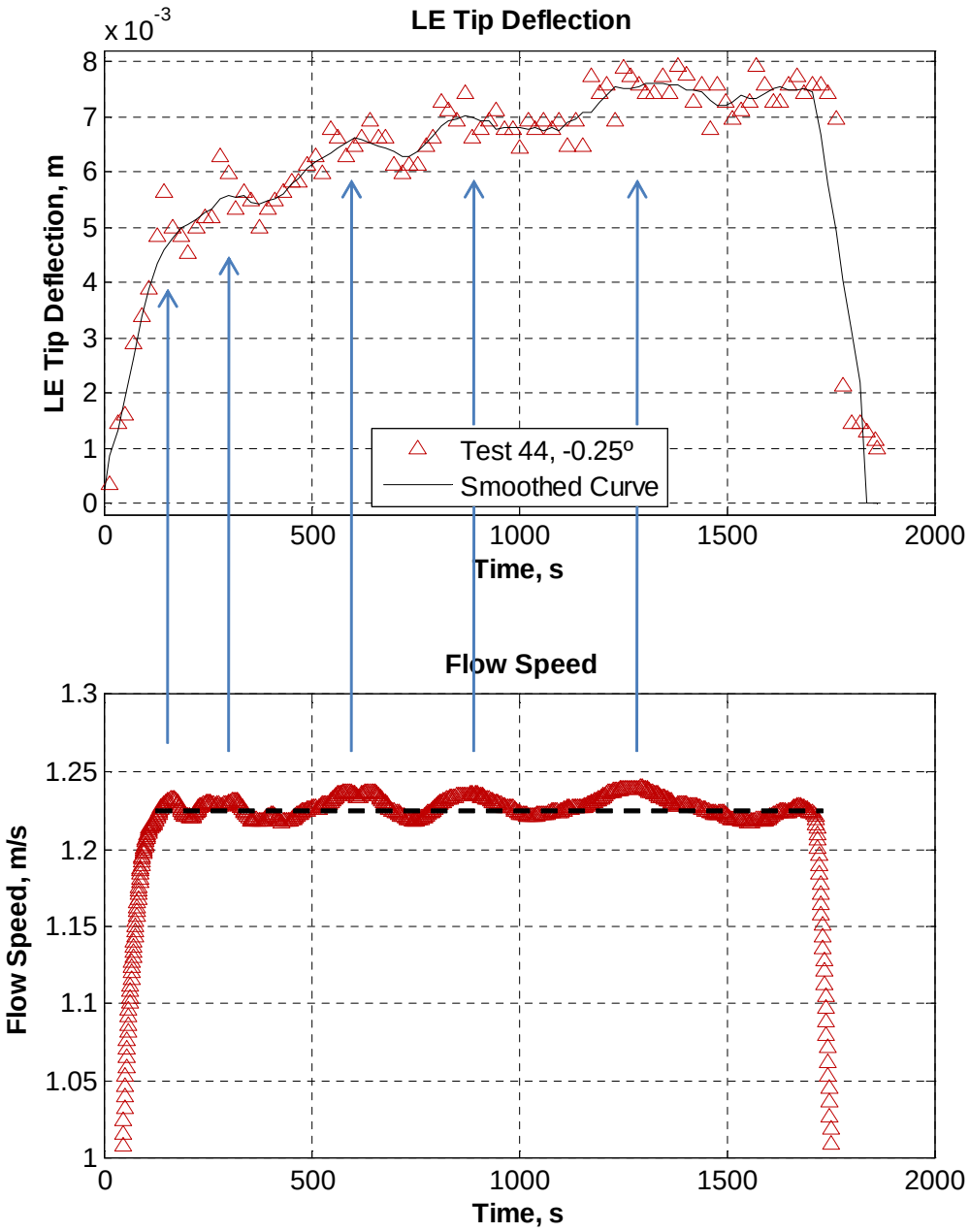

Fig. 4.26. Flow speed excursions strongly correlated to displacement excursions

where ω is the oscillating angular frequency, c is the chord length, and U is the flow speed. For k less than 0.05, the flow can be considered quasi-steady [80]. This yields a maximum frequency of approximately 0.4 Hz for the current system to be considered quasi-steady. Note that k is meant for periodic unsteadiness but is used here to qualitatively demonstrate the quasi-steady nature of these water tunnel tests. The time-averaged response (obtained by fitting a smooth curve, similar to the smoothed curve in Figure 4.26, through the data to ignore the flow-rate induced unsteadiness that is not part of the simulations) of the fin shown in Figures 4.21 and 4.22 has a frequency character several orders of magnitude less than the limit. Therefore, the FSI simulations of this system can be treated as quasi-steady.

The water tunnel test results presented here provide a validation data set for the FSI solver. Multiple angle of attacks were tested to address uncertainty in the actual angle of attack. The approach to validating the FSI solver is to infer the baseline angle of attack by simulating various attack angles until the simulation and water tunnel test results are in agreement. Comparisons of simulated and experimental results at multiple angles of attack are then used to assess the validity of the simulation results.

Chapter 5

FSI Solver Validation

The use of an elastomeric material for a turbomachine results in time-dependent performance characteristics because of the material's viscoelastic response to applied loads. The impeller's response to fluid loads can be non-linear due to not only viscoelasticity, but also due to large displacements. The FSI solver developed herein is used to evaluate the structural inverse method for viscoelastic materials. In preparation for this, the solver is validated and verified through experiments of a modified NACA 66 fin fabricated of the same Hapflex material as the expandable impeller pump. Similar to other elastomeric materials, Hapflex 598 exhibits a non-linear response when deformed to sufficiently high strains and the displacements are time dependent due to stress relaxation. However, the strains for this research are small enough to employ a linear-elastic modulus for the instantaneous part of the material model. The development and validation of the material model are described in Chapter 3. The structural solver validation is described here followed by validation of the FSI solver. First, however, a description of the mesh generation used for both the fluid and structural domains is provided.

5.1 Mesh Generation

5.1.1 Flow Domain

The flow domain is meshed with GridGen [110], which supports the creation of either structured or unstructured meshes. The flow model of the water tunnel test configuration uses a structured mesh with 256,000 hexahedral cells. The mesh

resolution was evaluated using a coarsened mesh having 45,200 hexahedral cells and a refined mesh having 664,000 hexahedral cells.

The mesh was created by first constructing a boundary mesh on the fin's surface, extruding the surface mesh to create an "o-grid" (the "peanut-shaped" mesh around the fin in Figure 5.1), and then the outer volume between the o-grid and the boundary surfaces was meshed, as shown in Figure 5.2. A top view of the entire mesh is shown in Figure 5.3.

While the cells are uniformly spaced in the vertical direction on the domain outer boundaries (Figure 5.4) they are concentrated in the tip region between the fin and upper wall of the test section, as shown in Figure 5.5. The cells are also concentrated in the flow-direction near the leading and trailing edges of the foil, as shown by these figures. The cells are concentrated near the leading and trailing edges because the flow field experiences large spatial gradients in these locations.

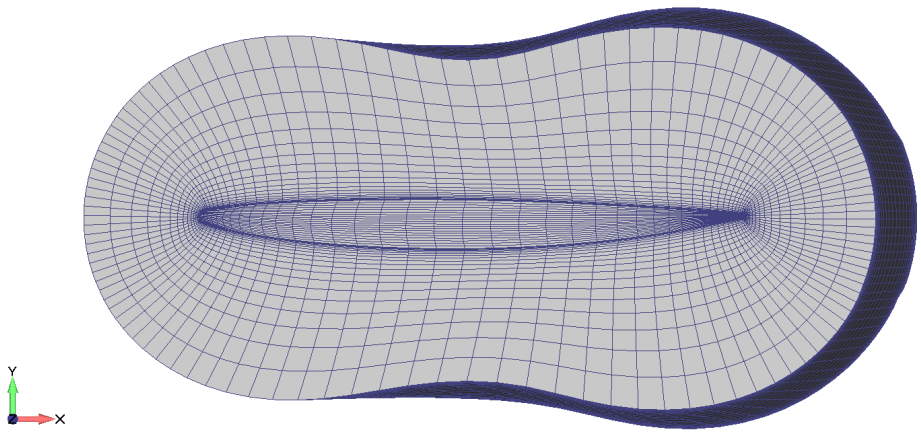

Fig. 5.1. Top view of "o-grid" during mesh construction for simulation of water tunnel test

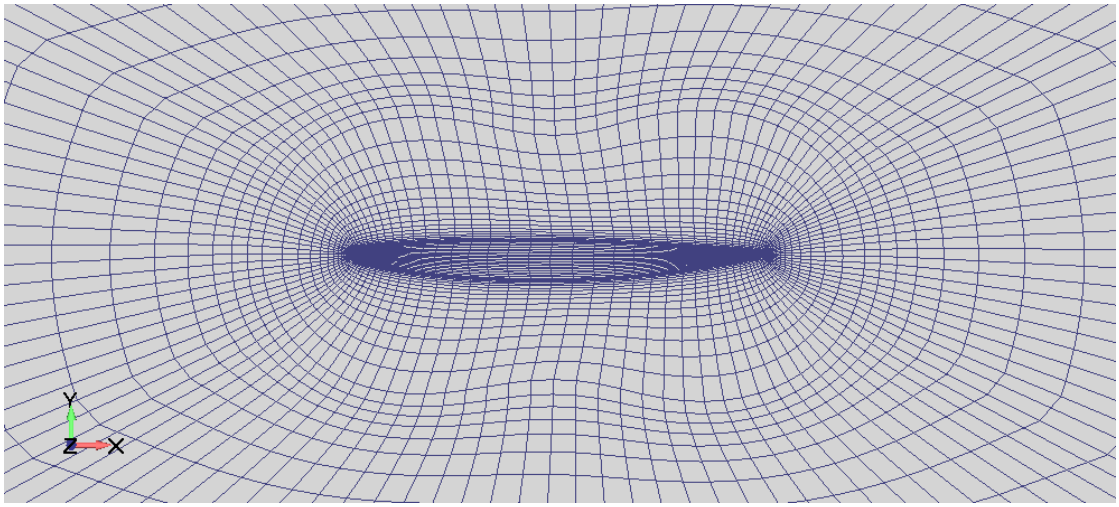

Fig. 5.2. Top view of water tunnel meshed flow domain, zoomed to show close-up of foil region

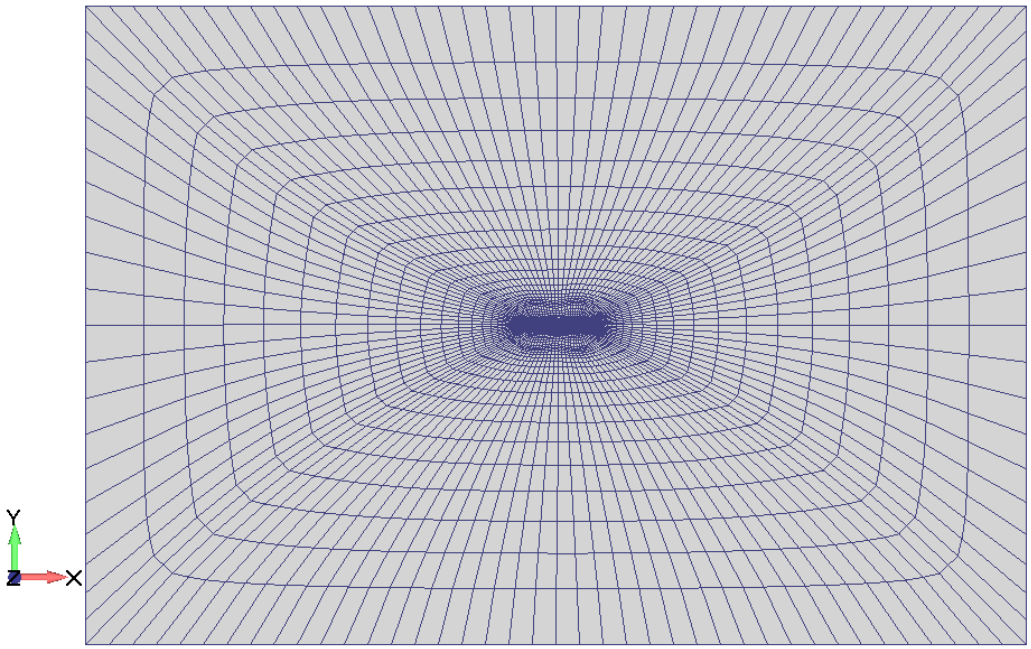

Fig. 5.3. Top view of water tunnel meshed flow domain

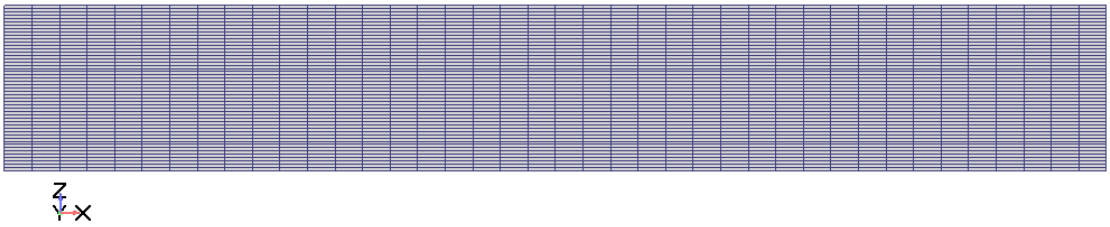

Fig. 5.4. Front view of water tunnel meshed flow domain

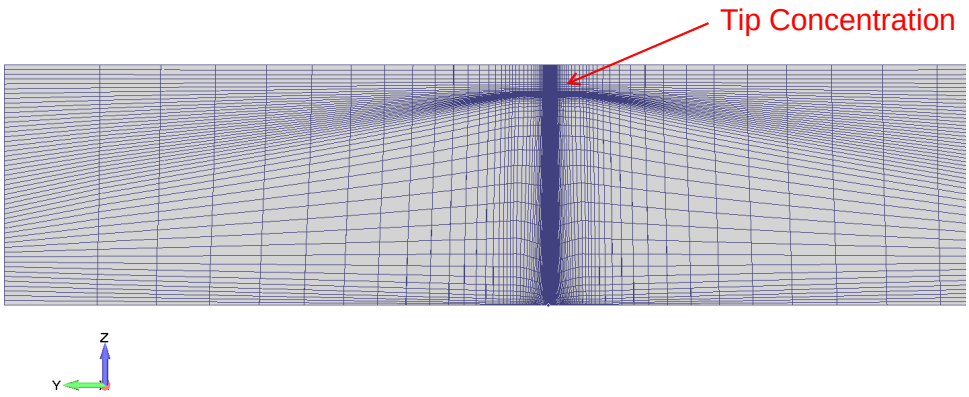

Fig. 5.5. Slice through mesh showing concentration of cells near foil tip

5.1.2 Solid Domain

Two techniques are used to generate structural meshes for this work. The first technique is applicable to simple models, like the beams described previously and in the next section, that can be extruded from planar elements. These meshes are created manually in the commercial software Femap [139]. The second technique, used for more complicated models of the fin and impeller, is a Matlab application written by the author called `bladeGen`. This application first uses Non-Uniform, Rational B-Splines (NURBS) [107] to parameterize a surface, and then applies a smooth mesh on the pressure and suction surfaces based on the NURBS, and finally extrudes elements through the thickness. The user is given two options for meshing at the leading and trailing edges; one option uses wedge elements and the second uses rotated hexahedral elements to produce a mesh consisting of only hexahedral elements. The latter approach, which is used for this work, results in an element cross-section distribution as shown in Figure 5.6. The program features a GUI (Graphical User Interface) front end, shown in Figure 5.7. As indicated by the GUI, the software allows the user to specify the number of elements in all three local directions (chord-wise, span-wise, and through-thickness) and allows biasing in the chord-wise and span-wise directions. Once satisfactory mesh distributions are achieved, the mesh resolution is changed by modifying the number of elements in any of the three dimensions. The user's previous settings are saved in the computer's registry (for the Microsoft Windows operation system) and become the software's default settings for ease in changing mesh resolution or other settings at a different time without needing to re-enter the values.

The software requires input of blade section information in the form of point locations sequentially ordered from the trailing edge along the suction surface to the leading edge and then back the pressure surface to the trailing edge for all sections (or slices of the fin), starting at the blade root and extending to the blade tip. A plot of the points required for the modified NACA 66 fin, obtained by slicing

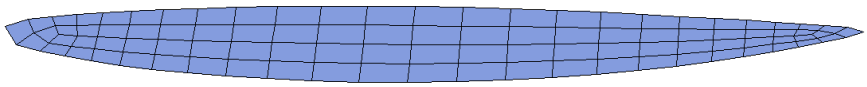

Fig. 5.6. Element through-thickness distribution for mesh with only hexahedral elements; pressure surface is on the top side of the mesh, suction surface is on the bottom side

the solid model at several span locations, is provided in Figure 5.8. Multiple fillet sections are used to ensure the mesh accurately represents the solid model in this region of high-curvature. Note that these sections are scaled based on the measured fin shrinkage described in Section 4.2. An example mesh of the fin, created using the `bladeGen` software, is shown in Figure 3.7.

5.2 Structural Solver

The objective of the structural solver is to accurately translate fluid forces acting on a structure to displacements of the structure. The ability of the structural solver developed for this effort to accomplish this objective is demonstrated in this section. Because the primary mode of deformation in the models studied here is bending, and because this usually causes the most difficulty due to *locking* effects for nearly-incompressible materials [23], comparisons are focused on bending loads. The first set of evaluations are for a slender beam model, followed by the simulations of the modified NACA 66 fin.

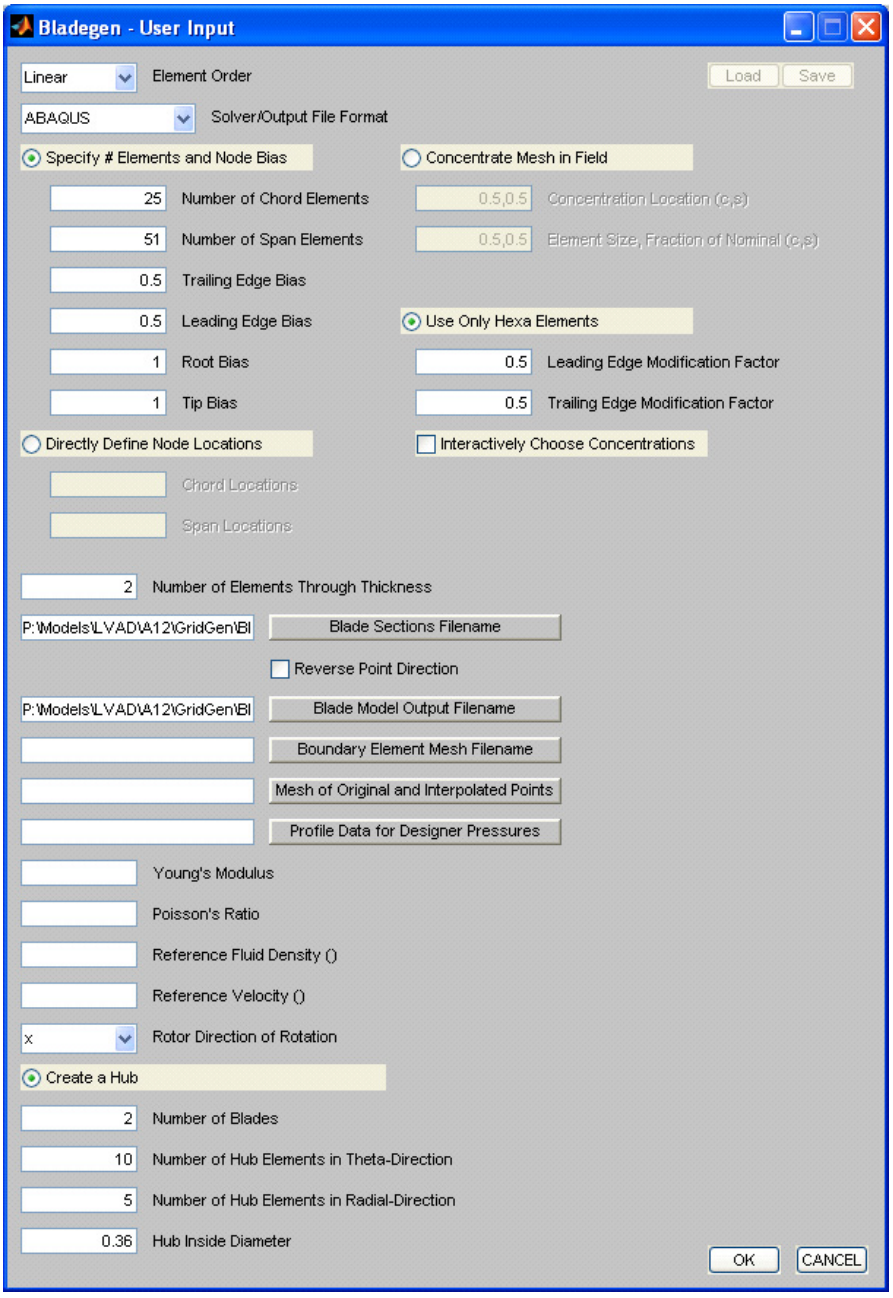

Fig. 5.7. Graphical user interface for `bladeGen` software

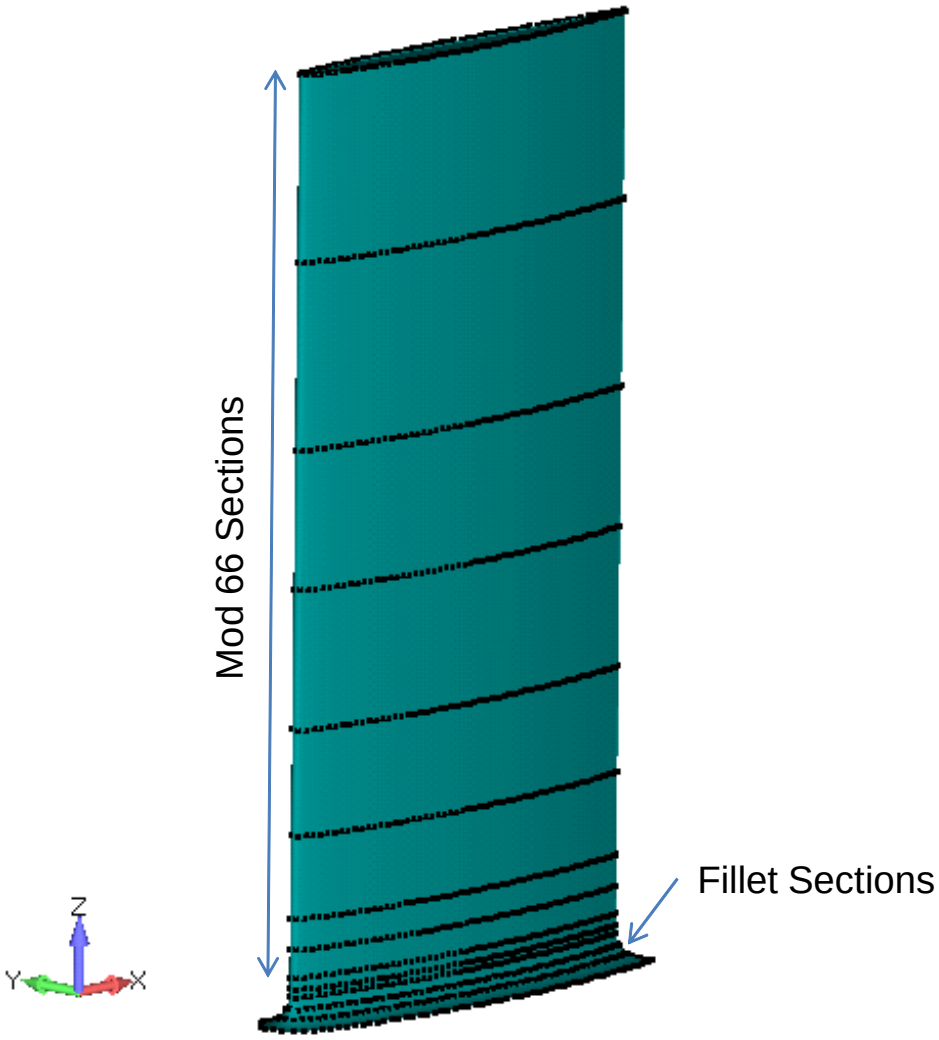

Fig. 5.8. Modified NACA 66 fin showing locations of section points for the FE mesh creation

5.2.1 Beam Bending Comparisons

The structural solver is validated by comparing simulation results from `feanl` to either analytical results for linear, small-deformation problems or simulation results from the commercial software Abaqus [57] for nonlinear, large-deformation problems. `Feanl` is validated first for small deformations and then for large deformations. While the solver does support load types other than concentrated forces (e.g., tractions and enforced displacements), comparisons are provided only for cases relevant to the modeling required herein.

The FE solver is first validated for the linear response of a fixed-free beam with a single, concentrated tip load. The analytical solution is derived by integrating over the beam length the beam bending moment relation ($\mathrm{d}^2y/\mathrm{d}x^2 = M/EI$) with the boundary conditions of $y(0) = 0$, $\mathrm{d}y/\mathrm{d}x\big|_{x=0} = 0$, and a tip load P at $x = L$:

$$y = \frac{Px^2}{6EI}\left(3L - x\right),$$

where x is the distance along the beam, y is the beam deflection, E is Young's modulus, I is the bending moment of inertia ($I = bh^3/12$ for the rectangular cross section used here), and L is the beam length. The beam dimensions are $b = 6.5$ mm, $h = 3.5$ mm, and $L = 125$ mm (Figure 3.10). A Young's modulus of 60 MPa is employed and for the finite element model a Poisson's ratio of 0.33 is applied. The applied load acts in the negative y-direction with a magnitude of 0.001 N. Three finite element models have been developed, each of different mesh resolutions. The coarsest mesh is shown in Figure 5.9. Comparisons of the FE-simulated beam deformations to the analytical results are provided in Figure 5.10. The results in this figure show that the FE model converges to the analytical solution with a maximum difference at the beam tip of 28% for the coarse mesh (100 elements), 4.6% for the medium mesh (1,600 elements), and 1.5% for the refined mesh (12,800 elements). These results are similar to those obtained from

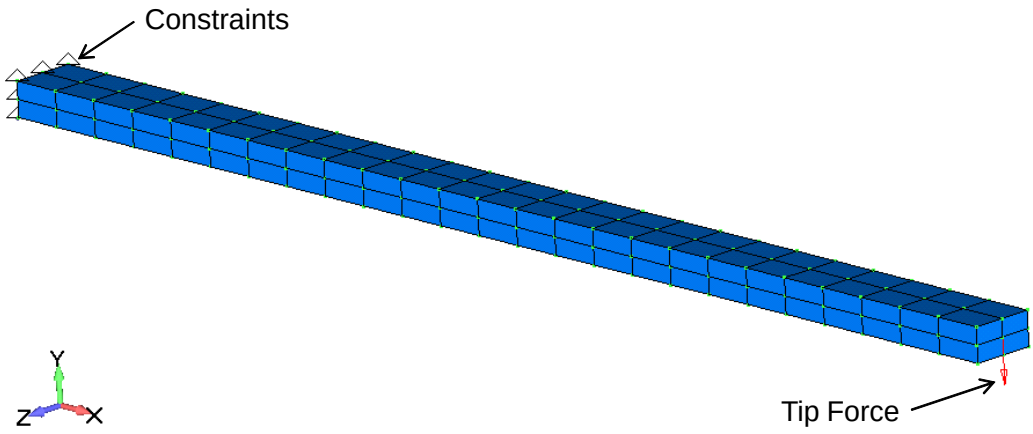

Fig. 5.9. Cantilevered beam coarse finite element mesh (100 elements) for validation

Abaqus: 26%, 3.9%, and 1.3%, respectively for the coarse, medium, and refined mesh models.

The structural solver is also validated against Abaqus for a cantilevered beam undergoing a large deformation. For this case, only a single mesh resolution is employed because two finite element models are being compared. This model uses the same dimensions used for the linear beam model, but the material properties are slightly different: a Young's modulus of 69. MPa and a Poisson's ratio of 0.49. Furthermore, the load magnitude has increased substantially to 2.2 N. The beam deformation as computed by `feanl` is shown in Figure 5.11 and a comparison between `feanl` and Abaqus of the beam shape along the beam centerline is shown in Figure 5.12. `Feanl` computes a beam tip displacement magnitude that is 0.31% less than the value computed by Abaqus, which is well within anticipated specimen-to-specimen variations of the hardware (see the beam deflection variations in Figure 3.14).

The `saveCurrentState` and `resetState` member functions described in Section 2.3.3 are required for tightly coupled solutions requiring sub-iterations.

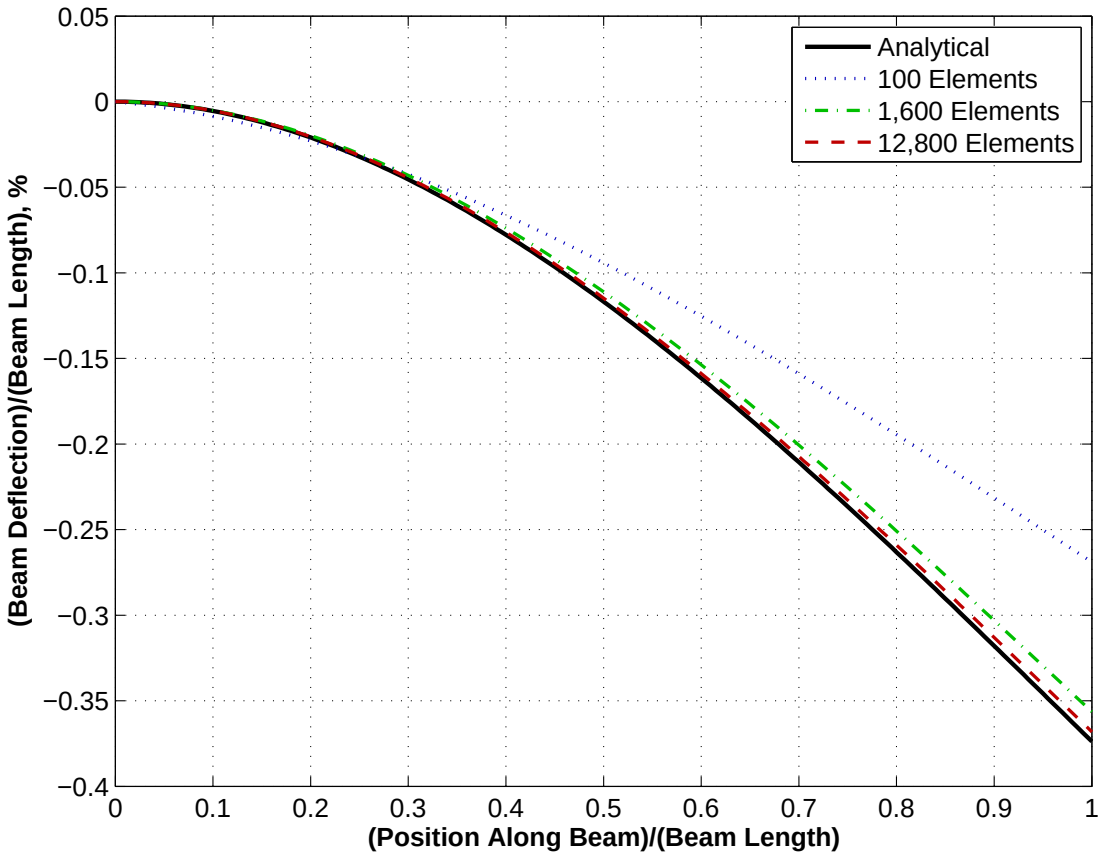

Fig. 5.10. Cantilevered beam linear model validation results showing finite element beam deformation compared to analytical results; the finite element results are the centerline nodal displacements in the y-direction (the load is applied in the negative y-direction, see Figure 5.9)

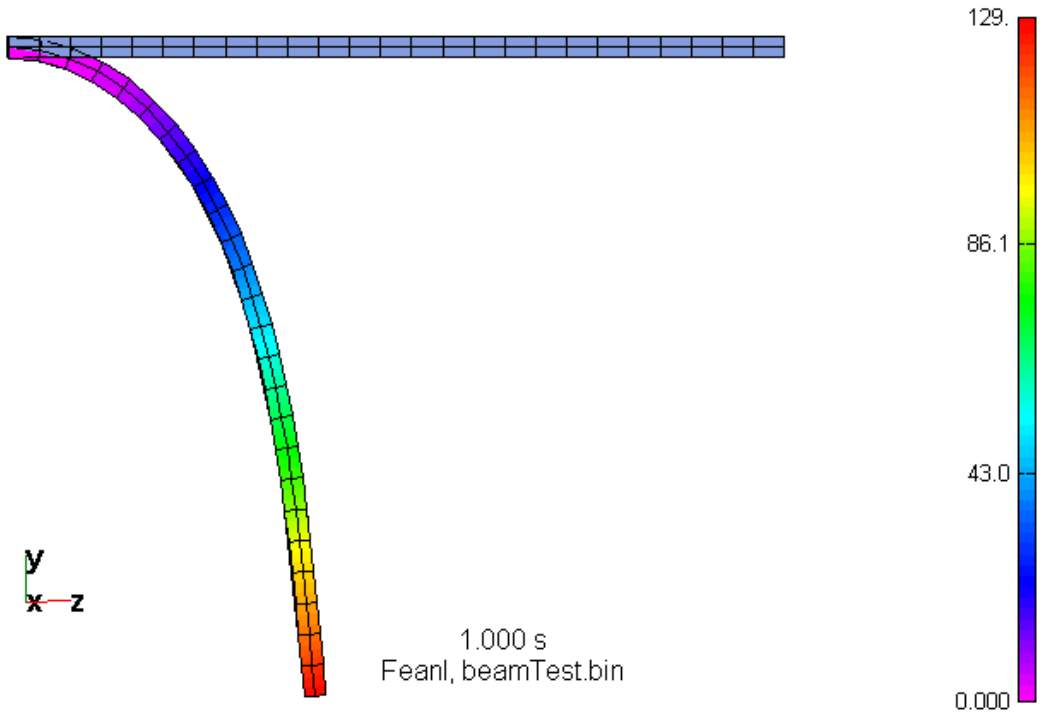

Fig. 5.11. Cantilevered beam nonlinear (large deformation) model deformation computed by `feanl`; the light blue elements represent the undeformed shape; the deformed shape is contoured by displacement magnitude in mm

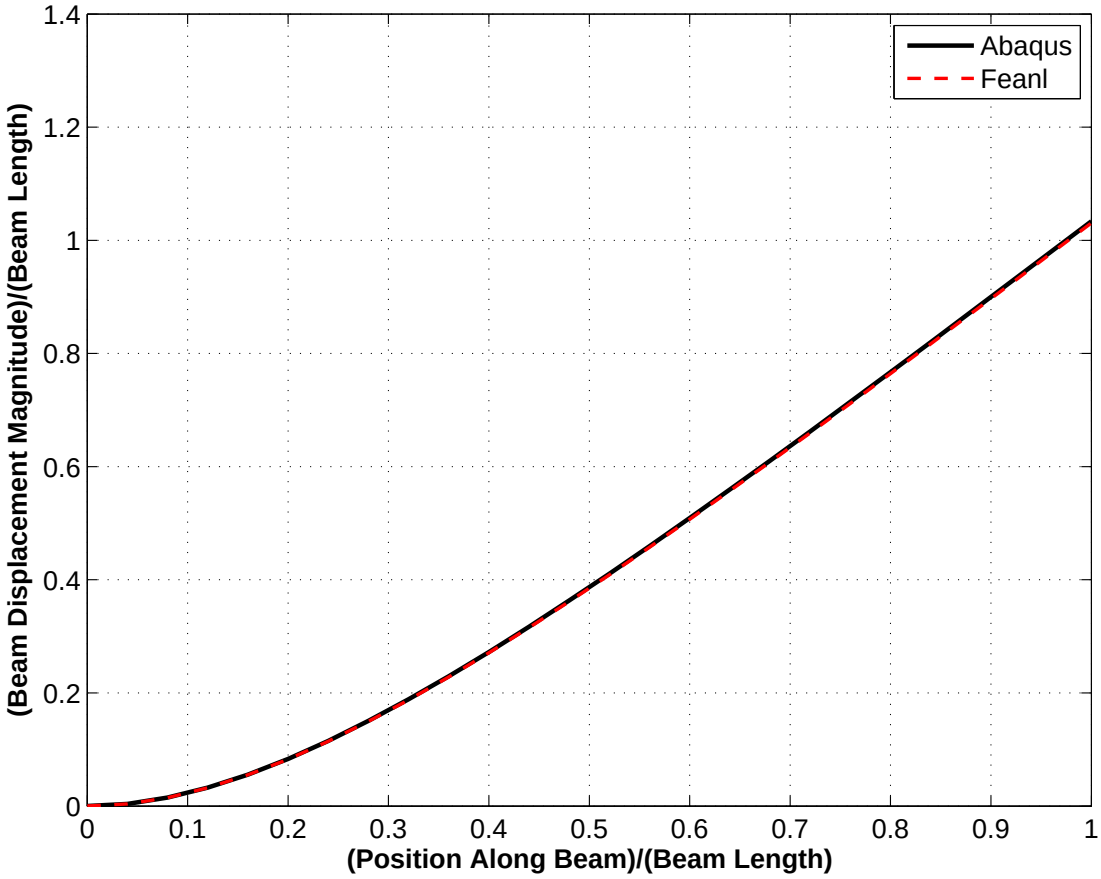

Fig. 5.12. Comparison of beam centerline vertical displacements computed by `feanl` and Abaqus for the cantilevered beam subjected to a large deformation

These functions were validated using the fixed-free beam of Figure 5.9 comprised of a viscoelastic material and a concentrated load near the beam's tip. The beam deformation was simulated to 2 seconds, the current state was saved, the simulation continued to 5 seconds where `resetState` was executed to reset the solver to the state at 2 s and then the simulation continued until the end time of 100 s. A plot of the tip deflection versus time, provided in Figure 5.13, demonstrates that the continuous and restarted results are identical, establishing the validity of this solver operation.

5.2.2 Modified NACA 66 Fin

The ability of `feanl` to accurately model the modified NACA 66 fin has been demonstrated previously in Section 3.5.2, where a mesh convergence study was reported and comparisons made to empirical results. The purpose of the present section is to describe additional modeling capability of `feanl`.

FSI simulations were originally performed using the refined x 2 mesh of the modified NACA 66 fin described above in Section 3.5.2. However, the structural solution times were prohibitive and thus two alternative approaches were pursued. The first was a parallelized solver using PetSc. While this solver showed improvement in system matrix assembly times because each processor creates matrix entries for its assigned elements, the default iterative solver converged very slowly if at all for these problems. Alternative sparse, distributed memory direct solvers, MUMPS [3] and SuperLU [85], were tried but without substantial solution time improvements.

In parallel with this development, a second approach was explored that involved alternative element formulations. Considered in this approach were pressure/displacement (i.e., hybrid) elements, selectively reduced-order elements, and

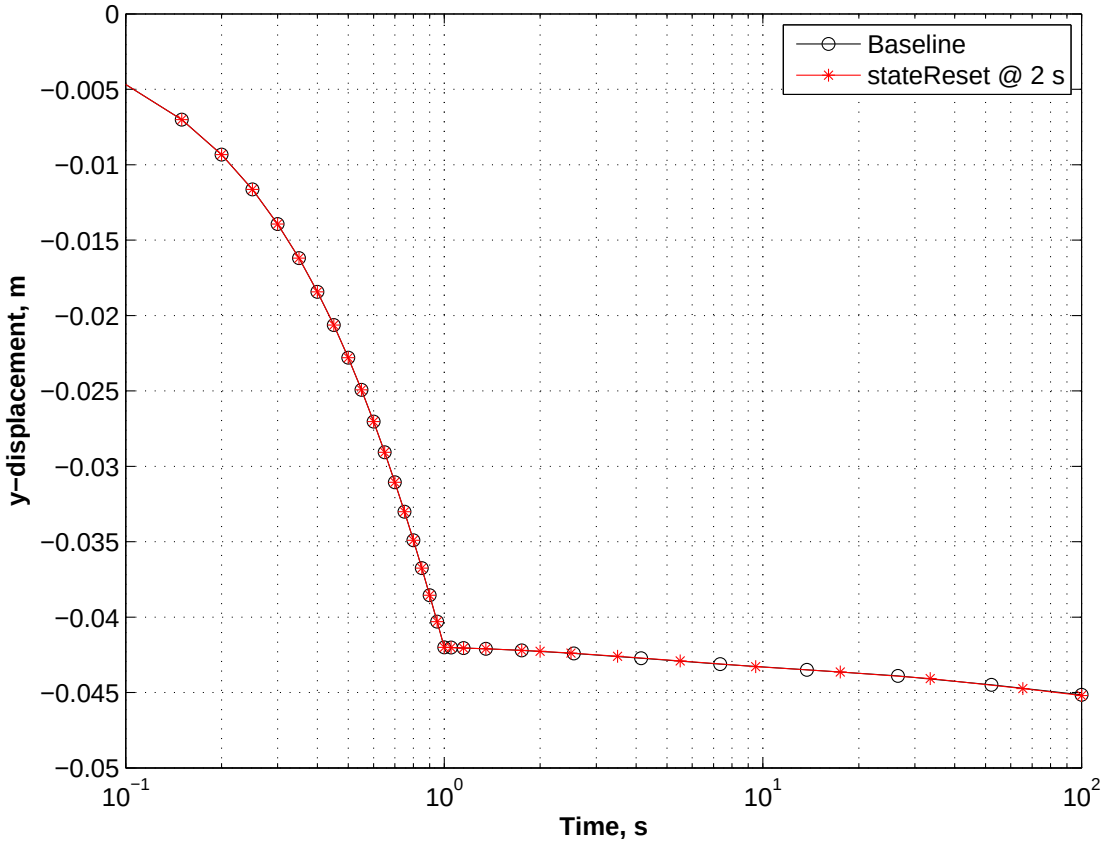

Fig. 5.13. Test beam model results demonstrating functionality of the `saveCurrentState` and `resetState` member functions of `Structure`

quadratic elements. Hybrid elements are designed specifically for nearly-incompressible materials and thus perform well. However, they require a substantial reformulation of the structural solver. It was then discovered that selectively reduced-order integration elements, as described by Hughes [62], perform equivalently to the hybrid elements. Quadratic elements were also evaluated because, relative to linear elements, they are not as susceptible to shear locking.

A comparison of the solution convergence of the coarse mesh model using selectively reduced integration elements and the highly-refined (refined x 2) mesh model of the fin is shown in Figure 5.14. The agreement is excellent with only a 0.06% difference between the tip displacement of the Hapflex 598 fin at 3600 s. Also shown in this figure is the tip-displacement for the quadratic-element model, which is within 0.7% of the refined x 2 mesh results. Given the good agreement between the models and having already implemented the FSI interface coupling algorithm for linear elements, the selectively reduced-order elements were chosen for all subsequent models over the quadratic elements.

The motivation for this investigation was to replace the somewhat large finite element model with a model that can be solved quickly based on long simulation times that occurred during the initial FSI simulations The original comparisons reported here were completed prior to implementing the reduced integration and quadratic element support in `feanl`, and therefore the comparisons to experimental results presented previously employ the refined x 2 mesh.

It should be noted also that similar results to the Hapflex beam models were obtained when comparing pressure/displacement hybrid and reduced-integration elements using Abaqus.

The difference between fin deformations for the large- and small-deformation FE formulation is shown for the fin subjected to the test tip load in Figure 5.15. It is apparent from the results in this figure that a large-deformation formulation is required for these analyses.

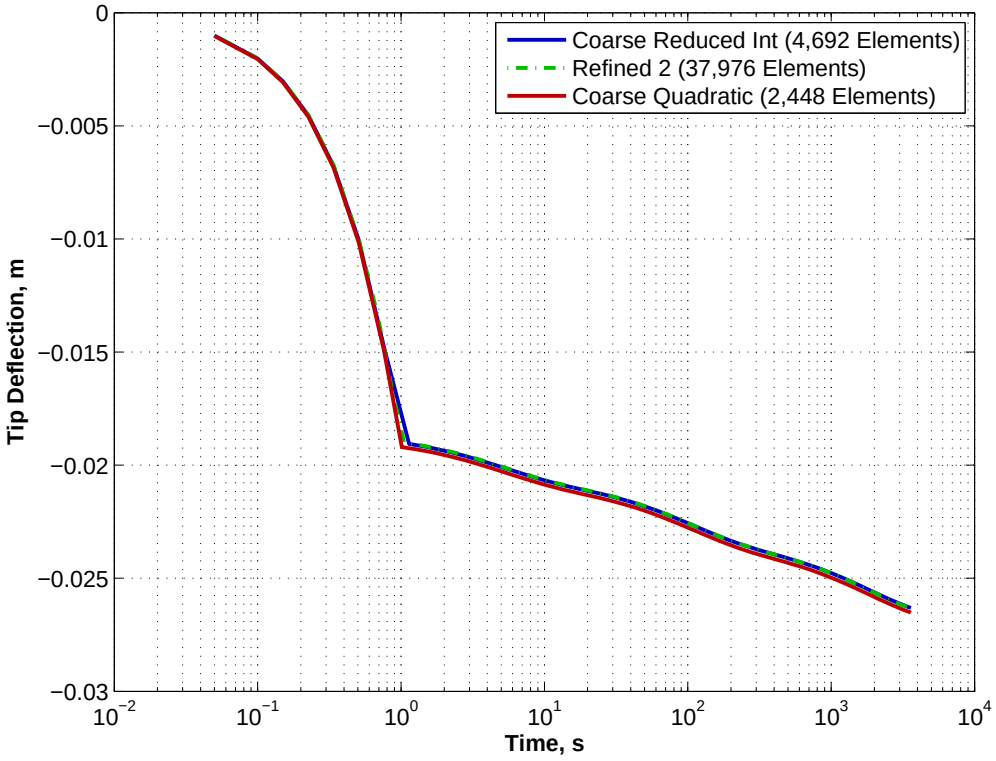

Fig. 5.14. Convergence of finite element fin model using preliminary Hapflex 598 material model

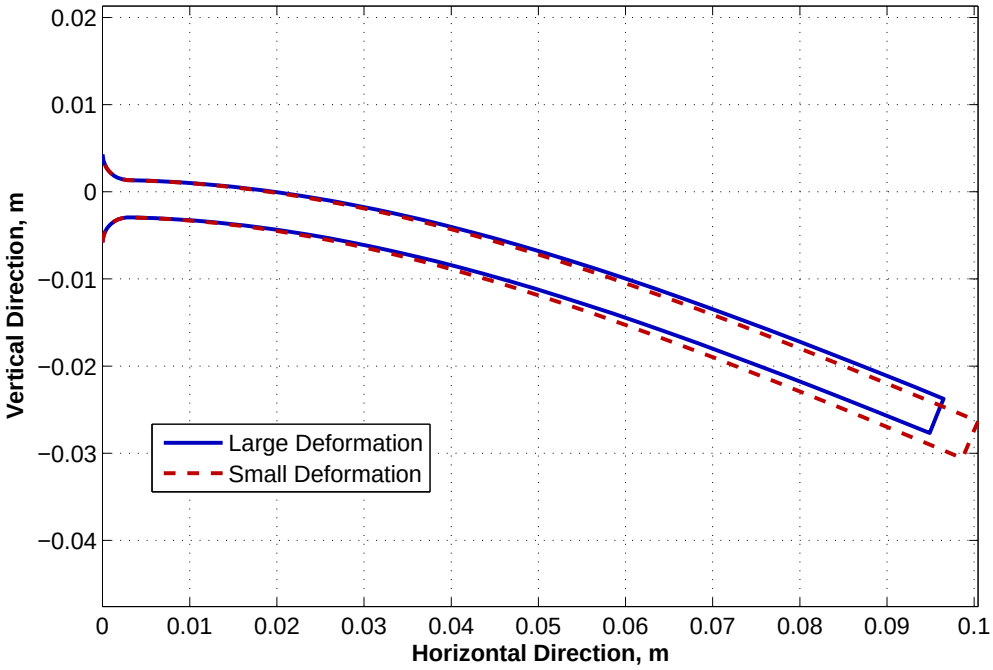

Fig. 5.15. Comparison of finite element results for large- and small-displacement formulations; outlines of deformed fin along the mid-span of the coarse fin model are shown

The maximum equivalent strain computed for the fin is approximately 1.47% as shown in Figure 5.16. This is slightly larger than the maximum strain observed for the Hapflex beams, but not substantially larger to invalidate the use of a linear material constitutive relationship based on the good agreement between the numerical and experimental results presented previously.

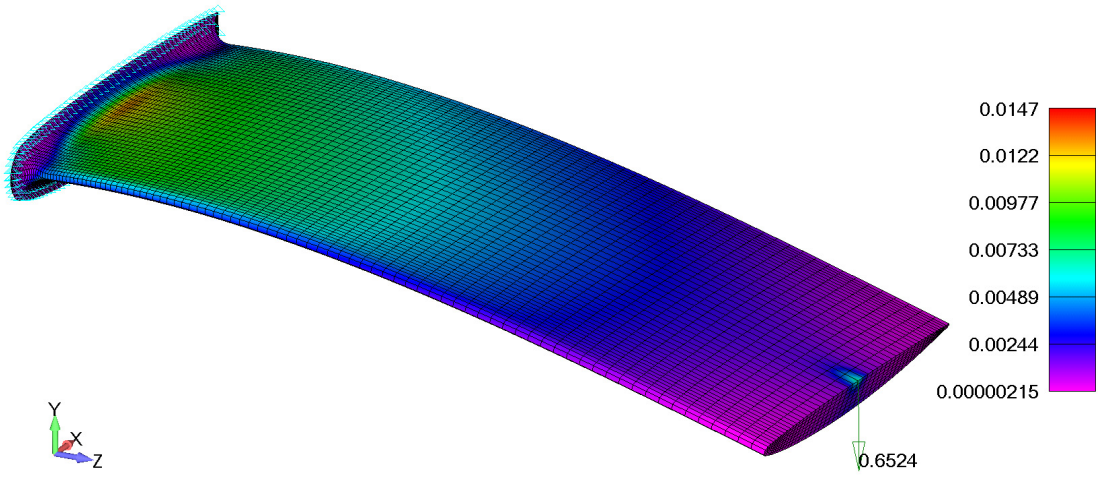

Fig. 5.16. Deformed fin after 1 hr of relaxation, contoured by equivalent strain

5.2.3 Solver Performance

The performance of the structural solver is compared to that of the commercial software Abaqus [57]. Two aspects are important in this comparison, which is focused on the large-deformation formulation. The first is the number of Newton-Raphson iterations required for a given solution increment. This evaluates the accuracy and completeness of the stiffness tangent of Equation 2.25. The second is the time required per Newton-Raphson iteration. This comparison contrasts time required to assemble the system matrix and residual vector and solve for the next solution guess.

The test problem used to compare the solvers is the refined x 2 hydrofoil model (see Section 3.5.2). This is the most germaine to the present work because it features large deformations and viscoelasticity. Both solvers were executed using an Intel dual quad core 2.39 GHz CPU having 8 GB RAM and running the Windows XP operating system. `Feanl` used approximately 160 MB RAM while Abaqus used approximately 130 MB, which is most likely due to different formulations of the problem (Total versus Updated Lagrangian). `Feanl` required 82 solution increments while Abaqus required 67 increments. This difference is due to the use of different time step sizes, which are determined by an automatic time stepping algorithm in each solver. For instance, `feanl` doubles the time-step size if three or less Newton-Raphson iterations are required for a solution step. Abaqus uses a more complex rule, which is not described here.

The number of Newton-Raphson iterations per solution increment are plotted in Figure 5.17 versus the solution increment. This figure shows that Abaqus generally requires far fewer time steps than `feanl` for a given solution increment. This is due to different convergence constraints, different residual tangents, or a combination of these. `Feanl` should be further evaluated in the future to improve the convergence characteristics. Comparisons of wall-clock times on a per-iteration basis, shows the solvers to be comparable. Abaqus requires approximately 2.93 s per iteration and `feanl` requires approximately 1.55 s. However, `feanl` automatically uses multiple processor threads when available for the linear equation solves and Abaqus does not. For this test case, `feanl` used two threads and Abaqus used only one. The two largest consumers of clock time for `feanl` are the matrix assembly and linear equation solution. For this test problem, `feanl` spent approximately 54% of the time assembling the system matrix and 42% of the time solving the system of equations. Similar information is not available for Abaqus. The results indicate `feanl` provides reasonable performance with respect to similar commercial software.

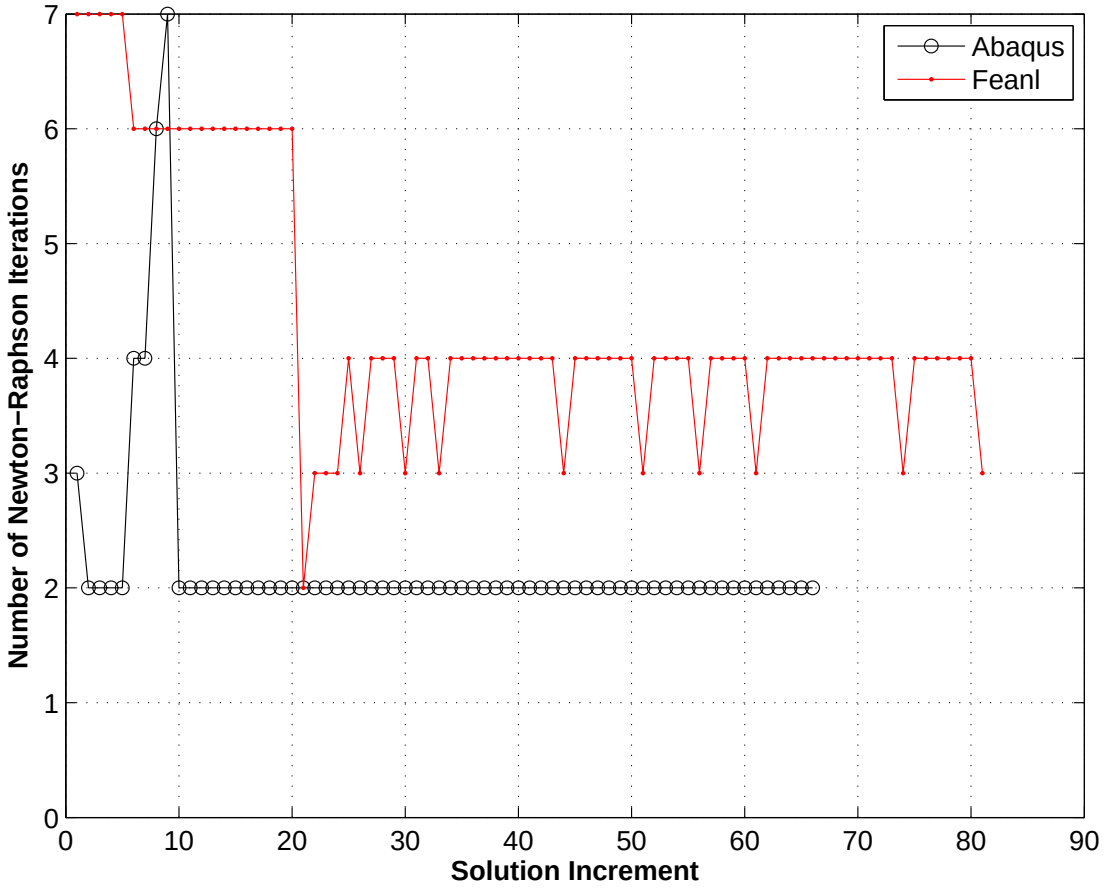

Fig. 5.17. Comparison of number of Newton-Raphson iterations per solution increment for `feanl` and Abaqus

5.3 Structural Inverse Solver

The inverse structural solver is initially validated using a test case similar to that employed by Fachinotti et al. [36]. Validation of the solver is also shown for each inverse analysis employed in this work within the relevant sections of this thesis. The test case of [36] is a two-dimensional cantilevered beam subjected to a tip force as shown in Figure 5.18. The case used here differs from that in [36] in that a linear-elastic material is employed here whereas their material was orthotropic. The inverted structural shape is shown in Figure 5.19. Application of the tip force to the inverted shape should cause the inverted shape to deform into the original shape if the inverse solver is functioning properly. Figure 5.20 shows the final shape of the inverted structure deformed by the tip force, colored by the displacement error in meters. The maximum node location error is 0.09%, thereby validating the inverse finite element solver for this large-deformation, linear elastic case. Note that the inverse solver implementation makes no distinction between constitutive materials and thus the solver should also function properly for the viscoelastic materials considered here. However, comparisons are performed elsewhere to ensure proper inverse shapes are used in the FSI simulations.

5.4 Flow Solver

The OpenFOAM `simpleFoam` flow solver is verified against the commercial software Fluent [46]. The test case for this comparison is the modified NACA 66 fin in the water tunnel. The model consists of uniform axial velocity of magnitude 1.2 m/s at the inlet, fixed pressure at the outlet, incompressible, laminar flow, and the rigid fin was set at zero angle of attack. Figure 5.21 shows the boundary conditions for this model. Contours of static pressure are compared in Figures 5.22 and 5.23 and the integrated stresses on the structure's surfaces are provided in

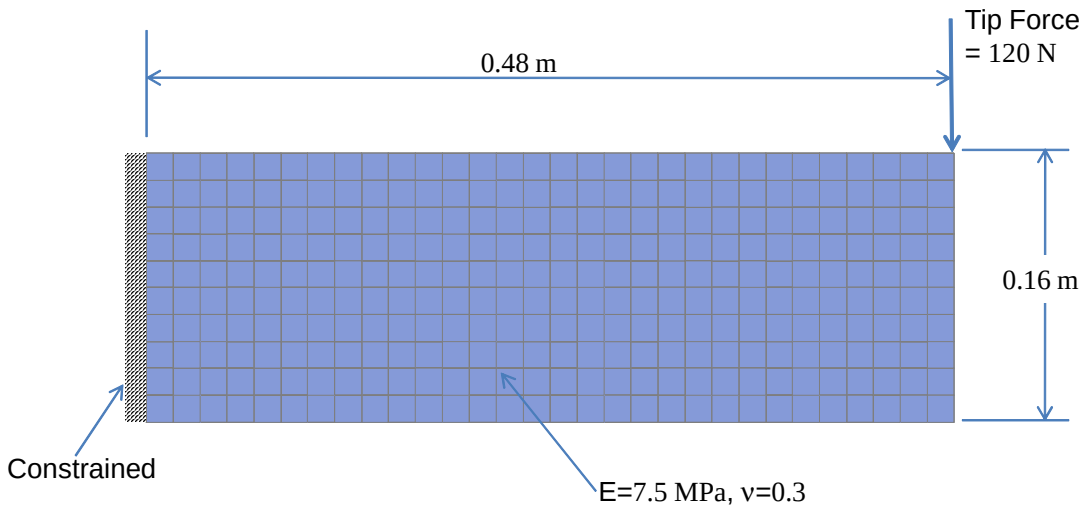

Fig. 5.18. Test case for structural inverse solver showing constraints, tip force, and the structure's dimensions

Table 5.1. These comparisons provide confirmation that the OpenFOAM solver is functioning properly.

A mesh resolution study for the flow solver has also been conducted. Three meshes have been used, a coarse mesh with 45,200 cells, a medium mesh with 256,000 cells (see Figures 5.1 through 5.5), and a refined mesh with 664,000 cells. The static pressure contours on the fin surface for the refined mesh are shown in Figure 5.24 and the net fin forces are compared to the medium-mesh fin forces in Table 5.2. With the exception of the force component in the z-direction, the medium mesh agrees within 1% of the refined mesh. Because of the minor effect the z-direction loads have on this application, and the computational cost of the refined mesh over the medium mesh, the medium mesh is used throughout the work reported here. The ultimate confirmation of model accuracy comes from comparisons of the FSI solver with experimental results, which is described below after a brief discussion of the mesh motion solver.

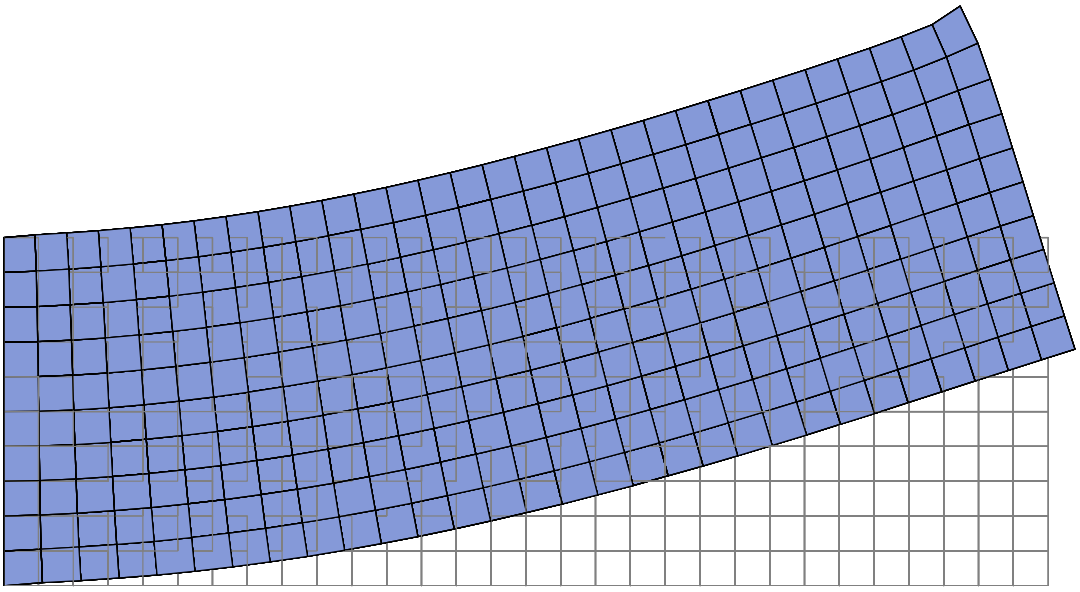

Fig. 5.19. Structural inverse solver test case showing the inverted shape (shaded elements) and the original shape (transparent elements)

Table 5.1. Structural forces (pressure + viscous) for solver validation study

Solver	F_x	F_y	F_z
Fluent	0.0927 N	-0.305 N	0.0321 N
OpenFOAM	0.0915 N	-0.309 N	0.0317 N
Difference	-1.27%	1.14%	-1.26%

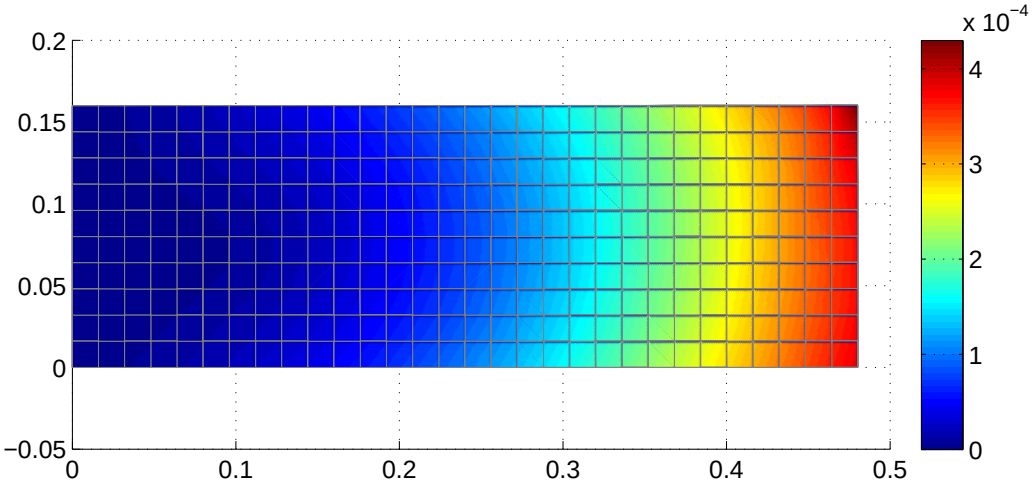

Fig. 5.20. Inverted model deformed into the original shape from application of the tip force; contoured by location error in meters with respect to the original shape

Table 5.2. Structural forces (pressure + viscous) for solver validation study

Number of Cells	F_x	F_y	F_z	$\lvert\boldsymbol{F}\rvert$	Difference
45,200	0.108 N	-0.337 N	0.0361 N	0.356 N	10.1%
250,000	0.0915 N	-0.309 N	0.0317 N	0.323 N	0.771%
664,000	0.0908 N	-0.306 N	0.0298 N	0.321 N	

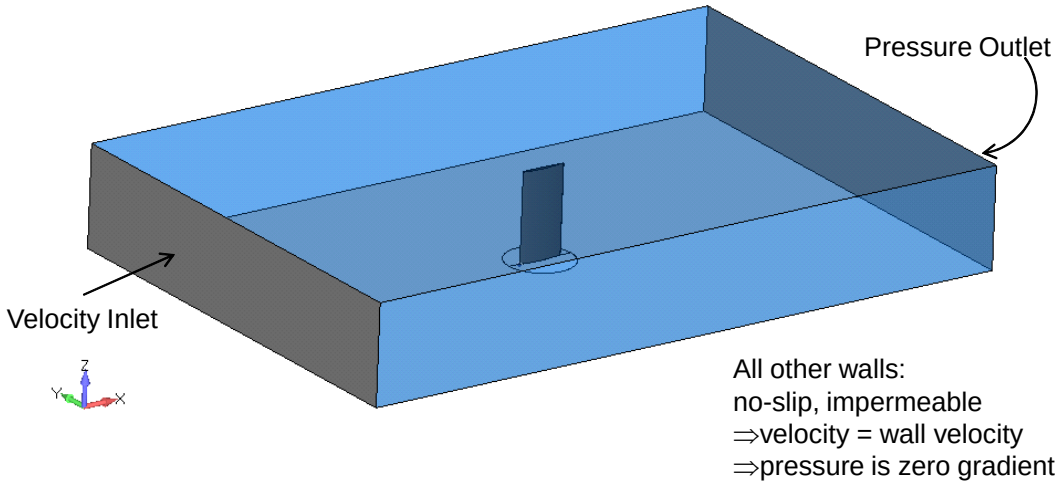

Fig. 5.21. Boundary conditions for the water tunnel simulations; all walls are stationary except the fin surfaces

5.5 Fluid Mesh Motion Solver

The purpose of the mesh motion solver is to track the fluid/solid interface and deform the fluid mesh to accommodate structure deformations while maintaining sufficient mesh quality. The validity of the mesh motion solver is therefore evaluated by ensuring that 1) the fluid/solid boundary from the perspective of the fluid mesh matches the fluid/solid boundary from the perspective of the structural mesh, and 2) the fluid mesh has acceptable quality. All mesh motion schemes employed in this work rely on a motion specification of $\Gamma_{F/S}$, which occurs by displacement interpolations within the structural solver to reference element locations corresponding to each fluid boundary vertex. Run-time checks are performed during each simulation to ensure all fluid motion vertices are correctly mapped to the structural elements and that the structure solver accurately locates each fluid motion vertex in its owner structural element, both of which were previously described. A failure in either regard is considered a fatal error.

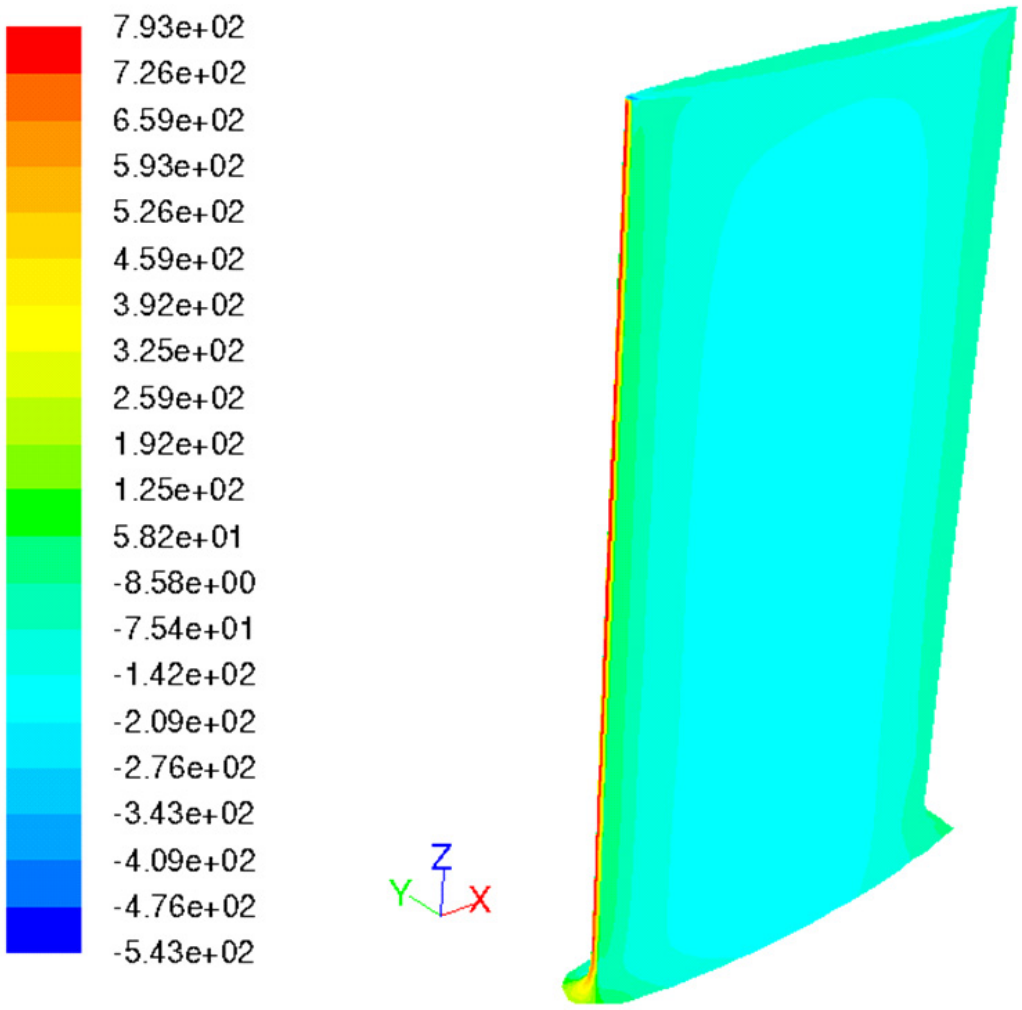

Fig. 5.22. Contours of static pressure (Pa) on fin surface as computed by Fluent at inlet flow speed of 1.2 m/s

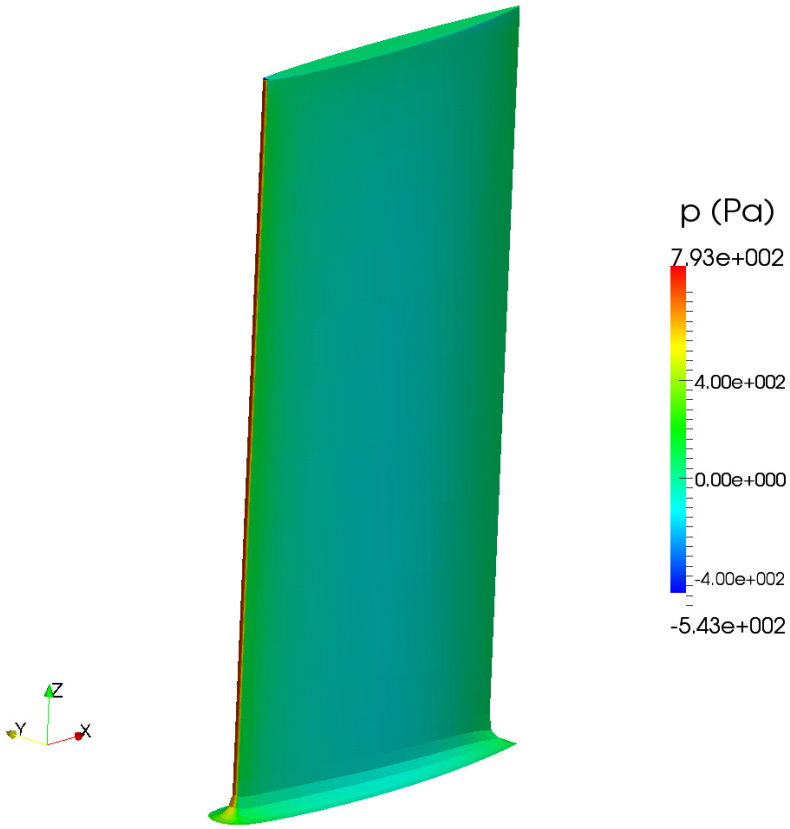

Fig. 5.23. Contours of static pressure (Pa) on fin surface as computed by OpenFOAM at inlet flow speed of 1.2 m/s using the medium-refinement mesh

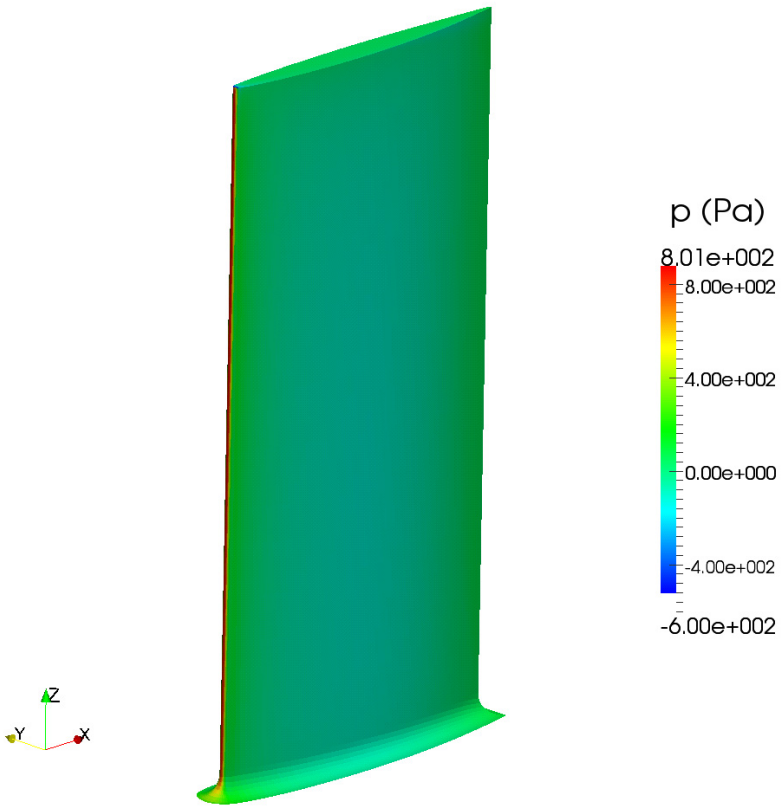

Fig. 5.24. Contours of static pressure (Pa) on fin surface as computed by OpenFOAM at inlet flow speed of 1.2 m/s using the refined mesh

The second criterion for the mesh motion solver is mesh quality, which not only concerns cell quality but loosely includes boundary conformity. The mesh's cell quality is checked after completion of each simulation using OpenFOAM's `checkMesh` utility, which checks things such as aspect ratios, volumes, non-orthogonality, and skewness [69]. The approach employed in this work is to ensure the mesh quality does not substantially degrade from the undeformed mesh quality. The boundaries are also reviewed to ensure the solver functions as planned and that gross errors like that shown in Figure 2.5 have not occurred.

In general, the mesh motion solvers employed here, with the exception of the Laplace tetrahedral decomposition solver, cause fatal run-time errors if they are not working properly. However, motion of the fluid mesh remains the most non-robust aspect of the three-field FSI solver implemented in this work. Further work is required in this area for future endeavors.

5.6 Fluid–Structure Interaction Solver

This section discusses the validation of the FSI solver for quasi-steady problems using the viscoelastic, modified NACA 66 fin subject to incompressible, laminar flow. The approach is to simulate the water tunnel test described in Chapter 4 and compare results for time-dependent fin deformation. The measured fin deflections have been quantified at the fin tips, and video images looking down from above the test section are available for qualitative comparisons. The simulations employ the structured mesh of the flow field described in Section 5.1.1 and the coarse structural mesh of Section 5.1.2 with selectively-reduced integration linear hexahedral elements. Motion of the flow mesh was accomplished initially using OpenFOAM's Laplace face decomposition solver and later transitioned to the custom RMF to enable the problem to be solved in parallel (to dramatically reduce the solution wall-clock times).

Simulations for multiple angles of attack were required because of the inability to accurately quantify the AOA during the water tunnel testing. Simulations for AOA's ranging from $+0.5$ to -2.5° were performed and then compared against the baseline AOA test results. Satisfactory agreement between the simulation and test results for the single AOA is sufficient to confirm validation of the solver because the temporal degree of freedom must also match. However, for additional confirmation comparisons are made for a second AOA. Recall, the absolute AOA for the water tunnel tests could not be defined, but changes in the AOA were very accurately and precisely controlled using a long lever arm and shim stock as previously described. Comparisons are reported here for the baseline AOA and the baseline AOA -0.25°. (The method used to modify the fin AOA for the FSI simulations is discussed in Section 2.5.1.)

The boundary conditions for the water tunnel test simulations include a prescribed velocity inlet, prescribed pressure at the outlet, and no-slip conditions at all surfaces as shown in Figure 5.21.

The inlet velocity profile is prescribed to be uniform and solely in the x-direction, with a time-varying magnitude. The water tunnel's test section velocity varied with time during startup from 0 to nominally 1.22 m/s in about 100 s. This temporal variation of U_x was quantified during some early tests and is used throughout all tunnel simulations reported here. A plot of the flow speed versus time is shown in Figure 5.25. This closely represents the experimental flow speeds as shown in Figure 5.26.

Comparisons of the leading and trailing edge tip deflections are provided next for the baseline AOA and baseline AOA -0.25° in Figures 5.27 and 5.28, respectively. These figures show good agreement between the measured and simulated blade tip deflections. Note that the ordinate scaling between these plots is consistent to facilitate amplitude comparisons between the different AOA's. The slope of the curves, caused by changes in load magnitude and fin stiffness as the

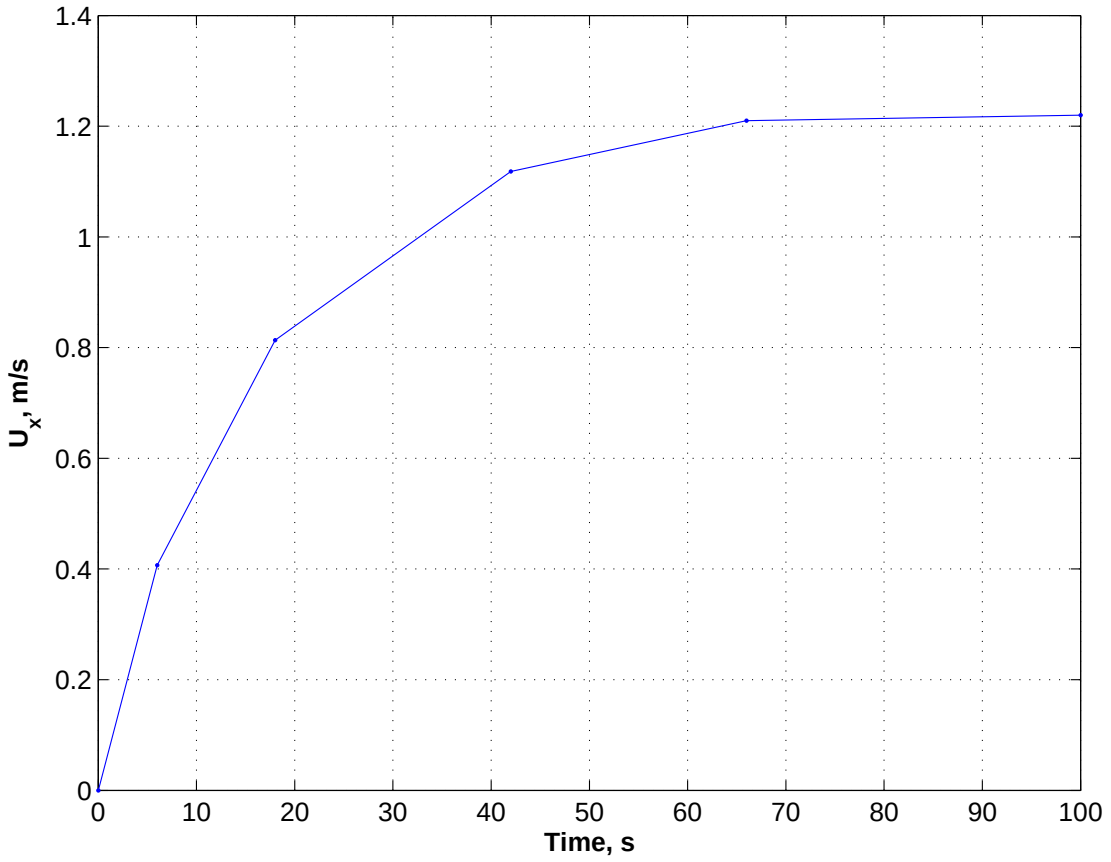

Fig. 5.25. The x-component of the inlet boundary's velocity, U_x, as a function of simulation time; $U_x = 1.22$ m/s for all times greater than 100 s

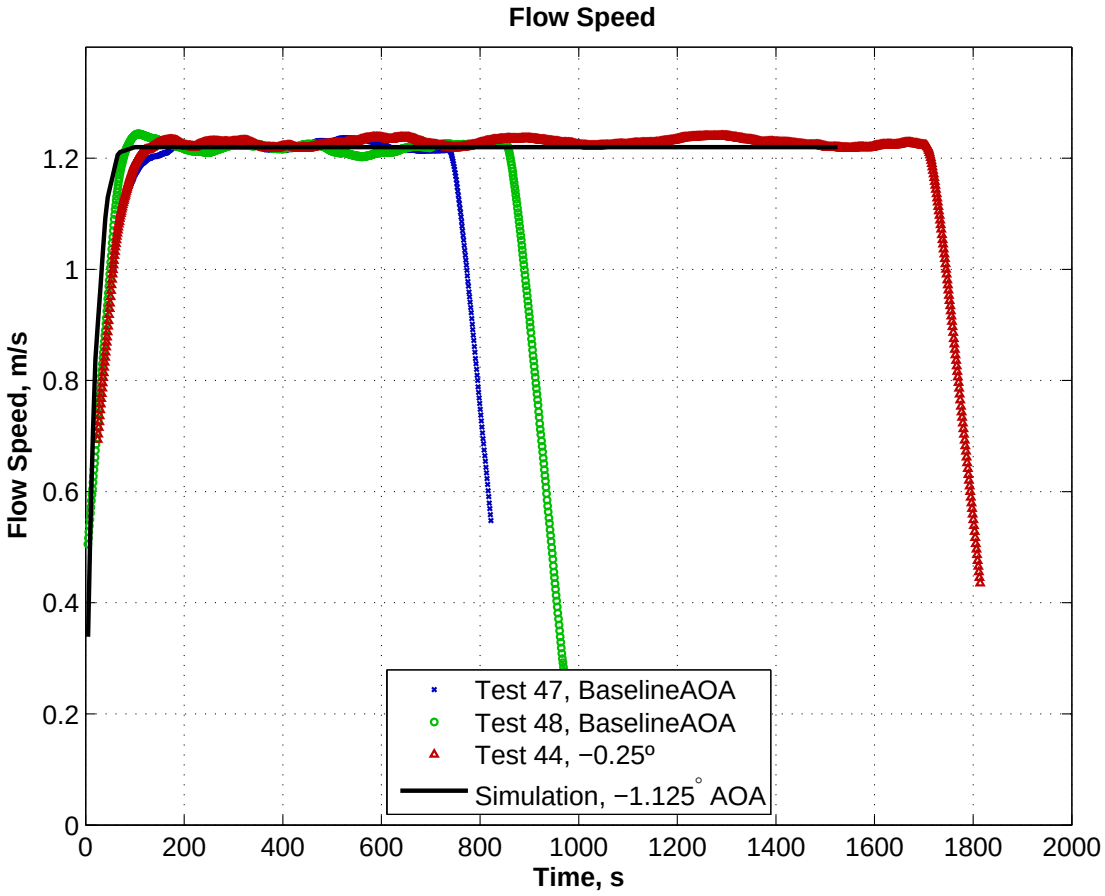

Fig. 5.26. Flow speed comparison for the tunnel tests and simulations

fin relaxes and changes the span-wise AOA, match well, as do the curve amplitudes. The fin deflection results are also presented in terms of pitch and heave in Figures 5.29 and 5.30, respectively. These figures show the same information as in Figures 5.27 and 5.28 but facilitate the interpretation of the results. As reported previously, noise in the load cell measurements precluded their use here. However, it is instructive to plot the fin lift (i.e., y-direction force) for the simulated results. These results, shown in Figure 5.31 for both AOA's, indicate the fin lift decreases with time as the fin deforms.

Images of the deforming fin from Test 44 at various times during the experiment and the corresponding FSI simulated results are shown in Figures 5.32 and 5.33. These images correspond to the tip deflection plots for the simulated -0.375° AOA in Figures 5.27 and 5.28. The images show good qualitative agreement of the simulation to the measured response.

The maximum strain levels in the simulated fin results are less than 0.7% as shown in Figure 5.34. These results are for the -1.375° AOA case, which has larger deformations than the -1.125° AOA case and therefore higher strain. The use of the linear-elastic material model as derived herein is therefore acceptable for these simulations.

5.6.1 Solver Performance

The performance of the FSI solver is characterized based on time required to solve each of the three fields: fluid, solid, and mesh motion. Relative comparisons of the different mesh motion techniques are then made. The modified NACA 66 fin having a -1.125° AOA and the medium-density fluid mesh is used for the timing study presented here. Figures 5.35 and 5.36 show the elapsed wall-clock times per solution time increment for the Laplace face decomposition and the custom RMF mesh motion solvers, respectively. Results were generated on a Linux cluster having 256 2.8 GHz 64-bit Intel processors with 2 GB RAM per processor. It is

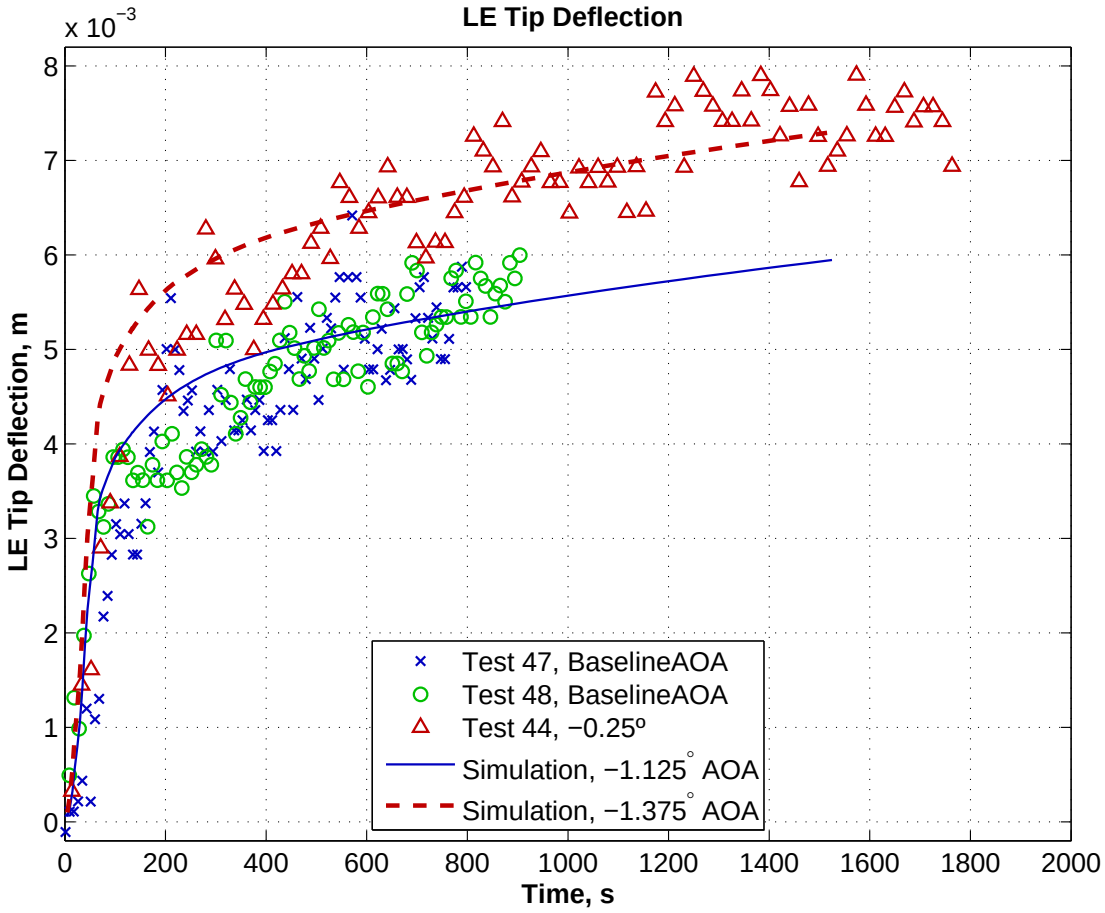

Fig. 5.27. Simulated and measured leading edge tip deflection

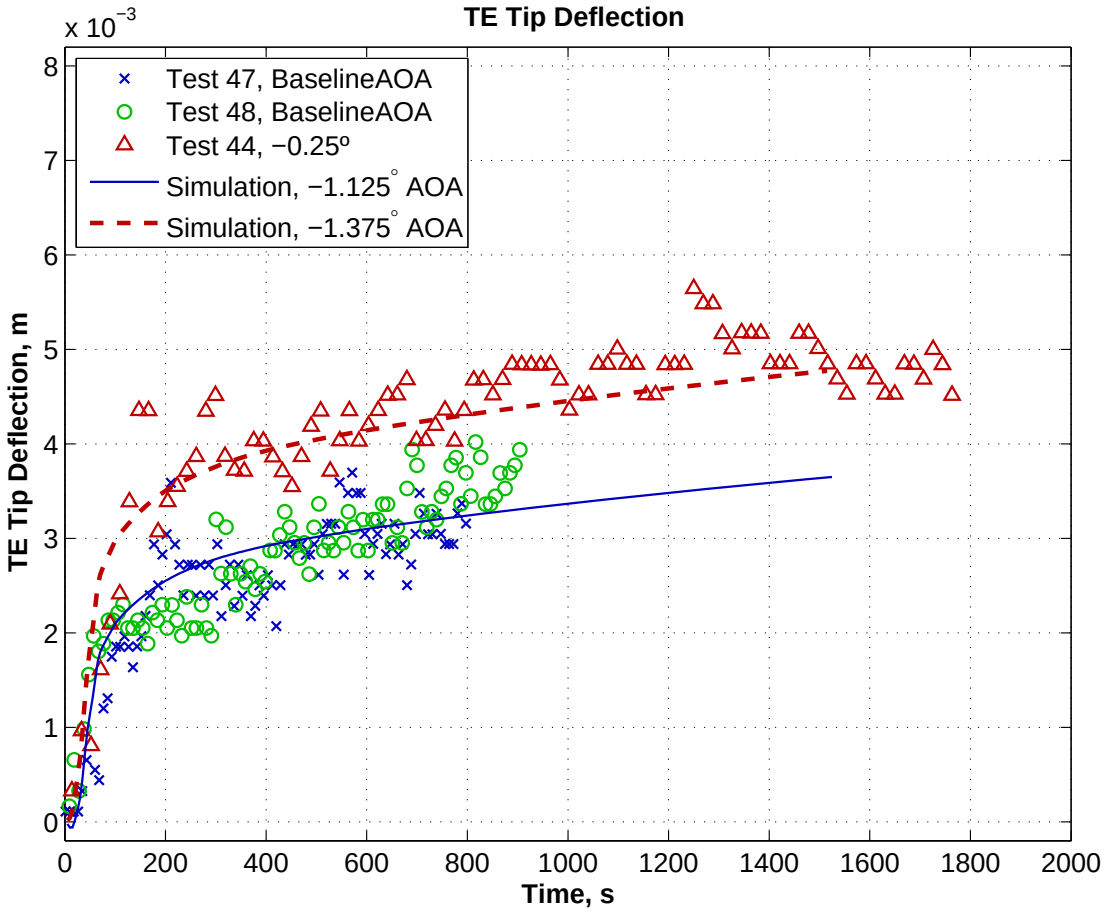

Fig. 5.28. Simulated and measured trailing edge tip deflection

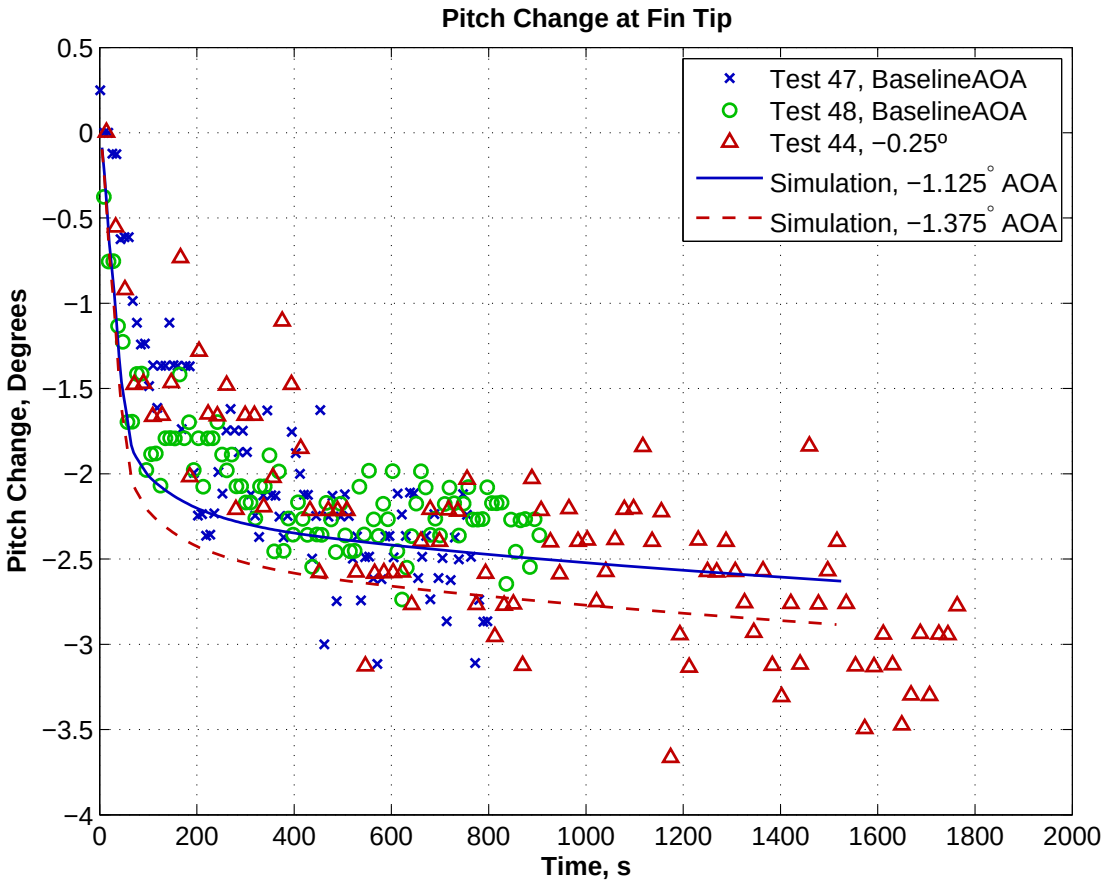

Fig. 5.29. Simulated and measured pitch angle change at the fin tip

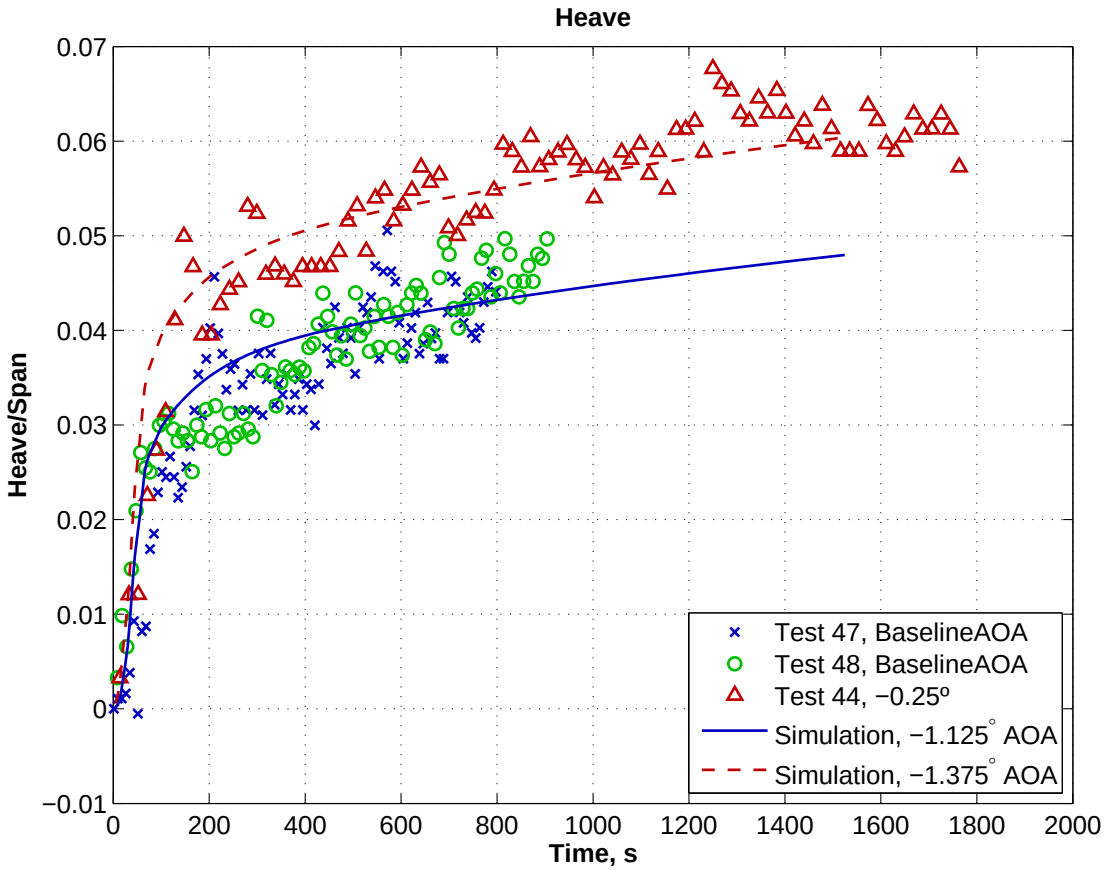

Fig. 5.30. Simulated and measured heave at the fin tip, normalized by the fin span

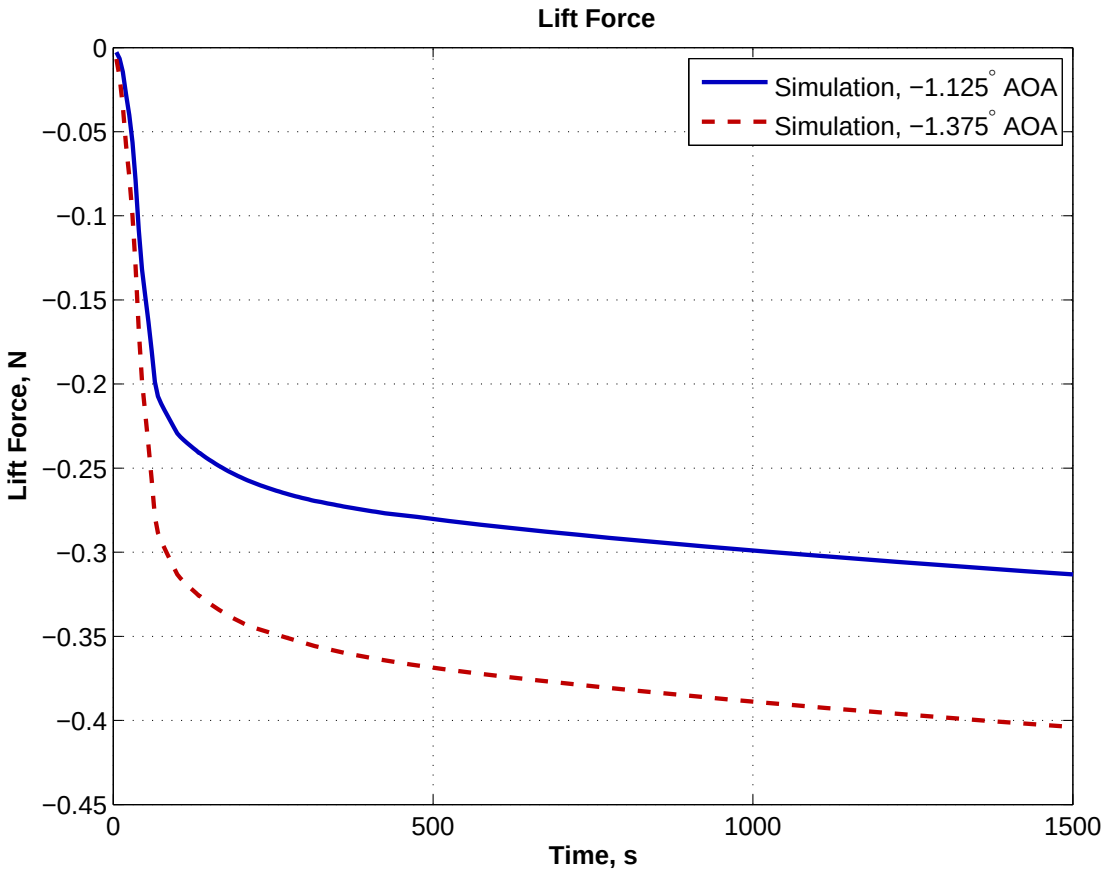

Fig. 5.31. Simulated lift force corresponding to the baseline and baseline -0.25° AOA

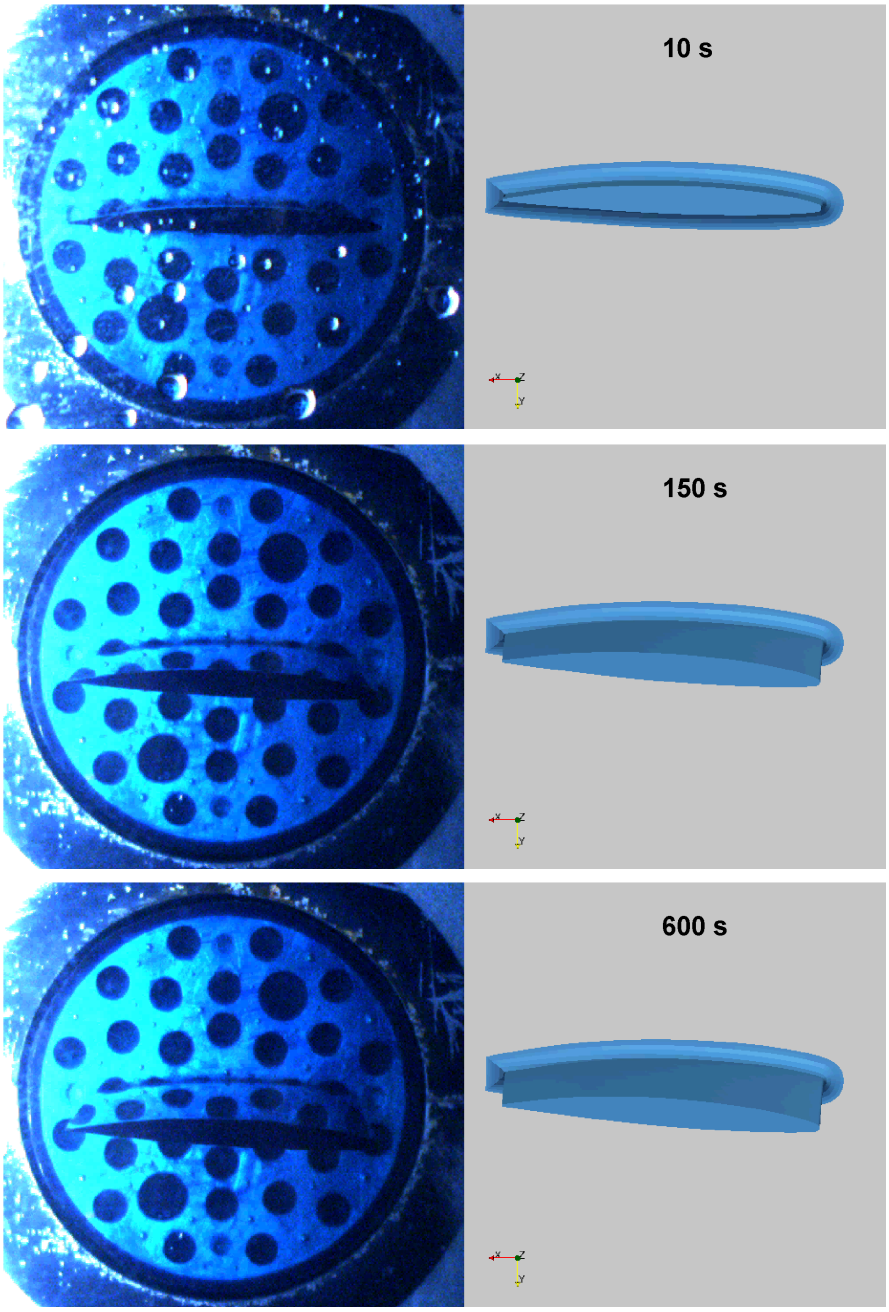

Fig. 5.32. Fin deformation comparisons between experiment and simulation for time 10 s to 600 s (window surface bubbles present in top left image)

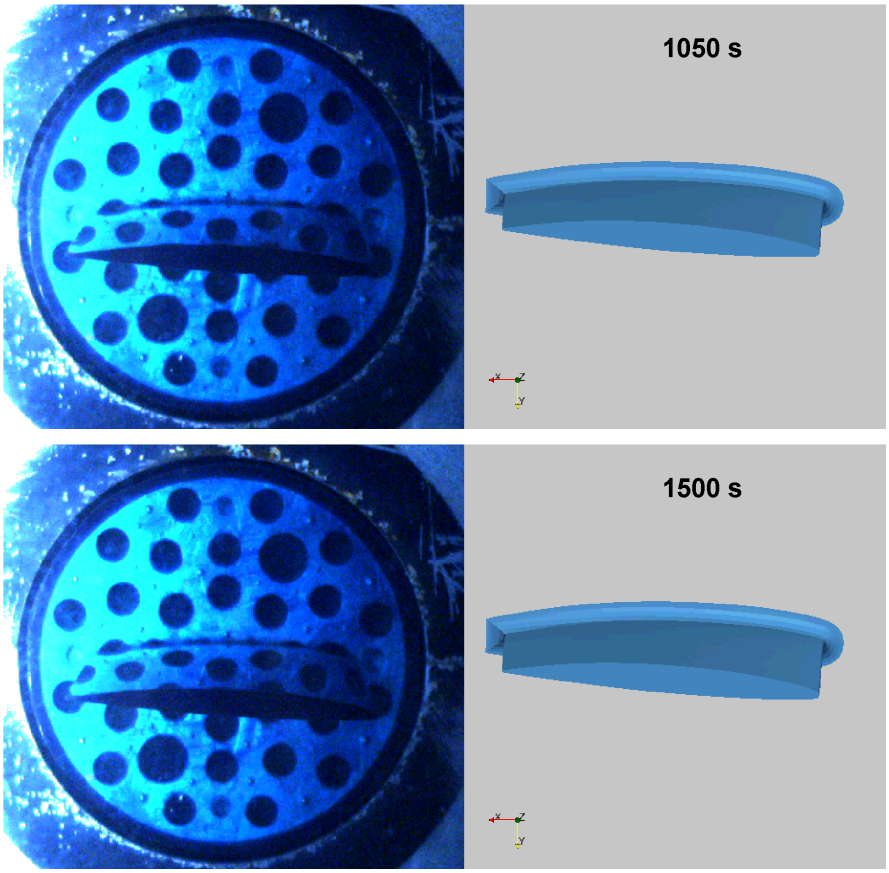

Fig. 5.33. Fin deformation comparisons between experiment and simulation for time 1050 s to 1500 s

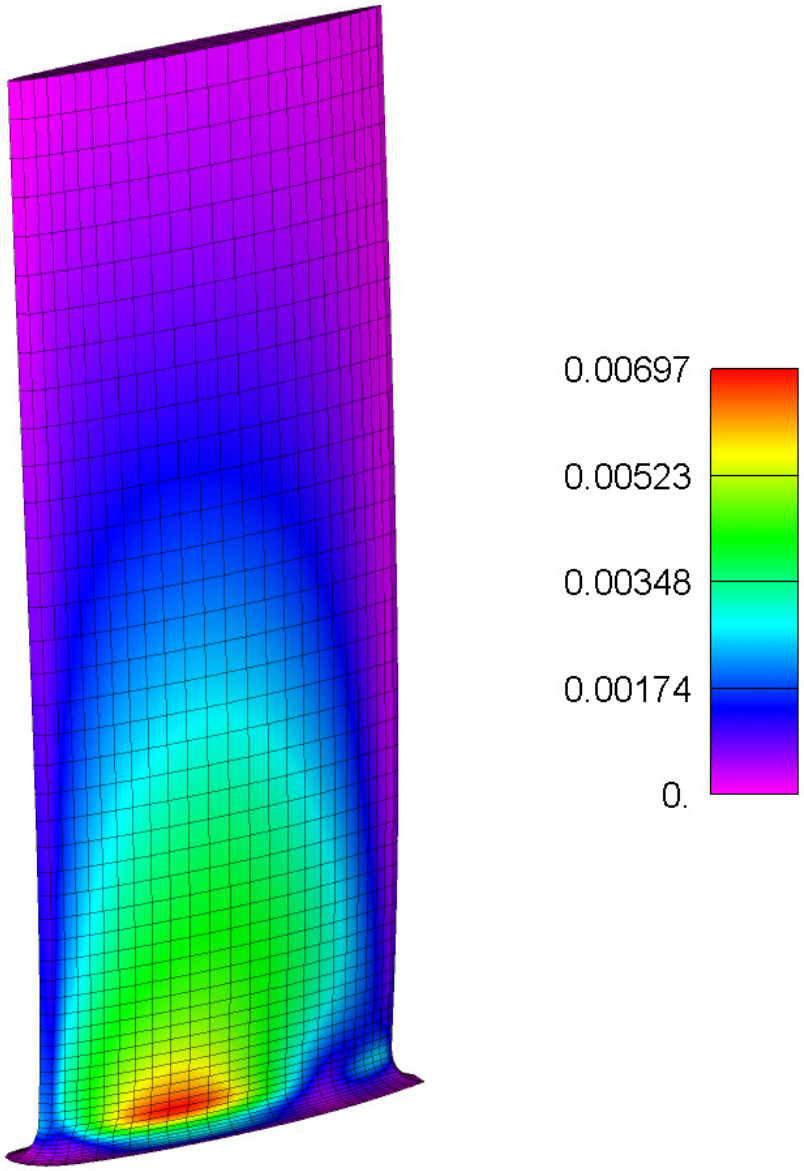

Fig. 5.34. Fin strain contour for the -1.375° AOA FSI simulations at $t = 1500$ s

evident from these figures that the flow solver requires the most time, followed by either the structural solver if the custom RMF mesh motion solver is used, or by the mesh motion solver if the Laplace face decomposition solver is used. Similar results are obtained for parallel execution of the solver using 16 processors, as shown by the results in Figure 5.37 for the RMF mesh motion solver. A summary of these results, in terms of the solution time percentages for each of the three fields per simulation, is provided in Table 5.3.

Table 5.3. Comparison of solver elapsed wall-clock times

Case	Flow	Structure	Mesh Motion
1 Processor, Laplace Face Decomposition	74.8%	4.1%	21.1%
1 Processor, RMF	93.2%	5.9%	0.9%
16 Processors, RMF	81.0%	18.5%	0.5%

It should be noted that the initial solves of the structure require the most time because it is during this time that the largest change in structure shape occurs. Several solution increments are required to simulate this time period. Later in the simulation, the structure's time step size is increased and requires less time per FSI time increment.

Memory requirements were characterized using the fin model for the Laplacian face decomposition and the custom RMF mesh motion solvers. The former requires approximately 1.9 GB, while the latter approach requires approximately 1.2 GB. Using the domain decomposition approach for parallel processing, the per-processor memory requirement decreases from these values.

5.6.2 Solver Subiterations

A study was conducted to determine the number of subiterations required to tightly couple the flow and structure solvers. The study was conducted using a

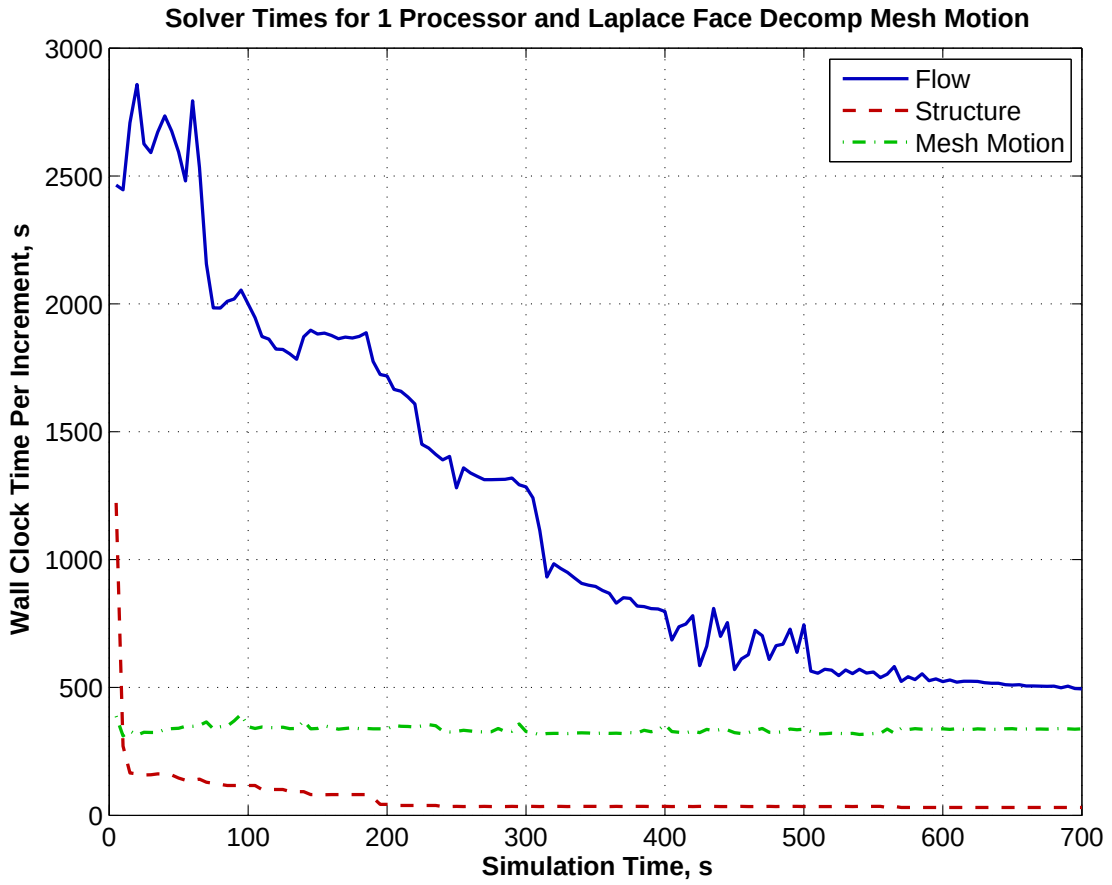

Fig. 5.35. Elapsed wall-clock times per solution time increment for each of the three FSI fields for the modified NACA 66 fin and the Laplace face decomposition mesh motion solver

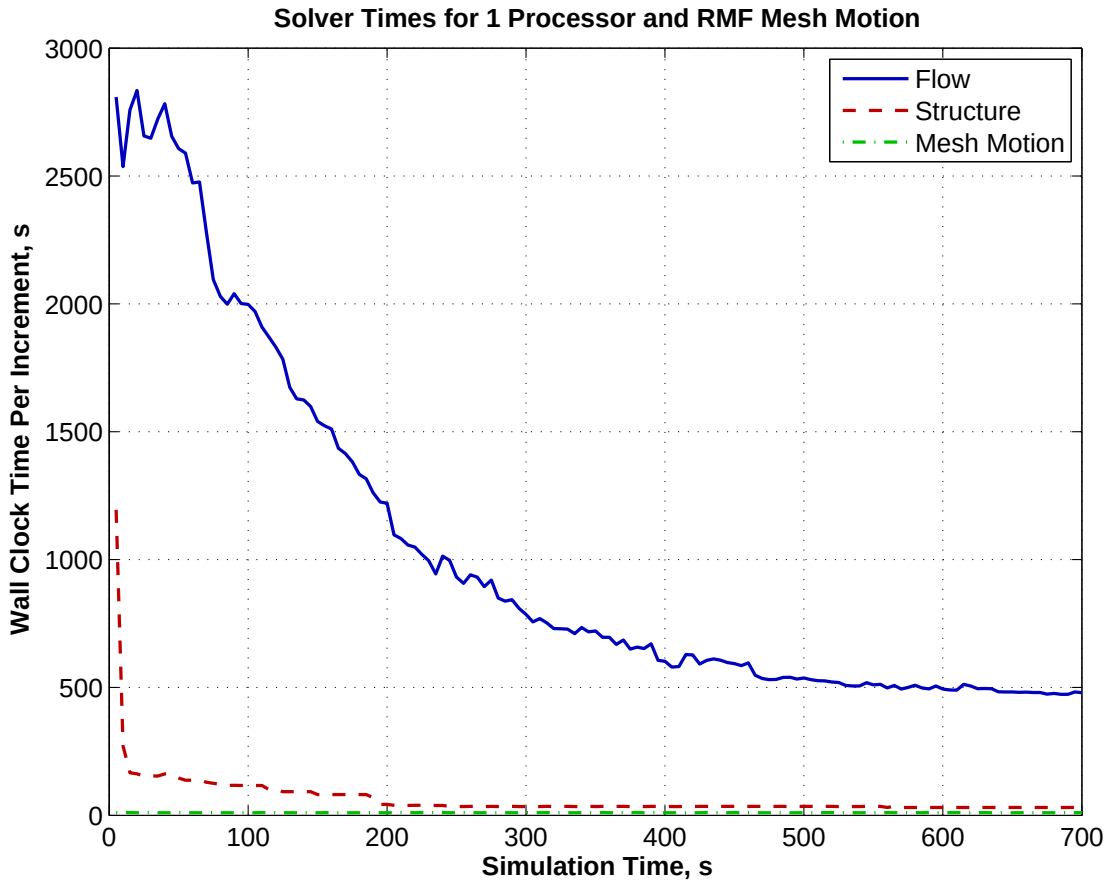

Fig. 5.36. Elapsed wall-clock times per solution time increment for each of the three FSI fields for the modified NACA 66 fin and the custom RMF mesh motion solver

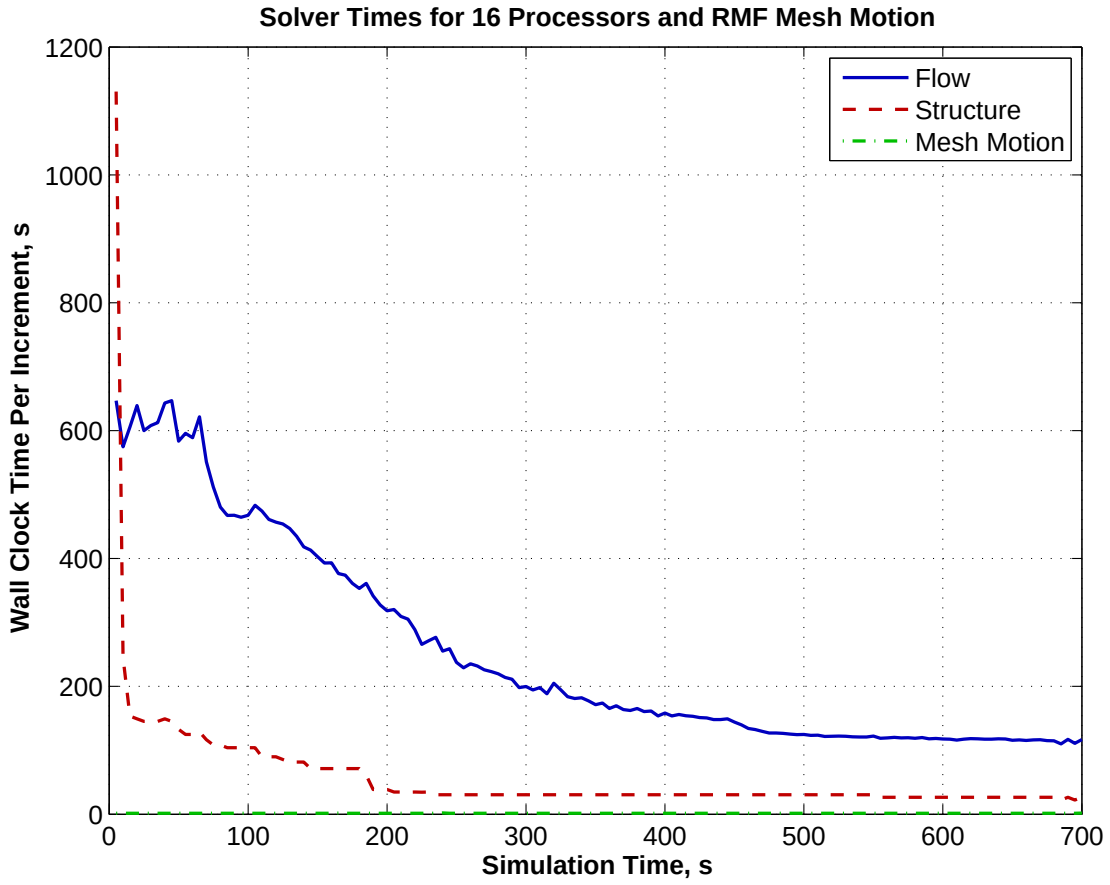

Fig. 5.37. Elapsed wall-clock times per solution time increment for each of the three FSI fields for the modified NACA 66 fin and the custom RMF mesh motion solver; solver executed in parallel on 16 processors

fixed relaxation parameter, ω_i, for the fixed-point relaxation shown in Figure 2.12. The medium-refined mesh with an AOA of -1.125° and inlet flow speed of 1.22 m/s was employed for this study. The leading and trailing edge points at 100% span and lift force versus time are compared in Figures 5.38, 5.39, and 5.40, respectively. The simulation results are reported to 700 s in these figures. The results show good agreement between each of the different degrees of coupling. However, if the results are plotted on a logarithmic scale for the abscissa, Figures 5.41, 5.42, and 5.43, it is evident that differences are largest during the first 100 seconds of the simulations. During this period, the inlet flow speed changes and the fin deformations are due to elastic relaxation, which has a short time scale. After 100 seconds, the inlet flow speed is constant and only the viscoelastic material relaxes. The results for coupling with less than ten sub-iterations are compared against the results using ten sub-iterations for the leading edge and lift force in Figures 5.44 and 5.45. A similar plot for the trailing edge deflection is not shown because the trailing edge deflection passes through zero causing the %-error results to be misleading (i.e., a division by zero). These results demonstrate that the use of 10 fixed relaxation sub-iterations provides results that are tightly coupled to within a few percent error during tunnel startup (i.e., $t \leq 100$ s) and much more accurate for $t > 100$ s. Note that these results employed a simulation time step size of 5 s. Decreasing the time step size would reduce the number of required sub-iterations for the same coupling accuracy or alternatively improve the coupling accuracy while maintaining the same number of sub-iterations. Based on these results, all simulations reported here employ a constant relaxation factor with ten sub-iterations and the dynamic relaxation factor of Equation 2.49 is not used.

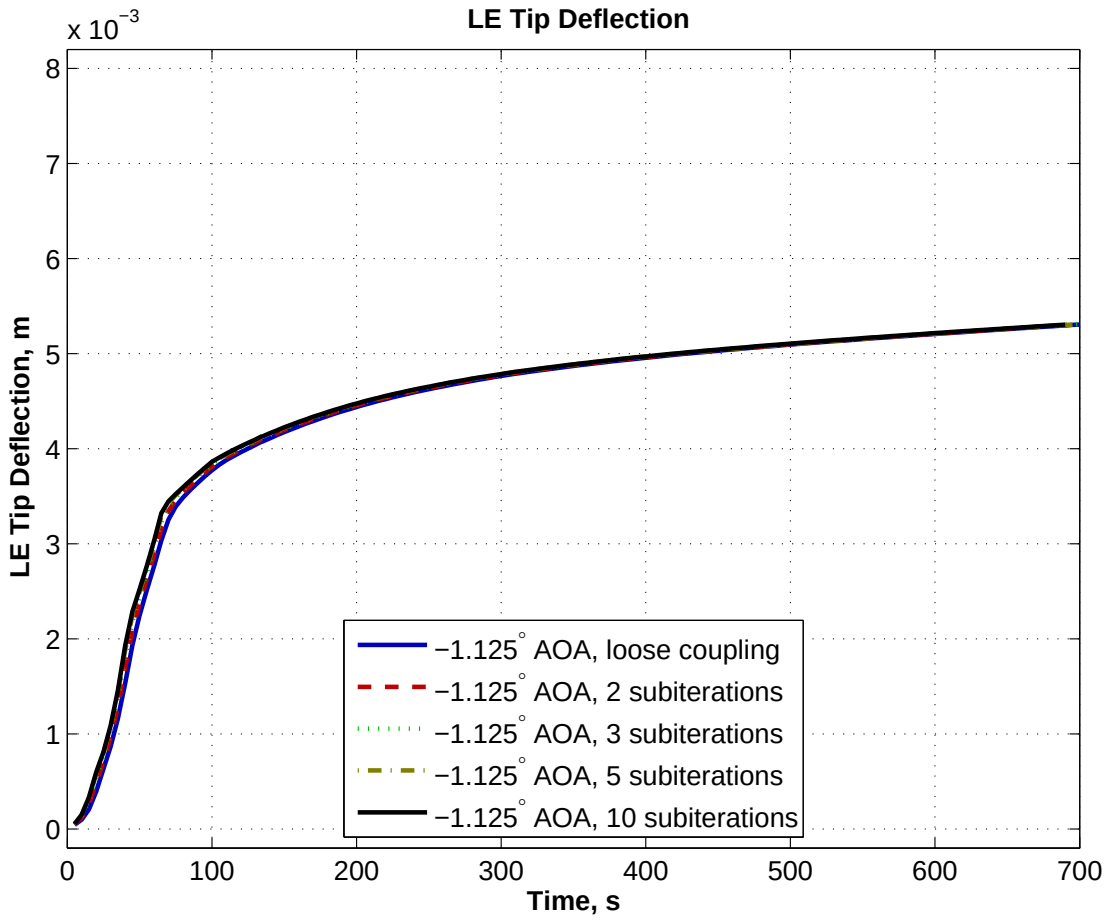

Fig. 5.38. Fin leading edge tip deflection at 100% span for various numbers of sub-iterations with constant under-relaxation

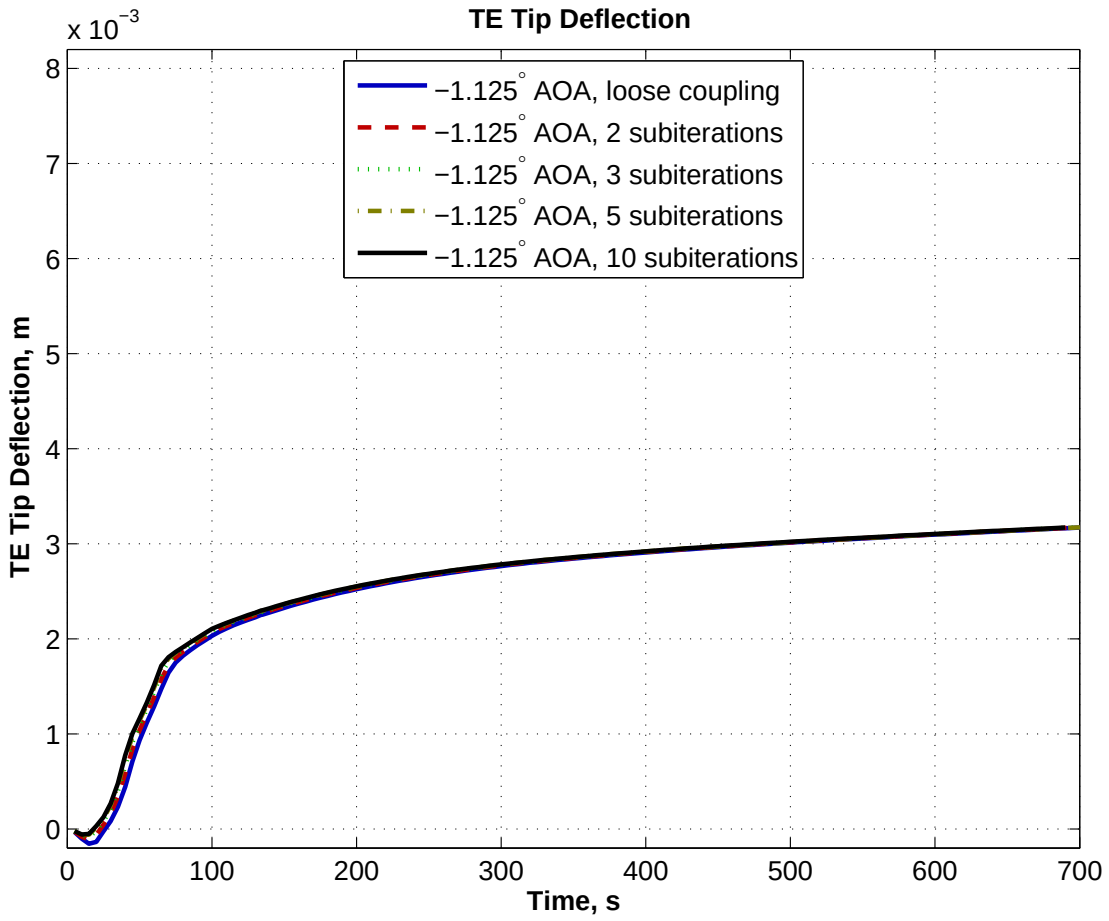

Fig. 5.39. Fin trailing edge tip deflection at 100% span for various numbers of sub-iterations with constant under-relaxation

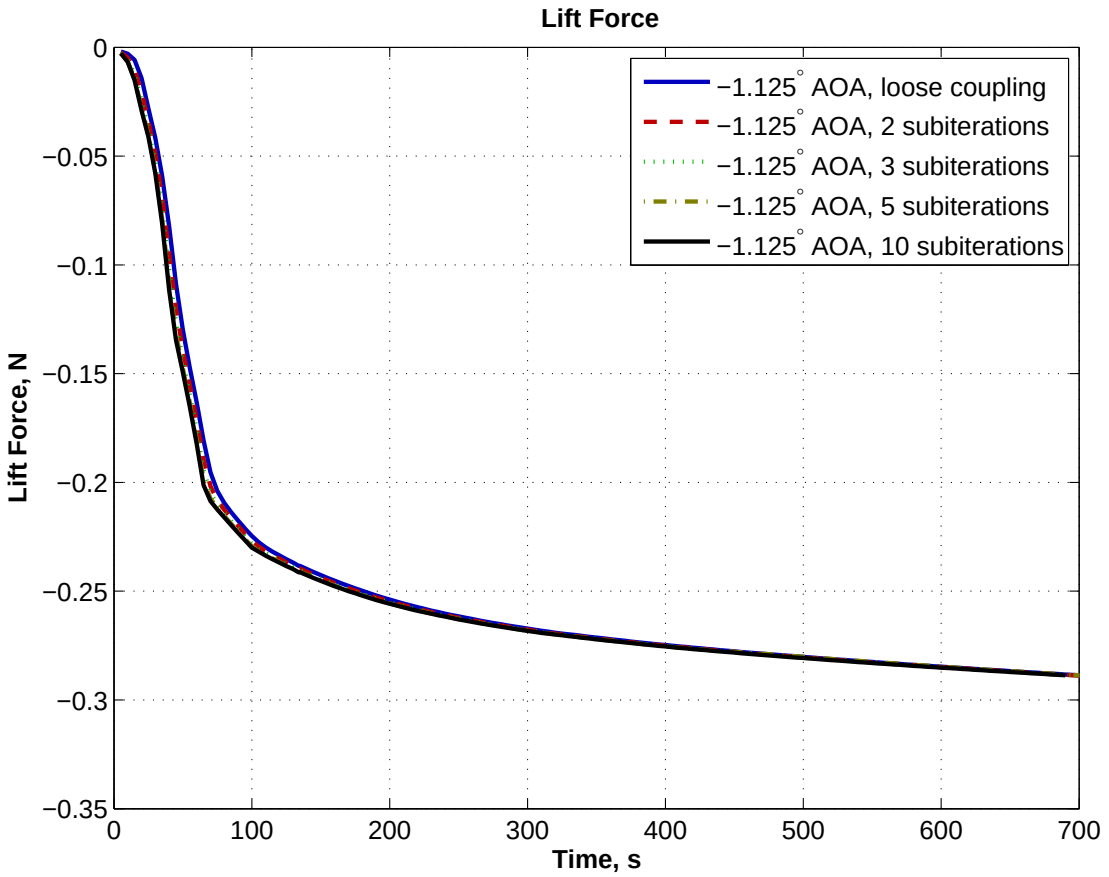

Fig. 5.40. Fin lift force for various numbers of sub-iterations with constant under-relaxation

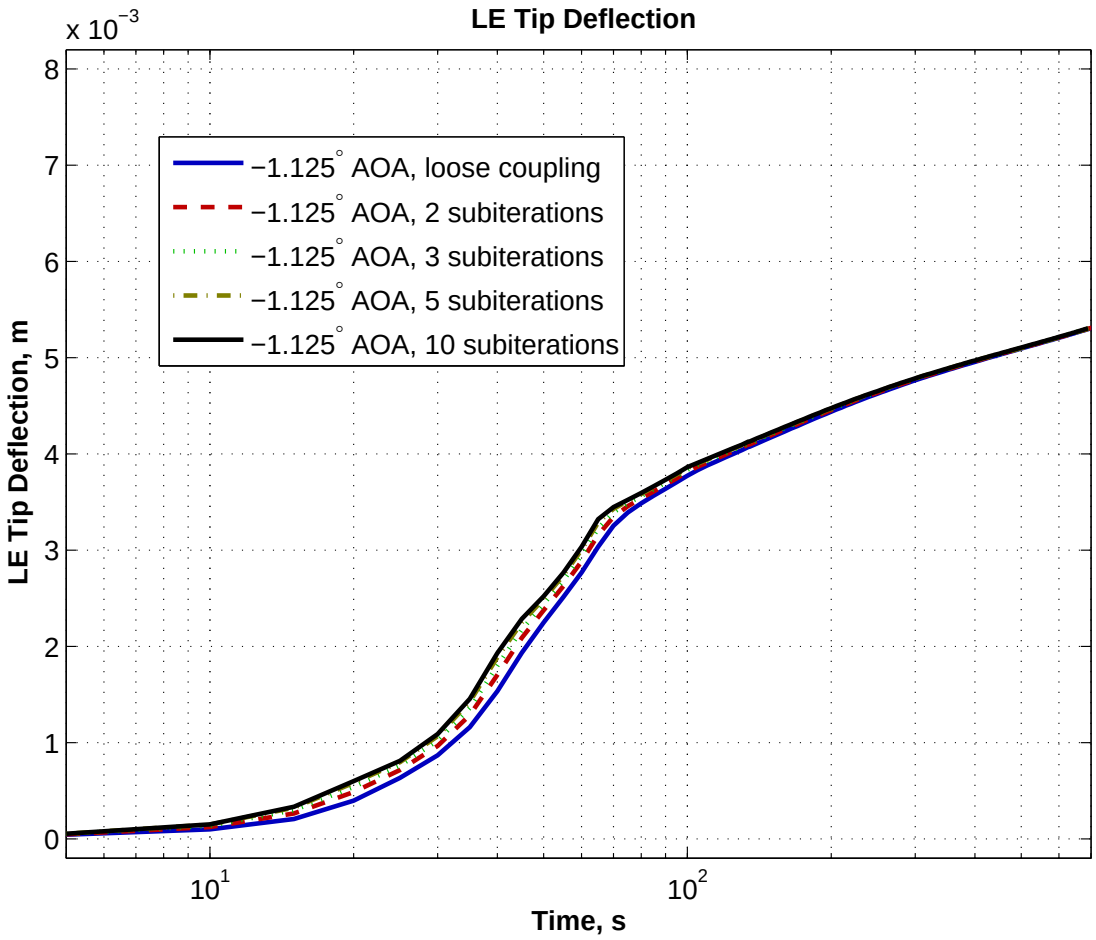

Fig. 5.41. Fin leading edge tip deflection at 100% span for various numbers of sub-iterations with constant under-relaxation; plotted on logarithmic scale

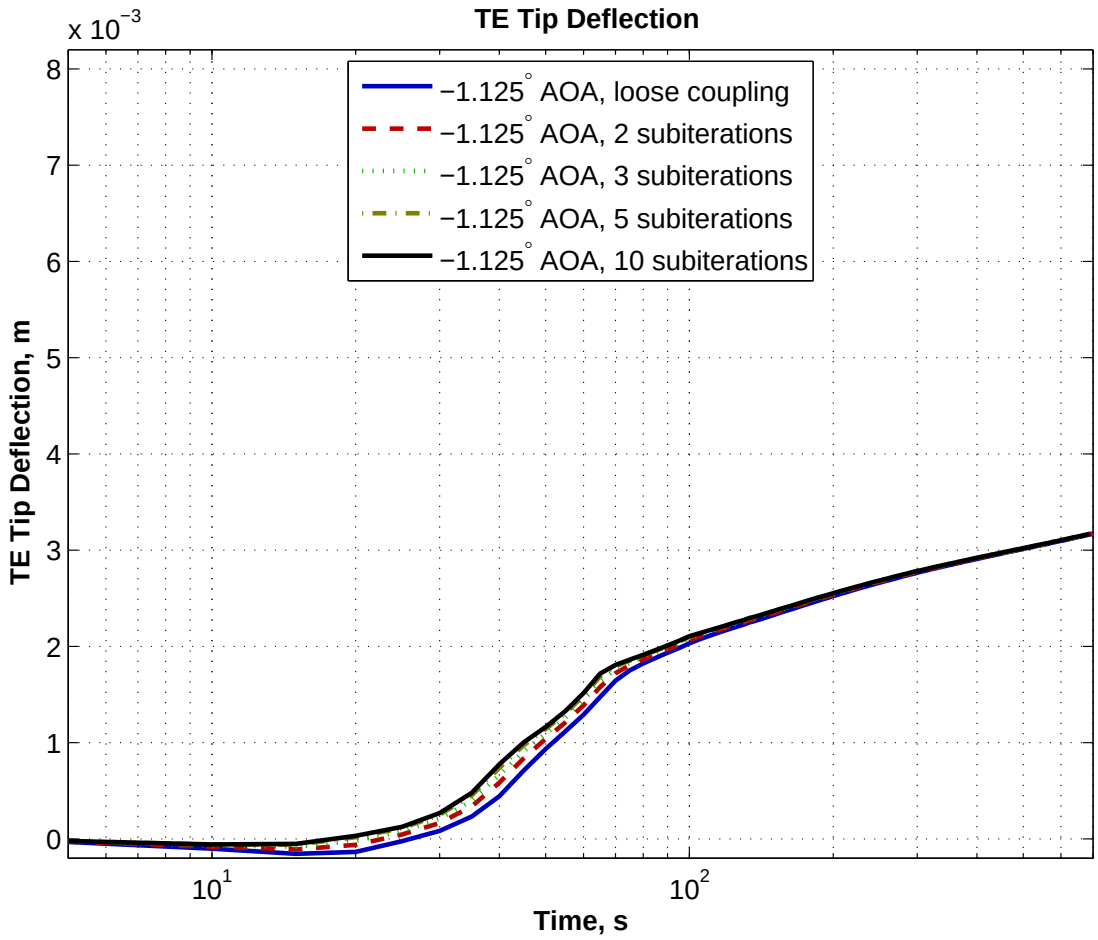

Fig. 5.42. Fin trailing edge tip deflection at 100% span for various numbers of sub-iterations with constant under-relaxation; plotted on logarithmic scale

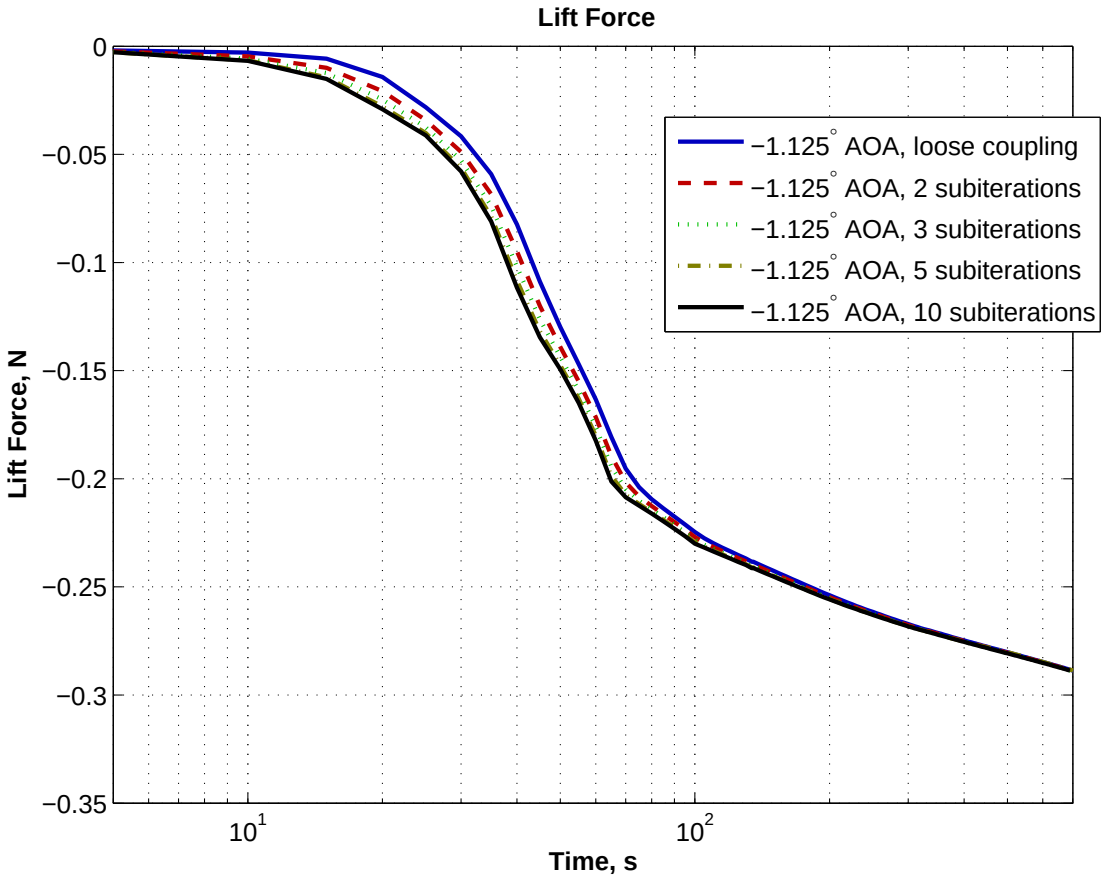

Fig. 5.43. Fin lift force for various numbers of sub-iterations with constant under-relaxation; plotted on logarithmic scale

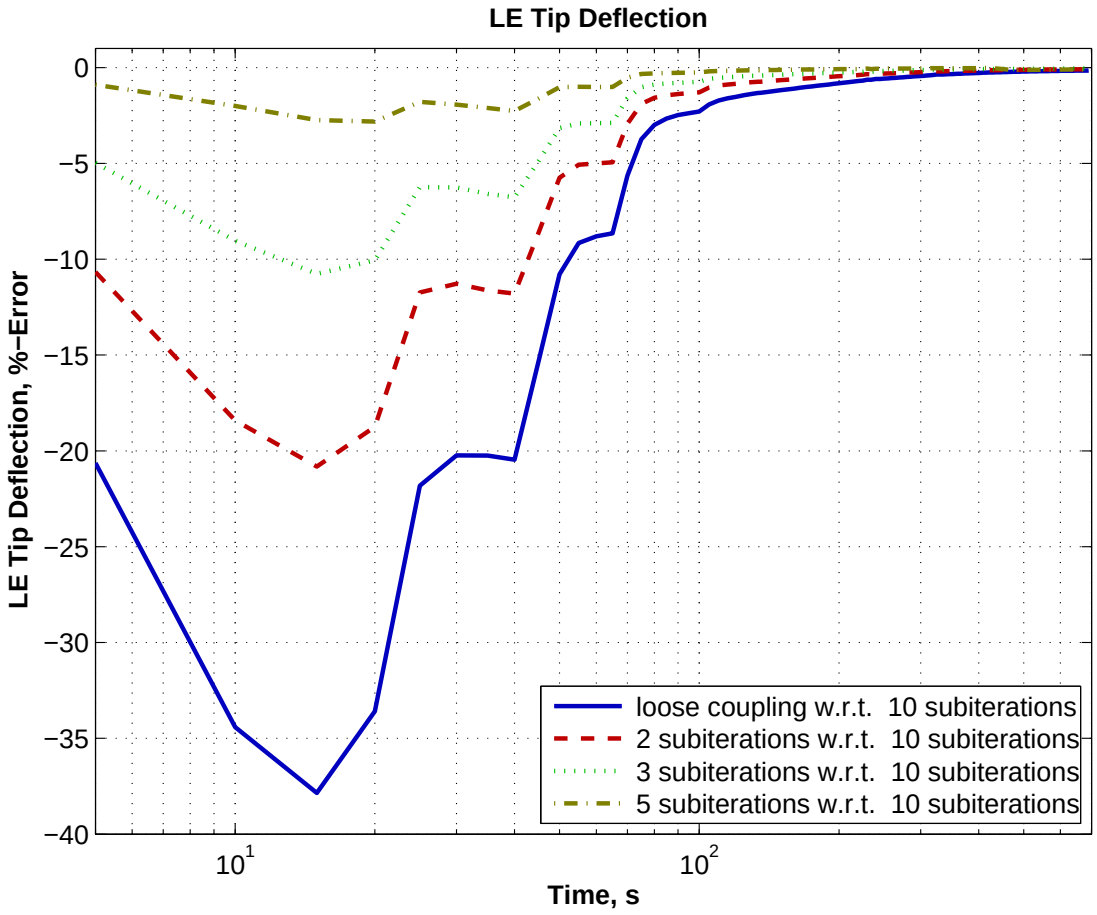

Fig. 5.44. Fin leading edge tip deflection at 100% span for various numbers of sub-iterations with constant under-relaxation compared to case with ten sub-iterations

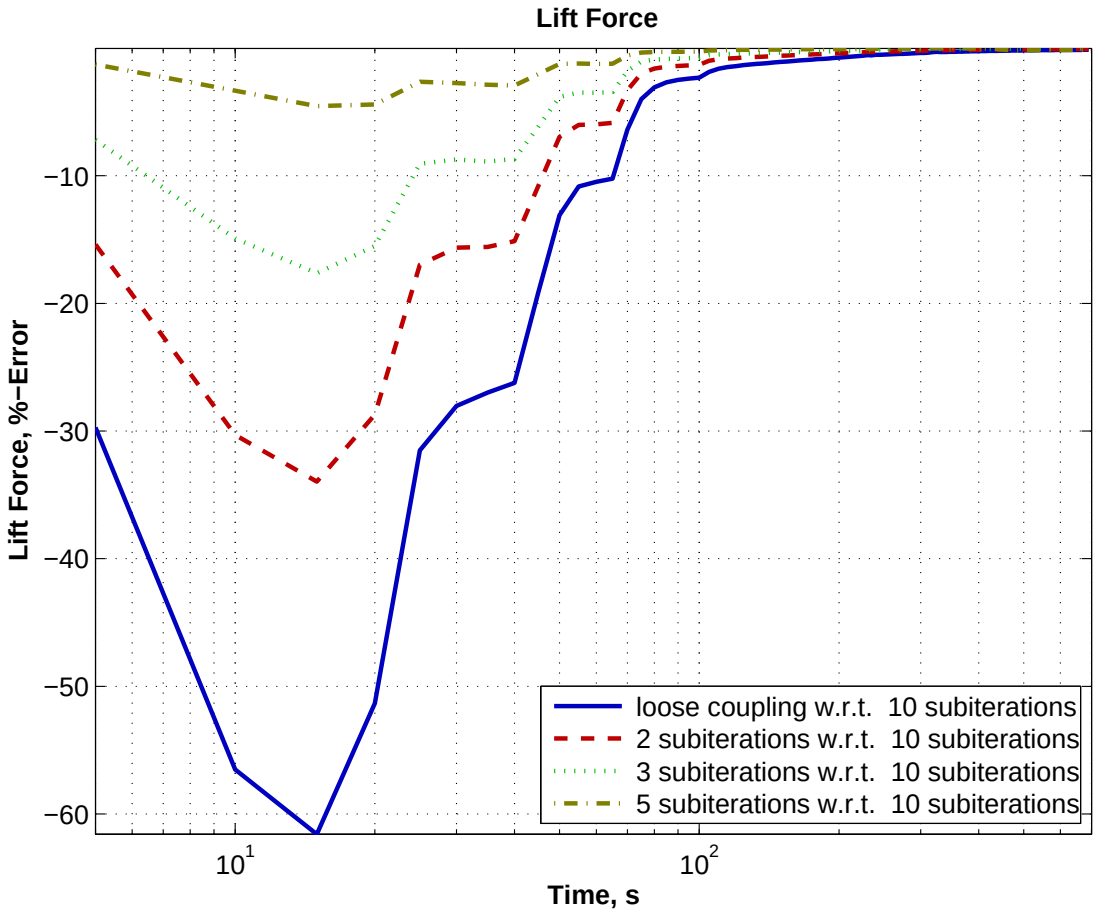

Fig. 5.45. Fin lift force for various numbers of sub-iterations with constant under-relaxation compared to case with ten sub-iterations

Chapter 6

Modified NACA 66 Fin Inverse Model Simulations

The validated FSI and inverse finite element solvers are applied to the modified NACA 66 fin to assess the validity of using the inverse approach for the time-dependent, viscoelastic response. The objective of the inverse model is to define an inverted shape that deforms into the desired (design) shape due to fluid loading.[1] The inverse simulations are evaluated for several fin AOA's. FSI simulations are used for each case to evaluate the validity of the inverted shapes.

6.1 Approach

Simulations of an inverted model require the following steps. First, the loads acting on a fin in its *design* shape are determined. This is accomplished by running a modified version of `simpleFsiFoam`, called `simpleFsiFoamRigidStructure`. This solver treats the structure as a rigid entity but still calculates the fluid loading on the structure based on the interface coupling scheme for the disparate meshes. These forces are applied to the structural model and the inverse FE solver, `ifeanl`, is used to determine node locations for the corresponding inverted model.

Barring the use of the flow solver's mesh motion capability, the next step would involve the creation of a new fluid mesh. Rather than performing this labor-intensive task, the mesh motion solver is used, along with the structural solver's member function `setNodeDisplacements`, to deform the fluid mesh. The `simpleFsiFoam` solver is once-again modified to accomplish this task, and the new solver is called `fsiInverseFoam`. This solver reads a text file containing the

[1] If the material exhibited purely elastic deformation, then the inverse shape would deform "into" the design shape. Viscoelasticity introduces a temporal component to the material's response and thus the shape will only correspond to the design shape for an instant as it "passes through" the shape.

structural nodal displacements from the design shape to the inverted shape, and moves the fluid mesh accordingly. The moved mesh then becomes the starting mesh for the inverse FSI analysis. The inverted structural model created from `ifeanl` is also used in the inverted FSI analysis.

The use of a stress-relaxing material requires the inverse evaluation to occur over a specified amount of time, and ideally the forward FSI analysis will show the inverted model deforming into the design shape after being exposed to the flow field for that same amount of time. Some error is anticipated because of the nonlinearity of the history-dependent, viscoelastic material. As such, load ramps are used when appropriate to approximate the anticipated load history of the fluid on the structure during deformation from the inverted shape to the design shape.

6.2 Results

The inverse analysis of the modified NACA 66 fin employs the same fluid and structural meshes described previously. The flow conditions are similar to those described elsewhere: laminar, incompressible, the same fluid and viscoelastic material properties used previously, and a uniform inlet velocity of $(1.2\hat{\boldsymbol{i}}, 0\hat{\boldsymbol{j}}, 0\hat{\boldsymbol{k}})$ m/s that is gradually applied with the temporal variation similar to that shown in Figure 5.25 but scaled by the factor 1.2/1.22 to yield a maximum velocity of 1.2 m/s.[2] The inverse structural analysis is used to determine the inverted shape corresponding to $1,000$ s of relaxation for the design-shape fluid loading at AOA's -1.5°, -2°, and -3°. Comparisons of the design- and inverse-shapes are provided in Figure 6.1 for an AOA of -3° showing a three-dimensional view of the fin outline and top view of all three AOA's in Figures 6.2 to 6.4. As evident from these figures, the difference between the inverse and design shape increases with increasing magnitude of the AOA, as expected. To ensure the inverted shape is accurate, the fluid design-shape

[2] The flow speed objective of the water tunnel tests was 1.2 m/s, but was closer to 1.22 m/s. Therefore, many of the FSI simulations were performed at 1.2 m/s and additional FSI simulations at 1.22 m/s were performed for the validation study.

loads are applied to the inverted models and comparisons are made between the deformed inverted shape at $1,000$ s and the design shape. The difference between each node of the design geometry and the deformed inverse geometry is shown in Figures 6.5 to 6.7 for the three AOA's. The maximum error for all cases is less than $6.5\text{x}10^{-7}$ m, which is about 5 orders of magnitude less than the chord length, thereby validating the inverse shapes.

Ideally, the FSI solver would show the inverted shape deforming into the design shape at 1000 s. Plots of fin tip location at the leading and trailing edges versus time are used to quantify the results. The results are provided in Figures 6.8 to 6.10. In each of these figures, it is evident that time at which the fin shape coincides with the design shape occurs later than the target time of 1000 s, between 40 and 80 s late for the cases shown here. The reason for this discrepancy is related to the material time-dependency and discrepancy between actual and simulated load history. The inverse shapes are determined assuming a linear load application from 0 to the maximum at the design conditions over a period of 100 s. In reality, the load time-dependency is unknown but could be estimated by iteration of the inverse problem stated here. Plots of load magnitude versus time for the inverse FE analysis and the FSI simulation of the inverted shape are shown in Figures 6.11 to 6.13, corresponding to the results in Figures 6.8 to 6.10, respectively. Each of these load plots shows the inverse FE analysis applied loads that are generally of larger magnitude than the load resulting from the FSI simulations. This discrepancy is the underlying cause of each of the FSI simulations missing the target times in Figures 6.8 to 6.10.

It is expected that increases in the AOA beyond a certain value will cause the forward FSI simulation to diverge from the design shape, but this is not explored here. Rather, it is observed that the inverse analysis technique can produce an inverted shape that deforms into the design shape. However, to arrive at the design shape at a specific time requires additional knowledge of the load, a priori.

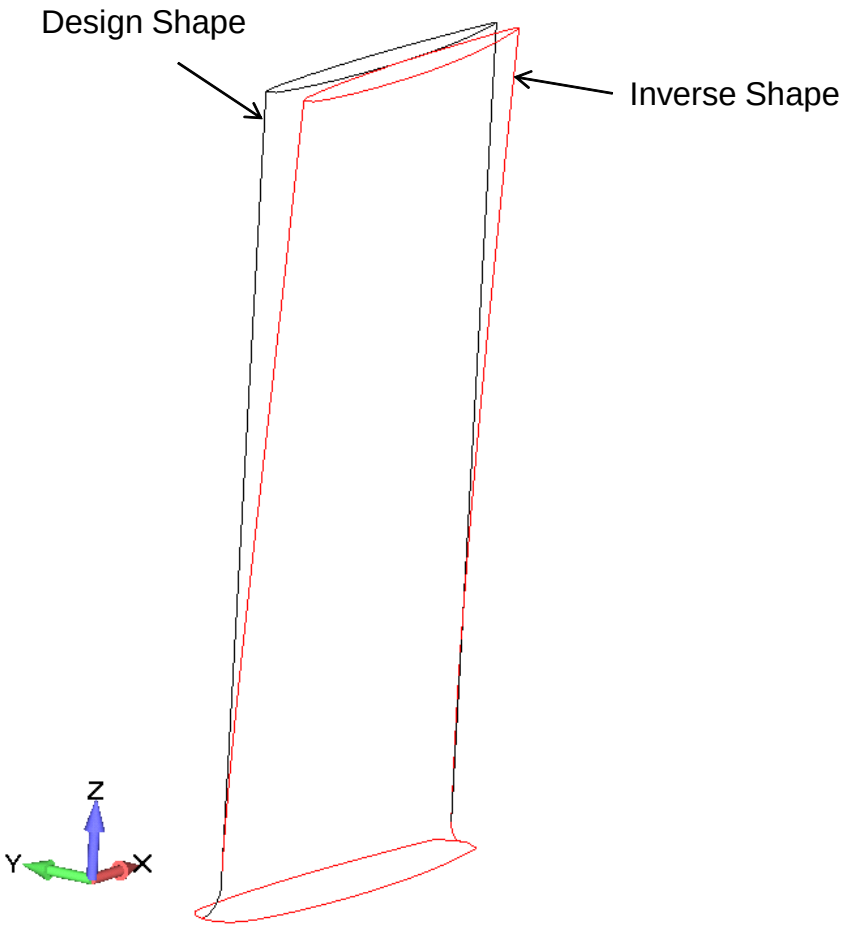

Fig. 6.1. Inverted fin shape corresponding to 1000 s of relaxation under the fluid loads for a fin with an AOA of -3° compared to design shape fin

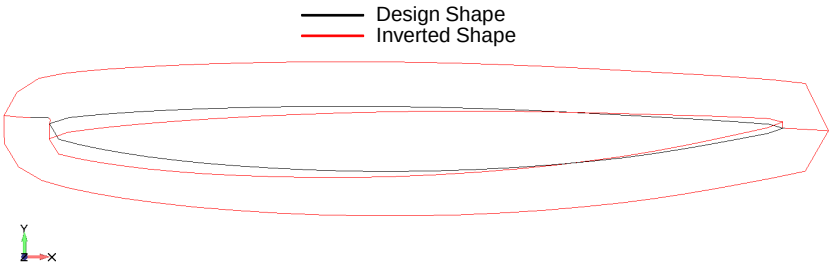

Fig. 6.2. Top view of inverted fin shape corresponding to 1000 s of relaxation under the fluid loads for a fin with an AOA of -1.5°

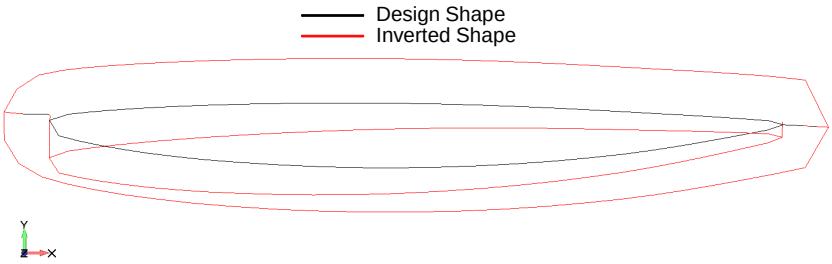

Fig. 6.3. Top view of inverted fin shape corresponding to 1000 s of relaxation under the fluid loads for a fin with an AOA of $-2.^{\circ}$

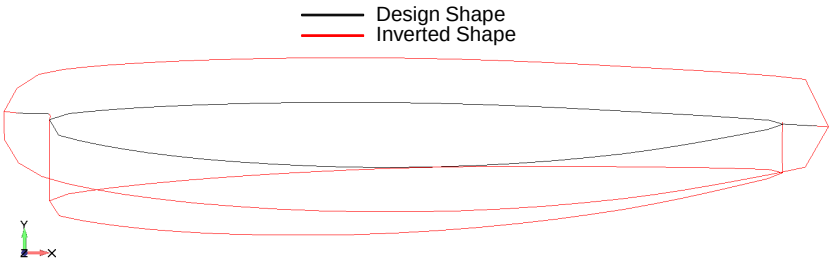

Fig. 6.4. Top view of inverted fin shape corresponding to 1000 s of relaxation under the fluid loads for a fin with an AOA of $-3.^{\circ}$

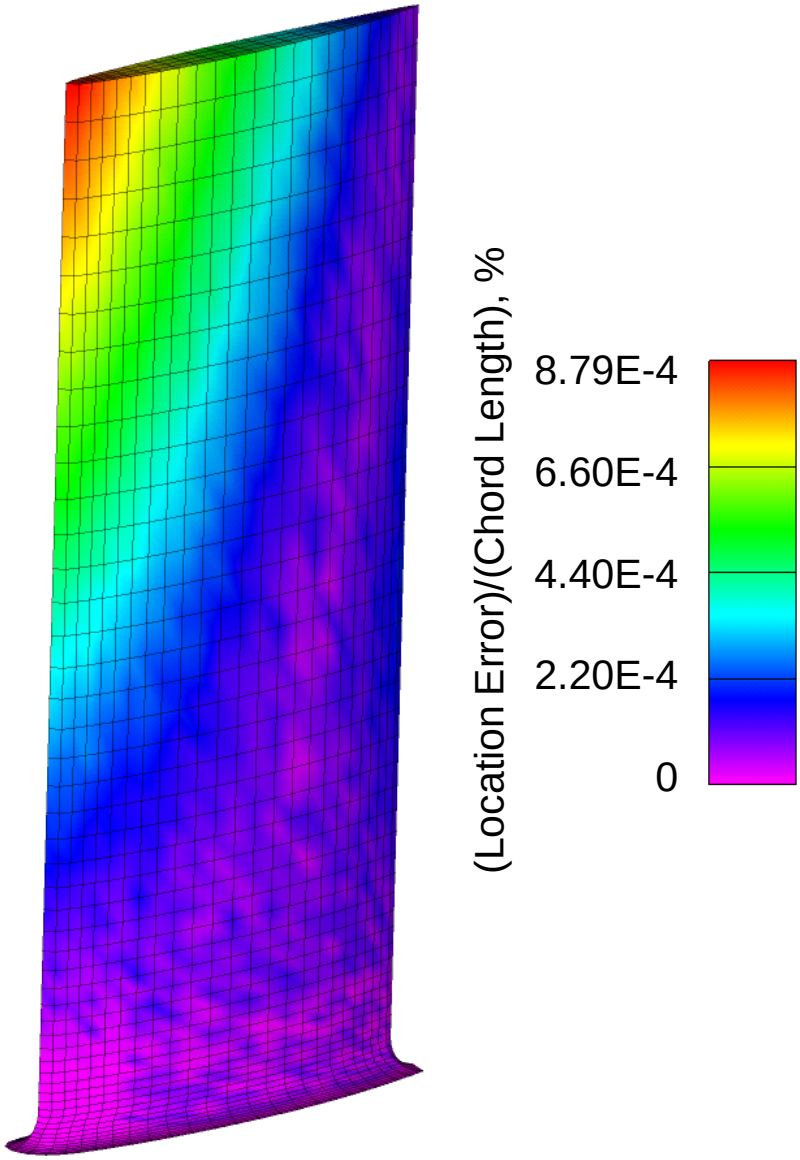

Fig. 6.5. Node location error (shown as a percentage of chord length) between design shape and deformed inverse shape for a -1.5° AOA

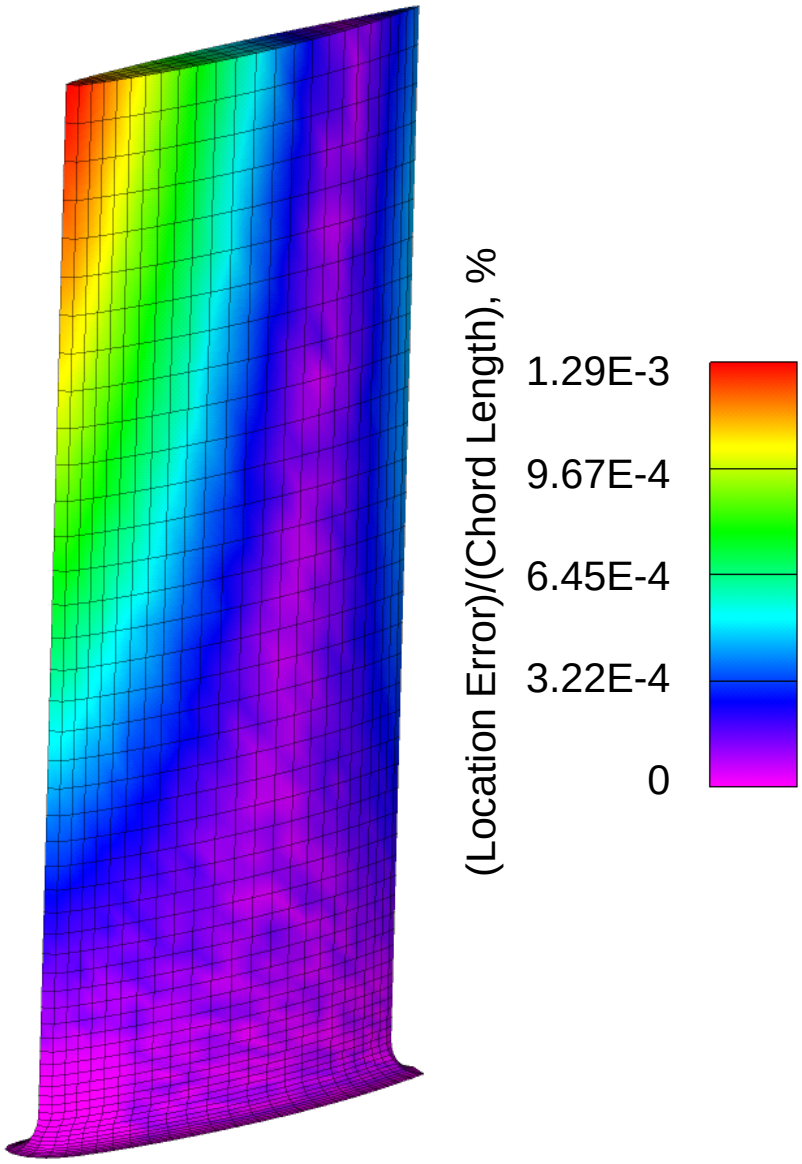

Fig. 6.6. Node location error (shown as a percentage of chord length) between design shape and deformed inverse shape for a -2° AOA

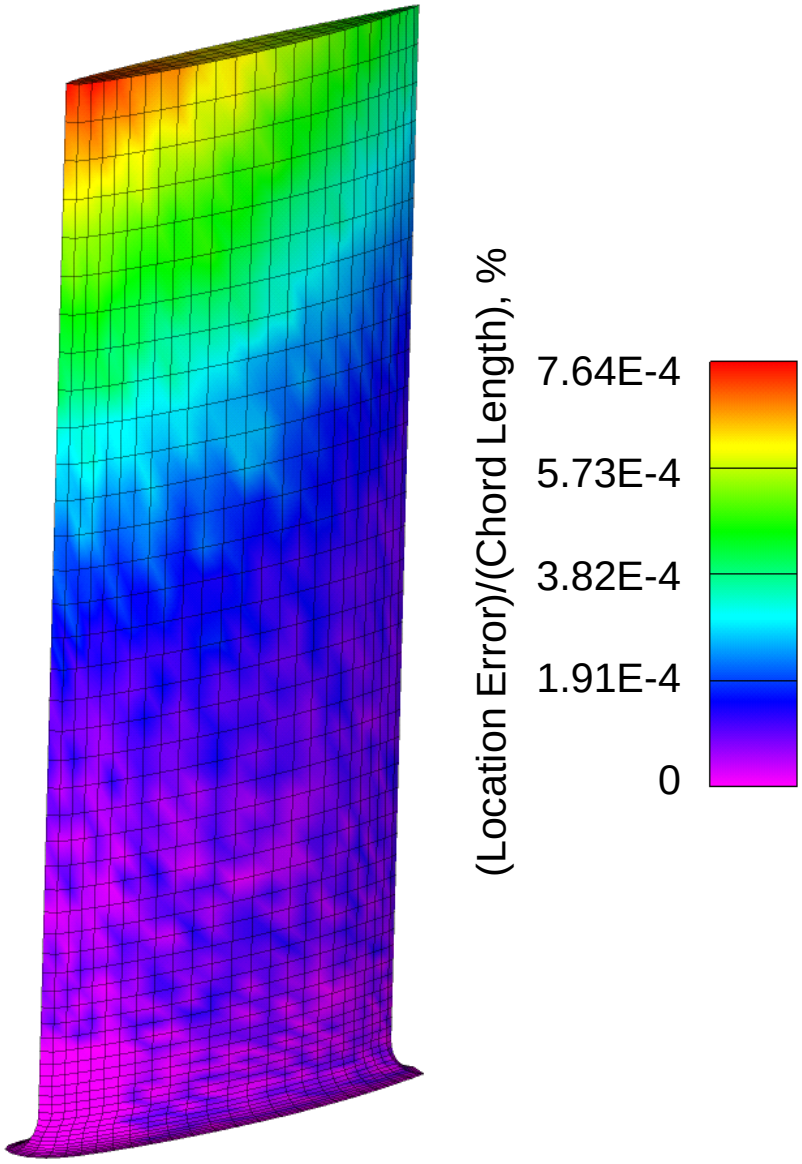

Fig. 6.7. Node location error (shown as a percentage of chord length) between design shape and deformed inverse shape for a -3° AOA

Such a requirement can be met with an iterative approach, but this is not explored here but instead recommended for future work in this area.

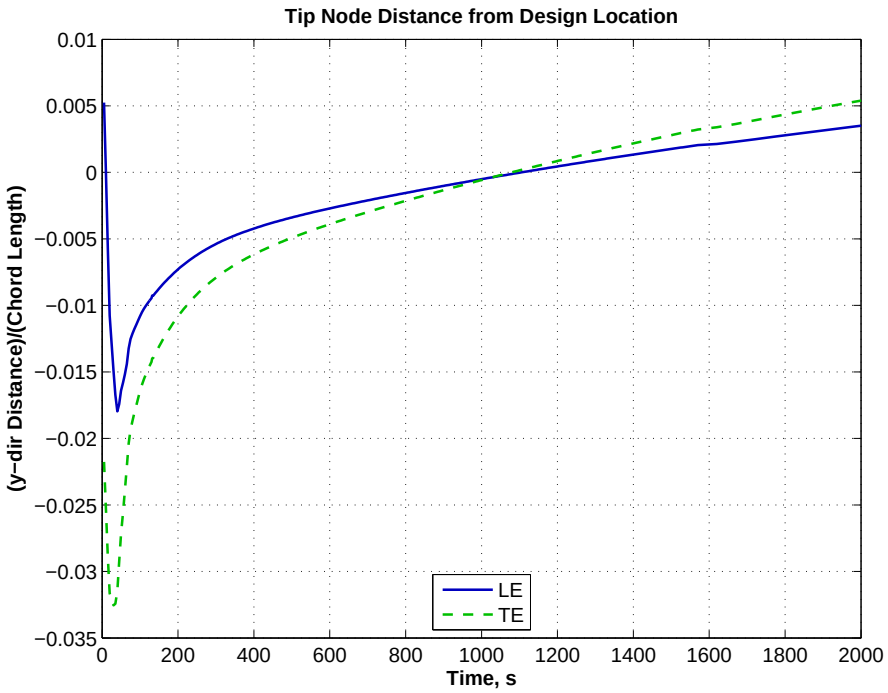

Fig. 6.8. FSI simulation of inverted fin for -1.5° AOA; deviation of leading and trailing edge points from the design shape location

6.3 Conclusions

The inverse structural FE solver accurately defines inverse shapes for the modified NACA 66 fin for the given load history. These inverse shapes do deform into the design shapes when subjected to flow stresses. However, slight discrepancies exist in meeting the target times at which the shape should conform to the design shape. This is attributed to inaccurate load histories used in the inverse FE analysis. For cases where the time at which the design shape is achieved is of

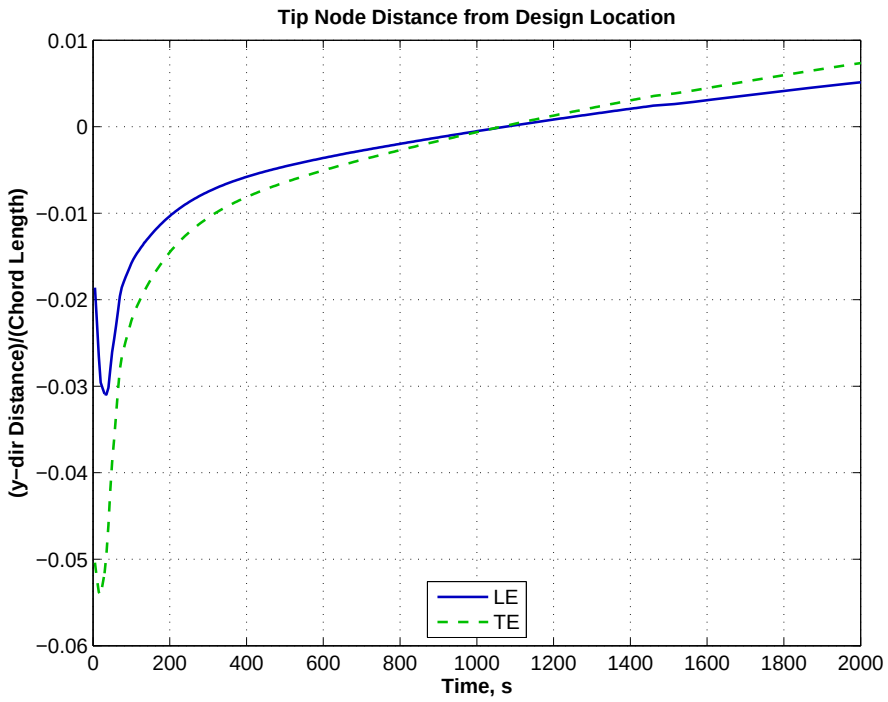

Fig. 6.9. FSI simulation of inverted fin for -2.0° AOA; deviation of leading and trailing edge points from the design shape location

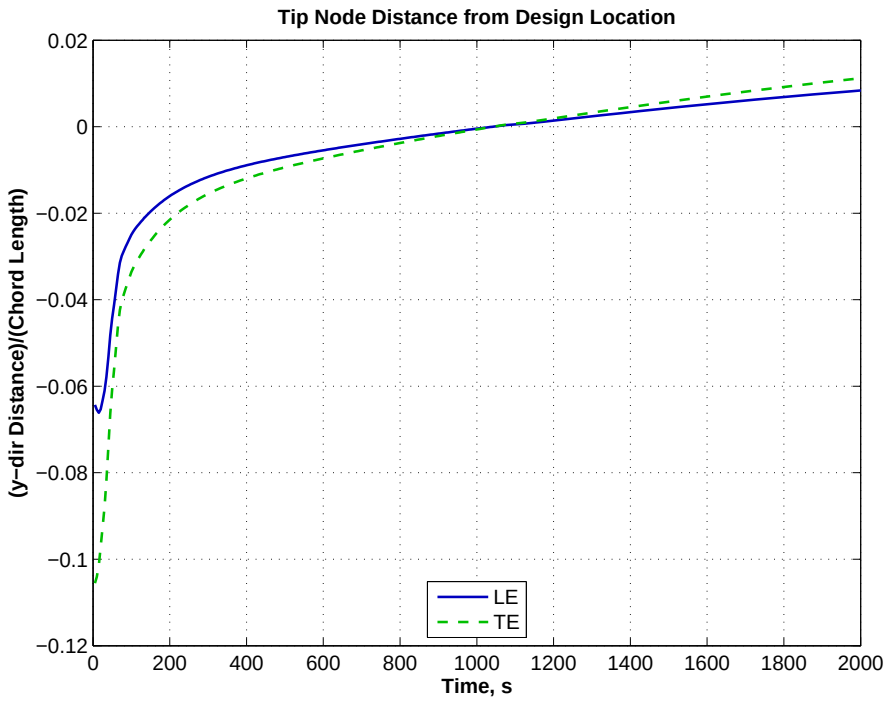

Fig. 6.10. FSI simulation of inverted fin for -3.0° AOA; deviation of leading and trailing edge points from the design shape location

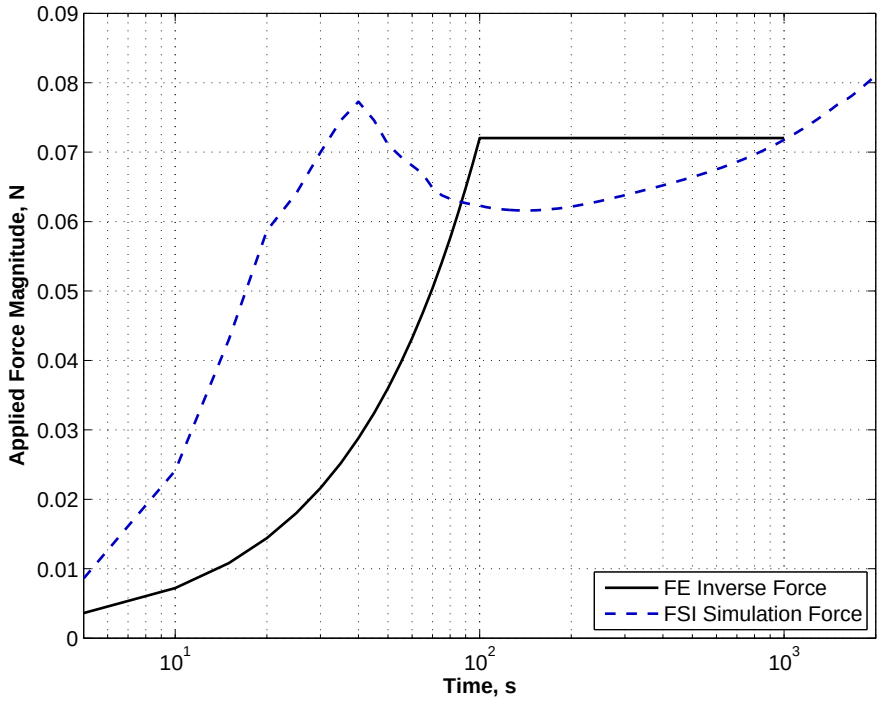

Fig. 6.11. FSI simulation of inverted fin for -1.5° AOA; FSI force compared to force used during FE inverse analysis

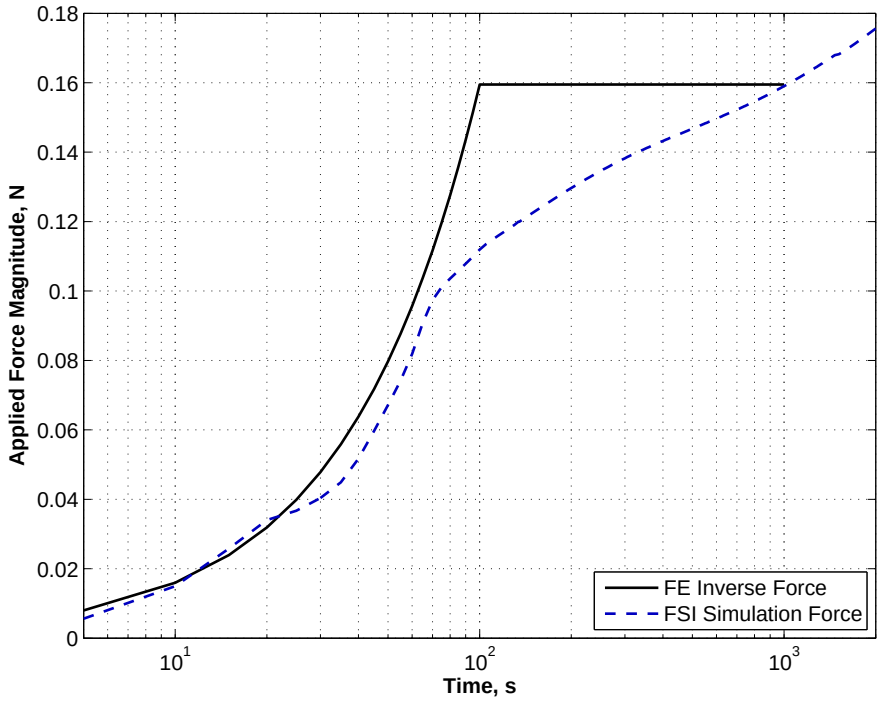

Fig. 6.12. FSI simulation of inverted fin for -2.0° AOA; FSI force compared to force used during FE inverse analysis

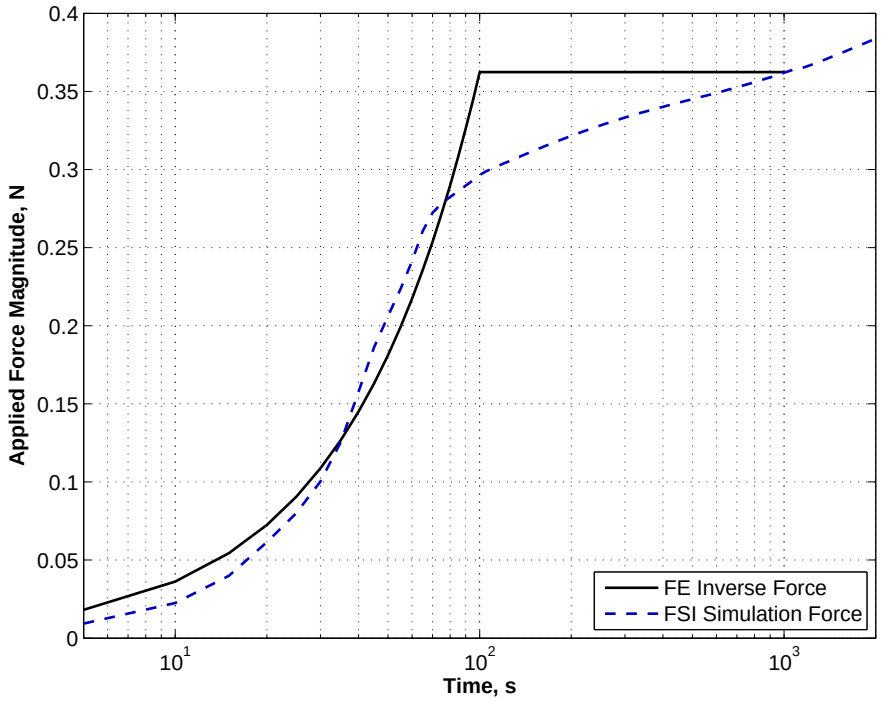

Fig. 6.13. FSI simulation of inverted fin for -3.0° AOA; FSI force compared to force used during FE inverse analysis

utmost importance, an iterative approach will be required to refine the load history used in the inverse structural analysis. It is also speculated that some inverse shapes can lead to a diverging system that requires a system modification, such as a time-varying AOA, to achieve the design shape. The next chapter demonstrates the inverse approach for the viscoelastic expandable impeller pump.

Chapter 7

Expandable Impeller Pump Simulations

One of the objectives of this thesis is to determine if an inverse technique can be used to derive a manufacture shape for a viscoelastic expandable impeller pump. The goal of the manufacture, or inverse, shape is to accommodate impeller deformations that will occur due to elastic and viscoelastic response of the impeller material to fluid stress at operating conditions. Ideally, the inverse shape will "pass through" the design shape at a prescribed time as described previously for the modified NACA 66 simulations. The FSI solver and the inverse technique have been demonstrated in previous chapters of this dissertation for a single fin subjected to quasi-steady flow. This chapter discusses the application of the FSI solver and the inverse technique to a rotating viscoelastic impeller comprised of two blades. The impeller is prescribed to rotate at 500 rev/s and the flow is treated as incompressible and laminar with a flow rate of 5 L/min. The impeller has an outside diameter of 5.9 mm, and operates in a 6.0 mm-diameter pipe, yielding a radial tip clearance of 0.05 mm. The fluid (blood) is approximated as Newtonian (see [35] for discussion of why a Newtonian model is acceptable for blood) with a kinematic viscosity $\nu = 3.302\text{x}^{-6}$ m^2/s and density $\rho = 1060$ kg/m^3. The Reynolds number based on chord and tip relative velocity is approximately $14,700$, which is far less than the critical Reynolds number of $200,000$. Therefore, the flow is modeled as laminar.

The purpose of this analysis is to demonstrate the FSI solver and the inverse analysis approach for the rotating impeller and thus a mesh convergence study is not performed. The creation of the fluid and structural meshes is described next, followed by the simulation results. Note that the pump impeller described in

Chapter 1 consists of two blade rows with two blades per row. The impeller used in this analysis consists only of the upstream blade row which exhibits the largest blade deformation.

7.1 Mesh Generation

Mesh creation for both the flow and structure domains follows a similar procedure to that used for the single fin model described in Section 5.1. Details of the mesh creation for the flow domain of the impeller model are provided next, followed by mesh creation for the impeller structure.

7.1.1 Flow Domain

The complexity of the impeller geometry, shown in Figure 7.1, made difficult the use of a structured fluid mesh having acceptable cell quality. An unstructured mesh was therefore created in GridGen. The mesh was created as a cyclic symmetric mesh with two-times symmetry (Figure 7.2) and consists of 500,178 tetrahedral cells. The mesh was then rotated about the symmetry axis to create a full mesh. The use of a full mesh eliminated the need for multi-point constraints and cylindrical coordinate systems in the structural solver.

7.1.2 Structural Domain

The impeller mesh is created using the `bladeGen` Matlab application described previously in Section 5.1.2. The blade sections used by `bladeGen` are shown plotted on the impeller geometry (note the use of symmetry requires the creation of only a single blade model) in Figure 7.3. The coarse section points shown in this figure are sufficient to create a fairly accurate representation of the blade. However, to ensure the blade surface nodes are correctly placed, each is projected onto the blade solid body's surface using the commercial software Femap[139].

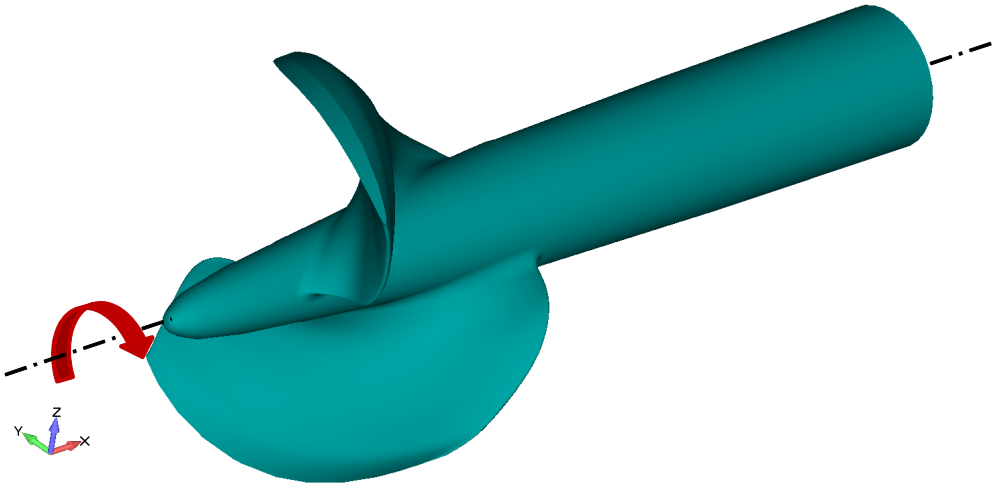

Fig. 7.1. Solid model of viscoelastic impeller

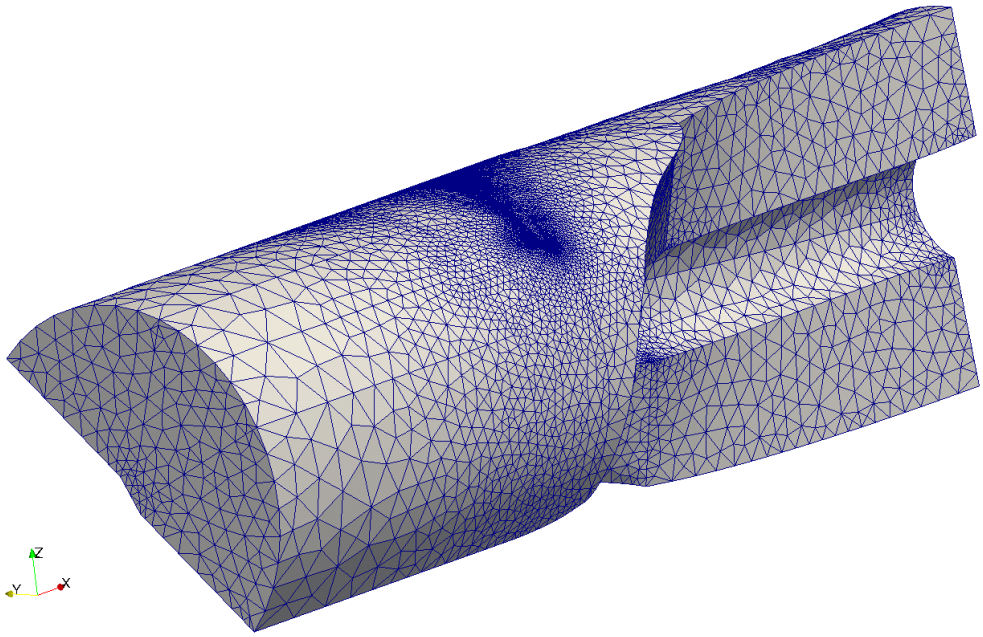

Fig. 7.2. Unstructured cyclic-symmetric fluid mesh for expandable impeller pump

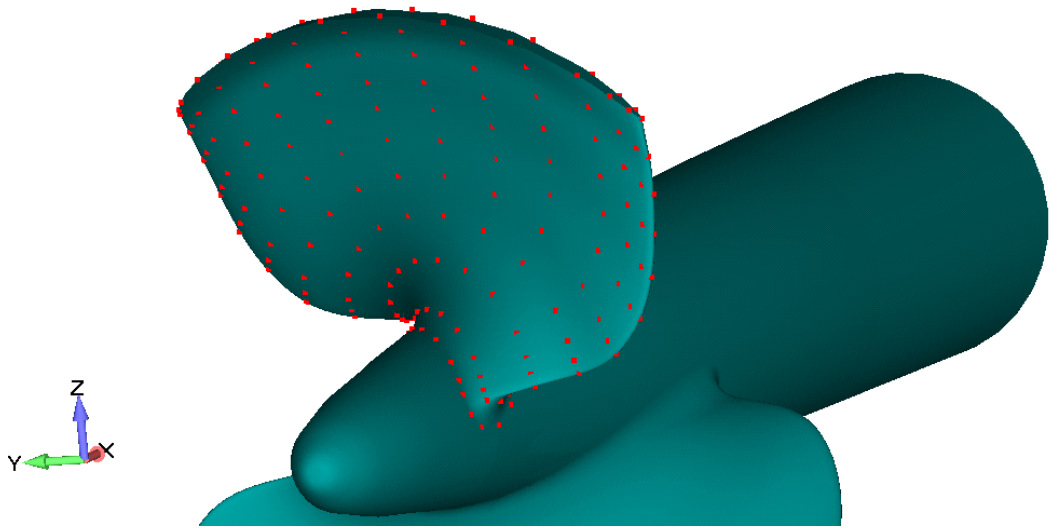

Fig. 7.3. Impeller blade section points plotted with the impeller solid model; these points are used by `bladeGen` to create the impeller FE mesh

The resulting finite element mesh of the impeller is shown in Figure 7.4. Note that the mesh was created solely by `bladeGen`, except for some hand-work to fix a few distorted elements in the hub and also to create the nose and downstream ends of the hub. The mesh shown in this figure consists of 9,676 hexahedral elements and 13,404 nodes.

The hub is comprised of an embedded tube that is much stiffer than the impeller material, which results in negligible rotation of the hub due to fluid loading on the blade surfaces. This fact enables the hub region to be eliminated from the structural model during the FSI simulations, which reduces the number of wetted surfaces that must be searched and interpolated for mesh motion. The final impeller model consists of two blades, each constrained at their roots as shown by Figure 7.5. This model is comprised of 4,896 elements and 7,696 nodes.

Note that the blade deformation due to centrifugal force is accounted for in the analyses. The deformations resulting from the force are very small as shown

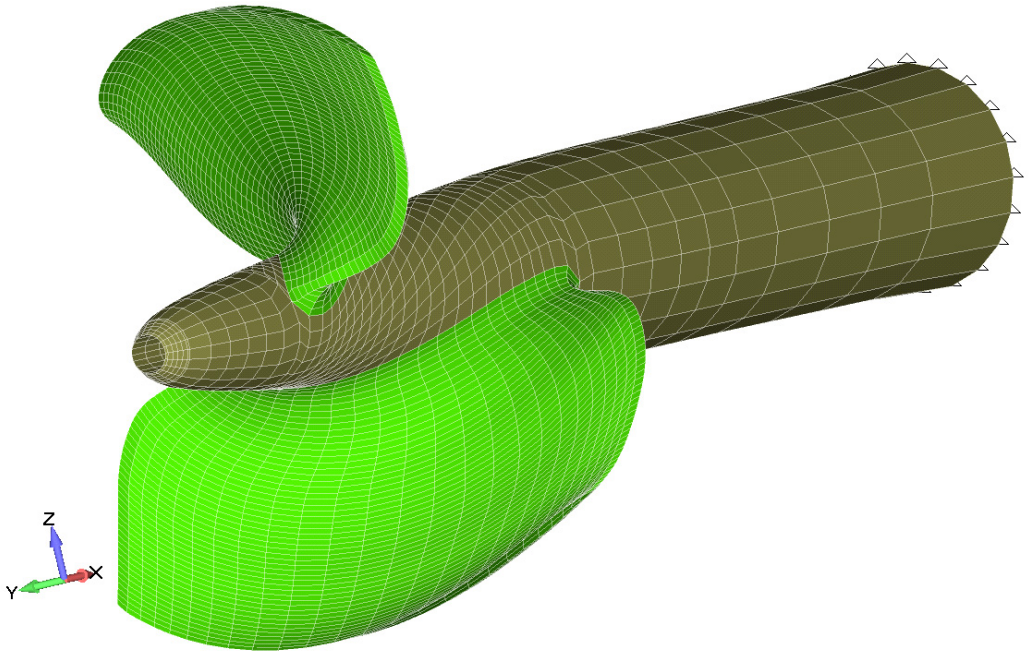

Fig. 7.4. Impeller finite element mesh showing the blade and hub section elements by distinct colors; the model is constrained at the downstream end of the hub

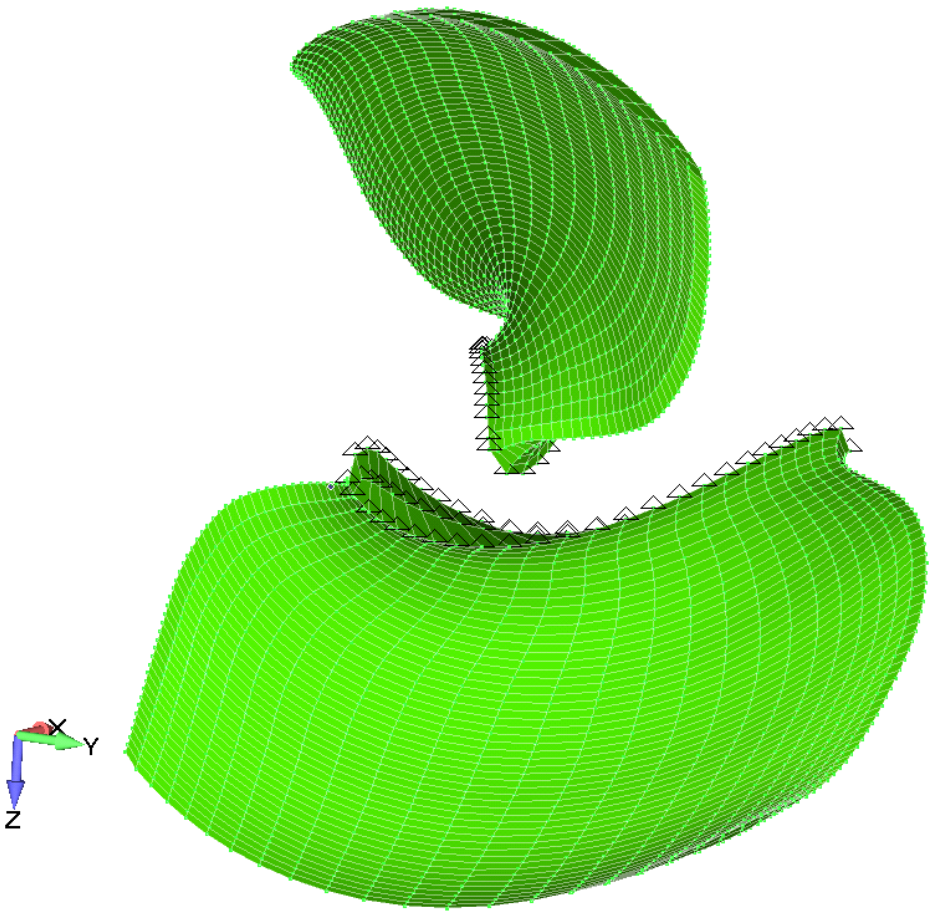

Fig. 7.5. The final impeller finite element mesh showing the blades constrained at the roots

by the displacement contours in Figure 7.6. These displacements are a result of only the centrifugal body force application to the impeller for a period of $1,000$ s.

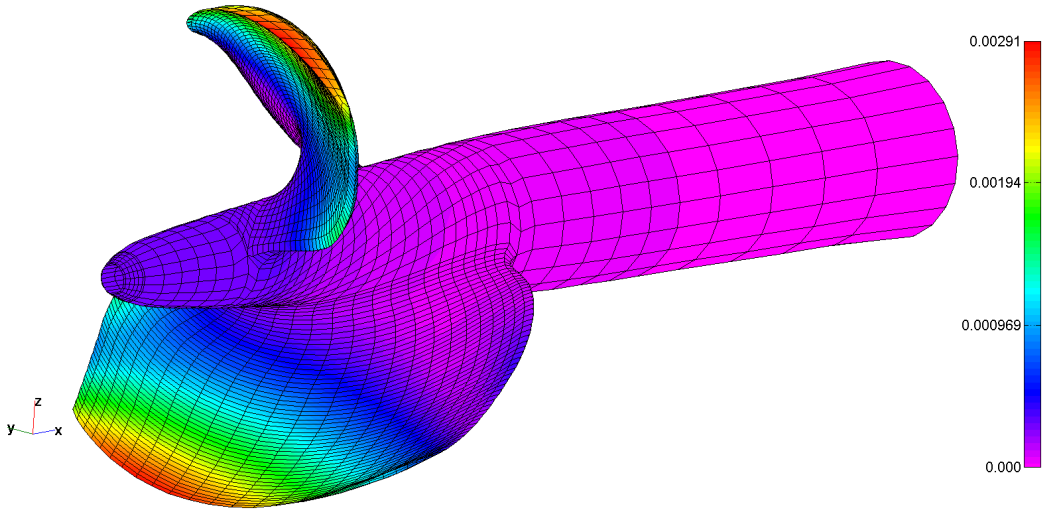

Fig. 7.6. Impeller deformations (contoured by displacement magnitude as a percentage of the impeller diameter) due solely to centrifugal force when rotating at $30,000$ rpm

7.2 Results

7.2.1 Rigid Impeller Simulation

Prior to showing blade deformation results, rigid-impeller results are presented in order to provide a qualitative check that the flow solver is functioning properly and to describe the flow boundary conditions. The prescribed flow rate of 5 L/min is introduced to the system as a uniform x-direction flow at 2.947 m/s. The velocity inlet and the stationary pipe wall velocities (in the absolute reference frame) are shown in Figure 7.7. The impeller (blades and hub) are prescribed to

rotate about the x-axis at 500 rev/s. Figure 7.8 shows the boundary velocity magnitude in the absolute frame of reference. The resulting pressure distribution over the pipe wall and impeller surfaces is shown in Figures 7.9 and 7.10, respectively. The net head rise across the impeller is 43.4 mmHg based on the impeller thrust.

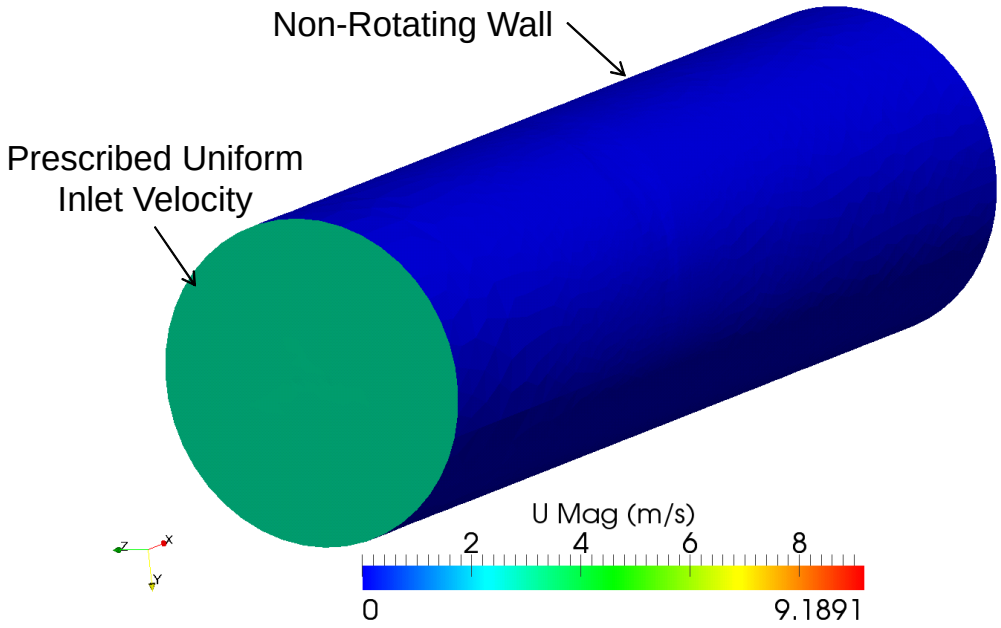

Fig. 7.7. Inlet and pipe wall velocity magnitude in the absolute reference frame for the impeller simulations

7.2.2 FSI Simulation

With a rigid-impeller flow solution that provides qualitatively correct results, a flexible impeller is introduced using the same viscoelastic material used previously. A tightly coupled FSI simulation is employed using the `MRFSimpleFsiFoam` described previously using ten coupling subiterations. The mesh motion solver employed for this analysis is the custom RMF technique described in Section 2.4. This mesh motion solver enables the simulations to take place in parallel. Sixteen

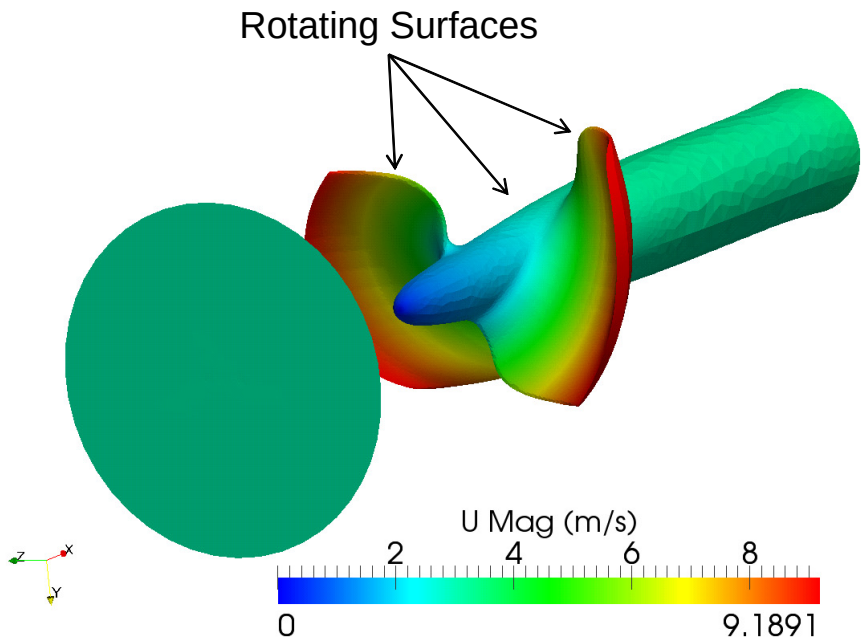

Fig. 7.8. Impeller and hub velocity magnitude in the absolute reference frame for the impeller simulations

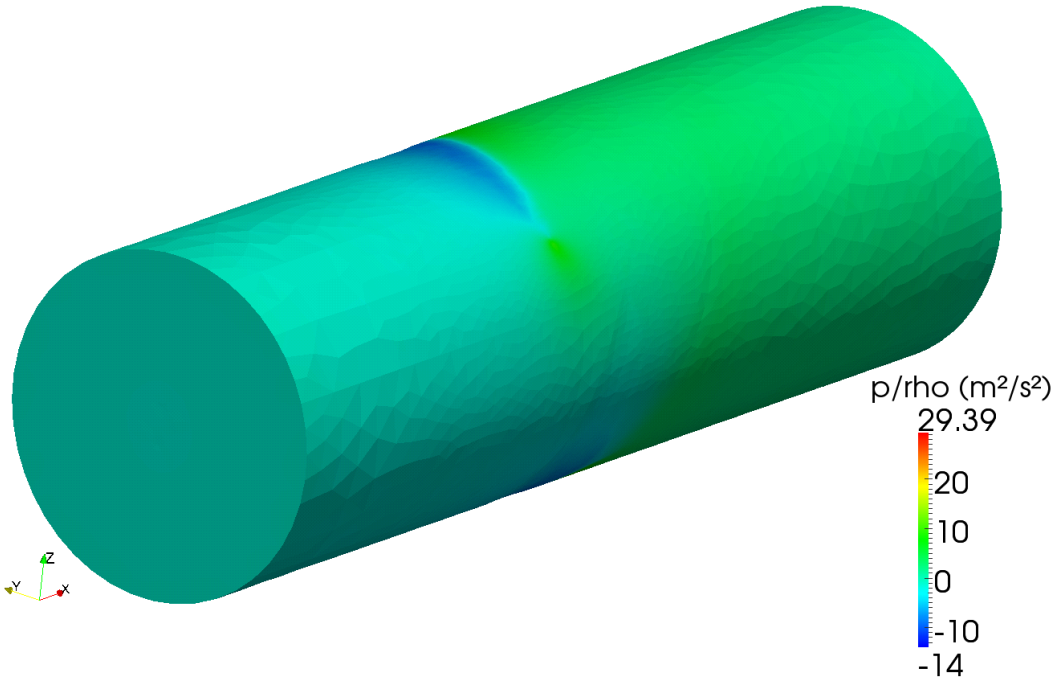

Fig. 7.9. Inlet and pipe wall pressure magnitude for the prescribed boundary conditions and a rigid impeller

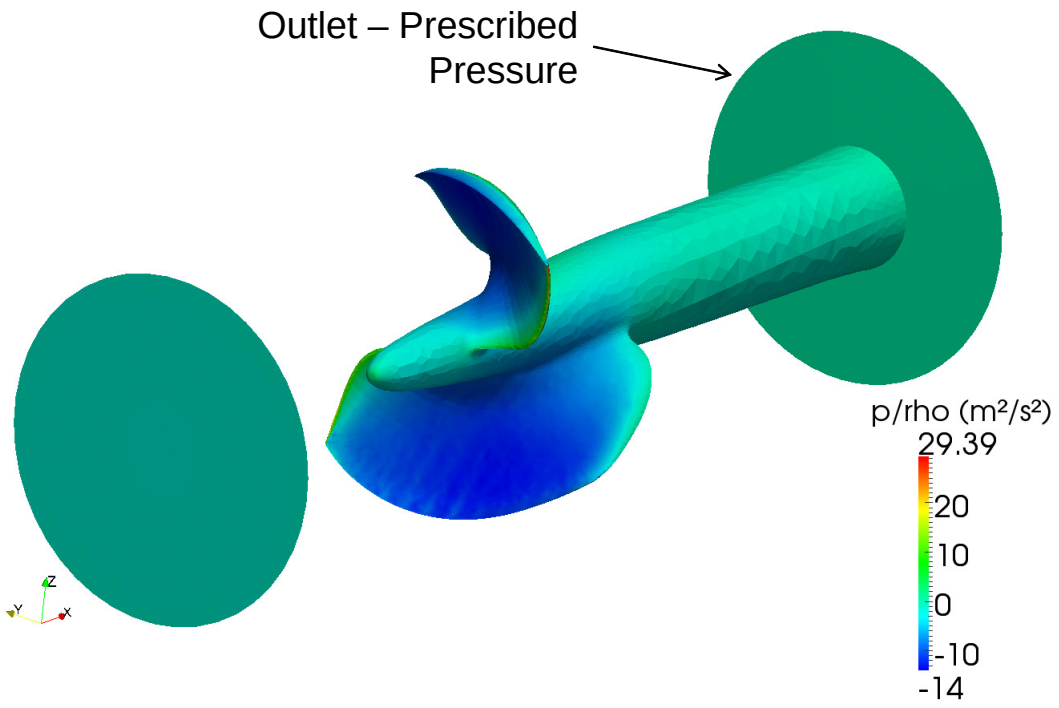

Fig. 7.10. Pressure contours on the impeller and hub for the prescribed boundary conditions and a rigid impeller

processors were used to solve the problem on the ARL/Penn State 256 processor computer previously described. The simulations take place over $1,450$ s. There is no significance to the simulation time other than it provides sufficient information to show trends in the results.

Figure 7.11 shows pressure contours on the flow boundaries and some streamlines contoured by flow speed for various times during the simulation. The $t = 0$ s results occur just prior to the impeller deforming and the largest impeller deformation per solution time step occurs during the initial deformation (the elastic part of the deformation). Except for changes during the initial elastic deformation, there is little change in the flow character during the viscoelastic relaxation.

The impeller deformation at 1450 s is shown in Figure 7.12 contoured by total displacement magnitude and in Figure 7.13 showing contours of radial deformation. The maximum radial deformation occurs at the blade tip and represents an increase in the tip clearance by approximately 78.4%. A plot of the impeller minimum tip clearance, normalized by the design tip clearance, versus time is provided in Figure 7.14. The pump head rise follows a trend similar to that of the tip clearance and is shown in Figure 7.15.

The decrease in pump performance can be offset by accounting for the impeller deformations during the design phase. This is demonstrated in the next section.

7.2.3 Inverse Simulation

The inverse analysis for the expandable impeller pump employs the same procedure used previously for the modified NACA 66 fin. The first step in the procedure is to compute the impeller loads for a rigid impeller in the desired design configuration. These loads are then applied to the impeller for the inverse structural analysis. Unlike the fin analysis where the flow speed is increased linearly from zero to the maximum value, the fluid load is applied instantaneously in the

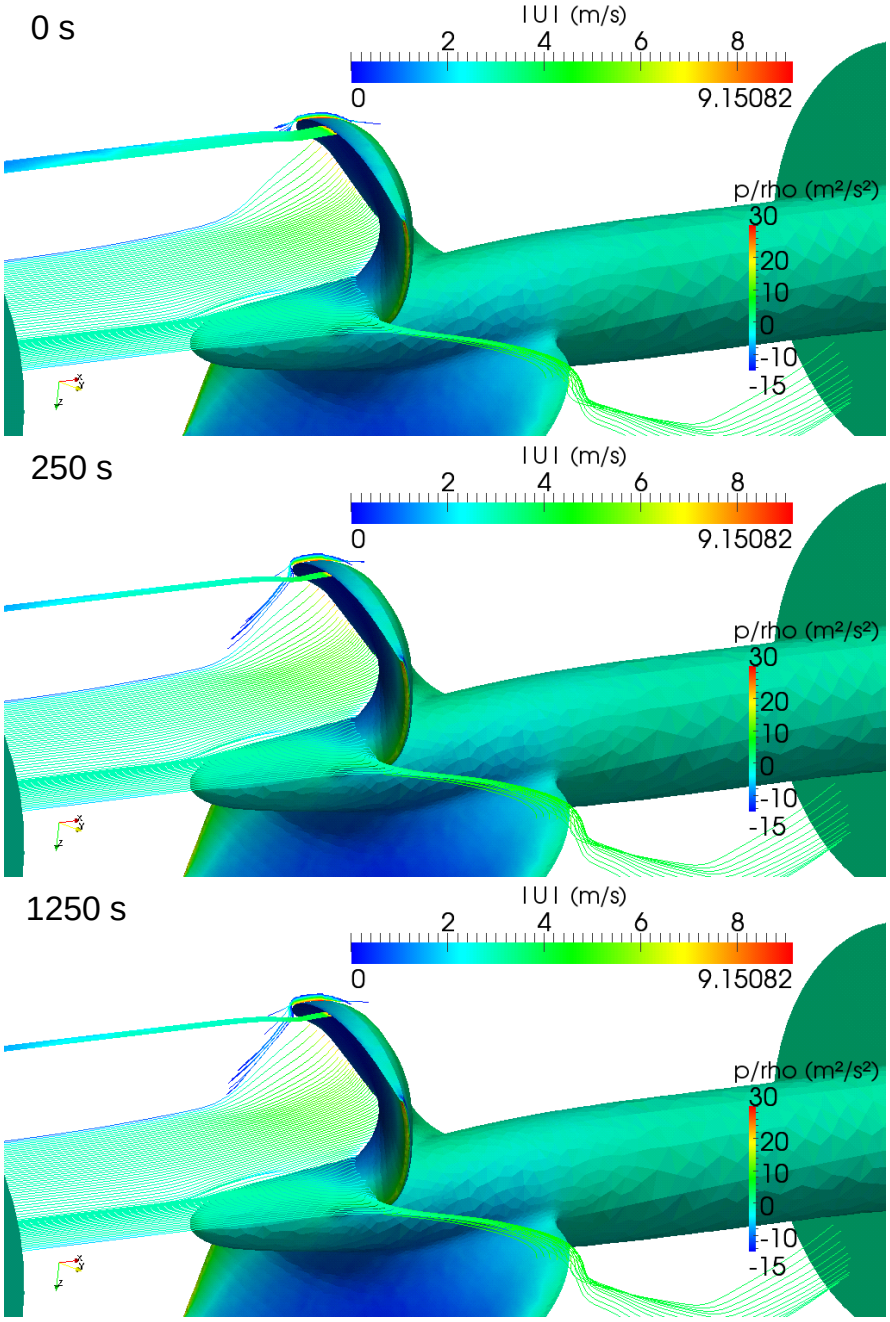

Fig. 7.11. Impeller pressure contours on the flow boundaries for select simulation times from 0 s (just prior to the impeller deformation) to 1250 s; streamlines are shown colored by velocity magnitude

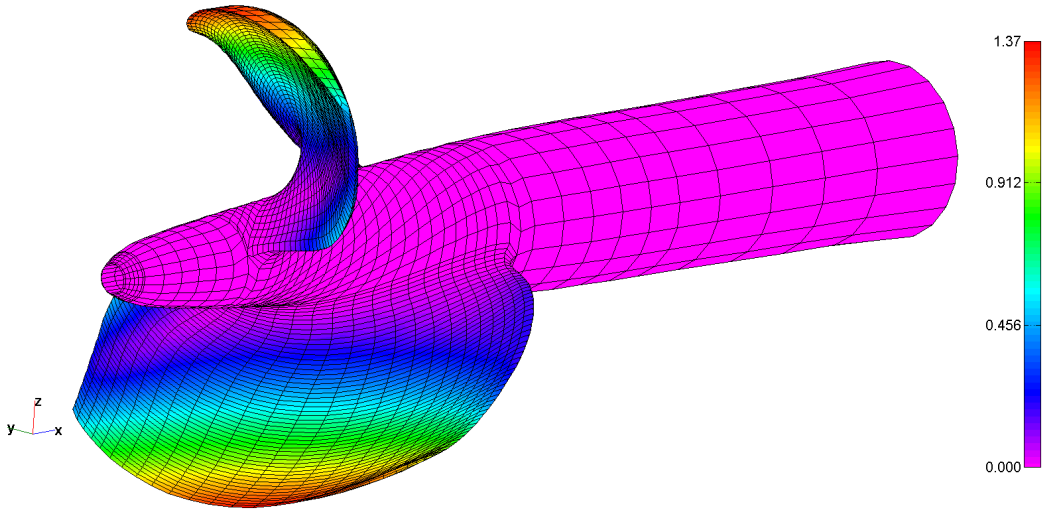

Fig. 7.12. Impeller deformation at 1450 s contoured by displacement magnitude as a percentage of the impeller diameter

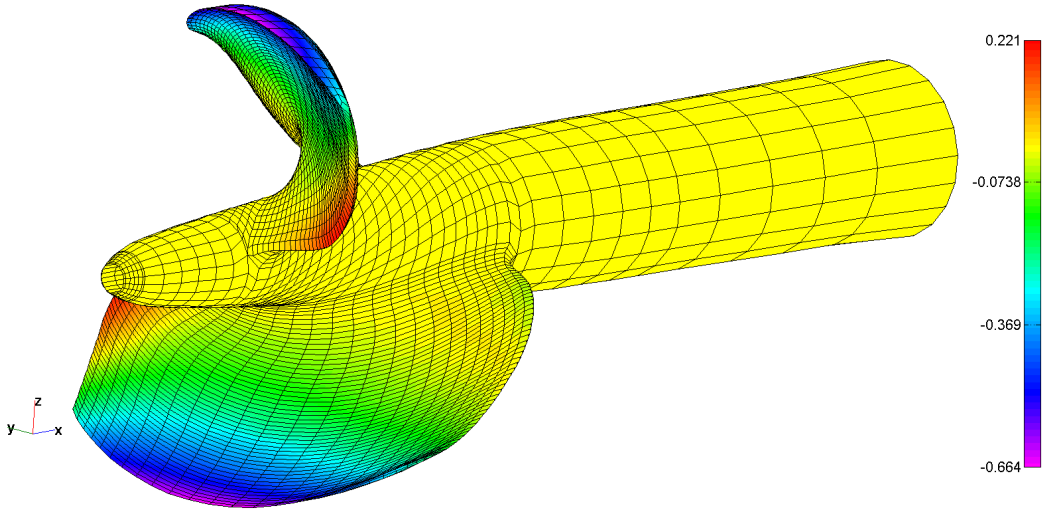

Fig. 7.13. Impeller deformation at 1450 s contoured by radial displacement as a percentage of the impeller diameter

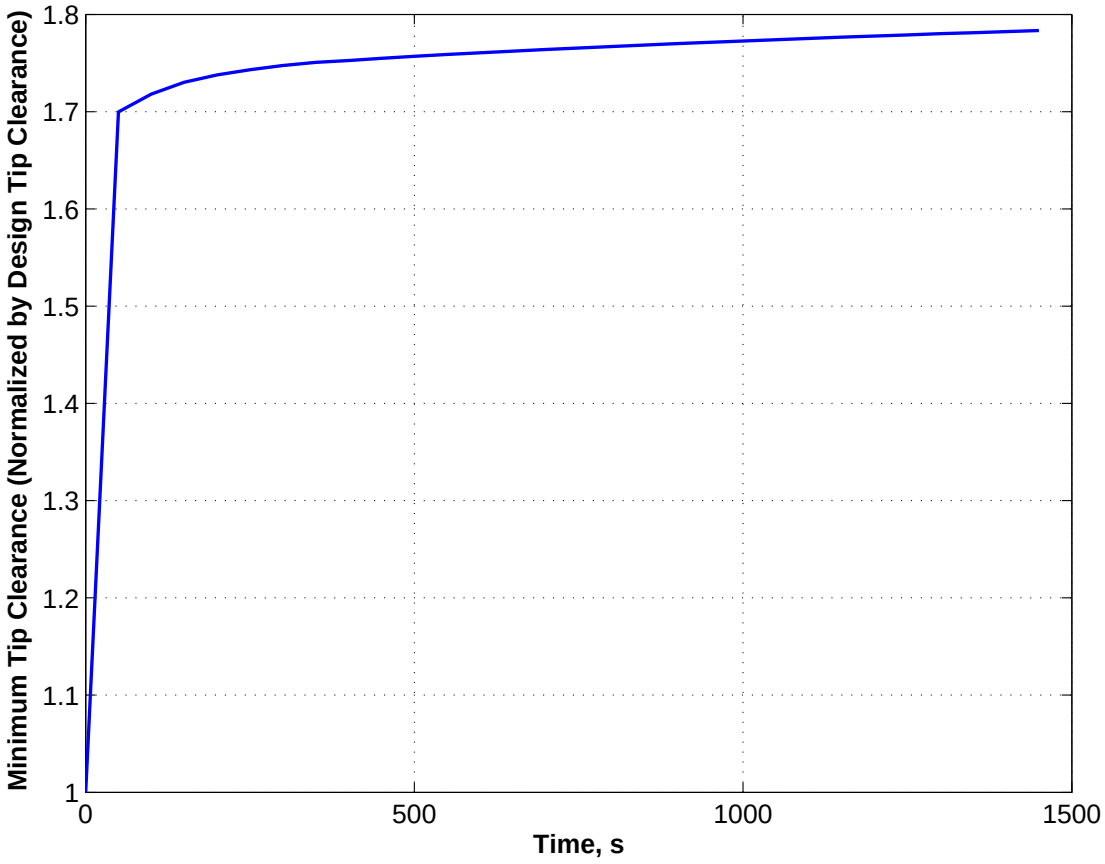

Fig. 7.14. Impeller tip clearance change with time

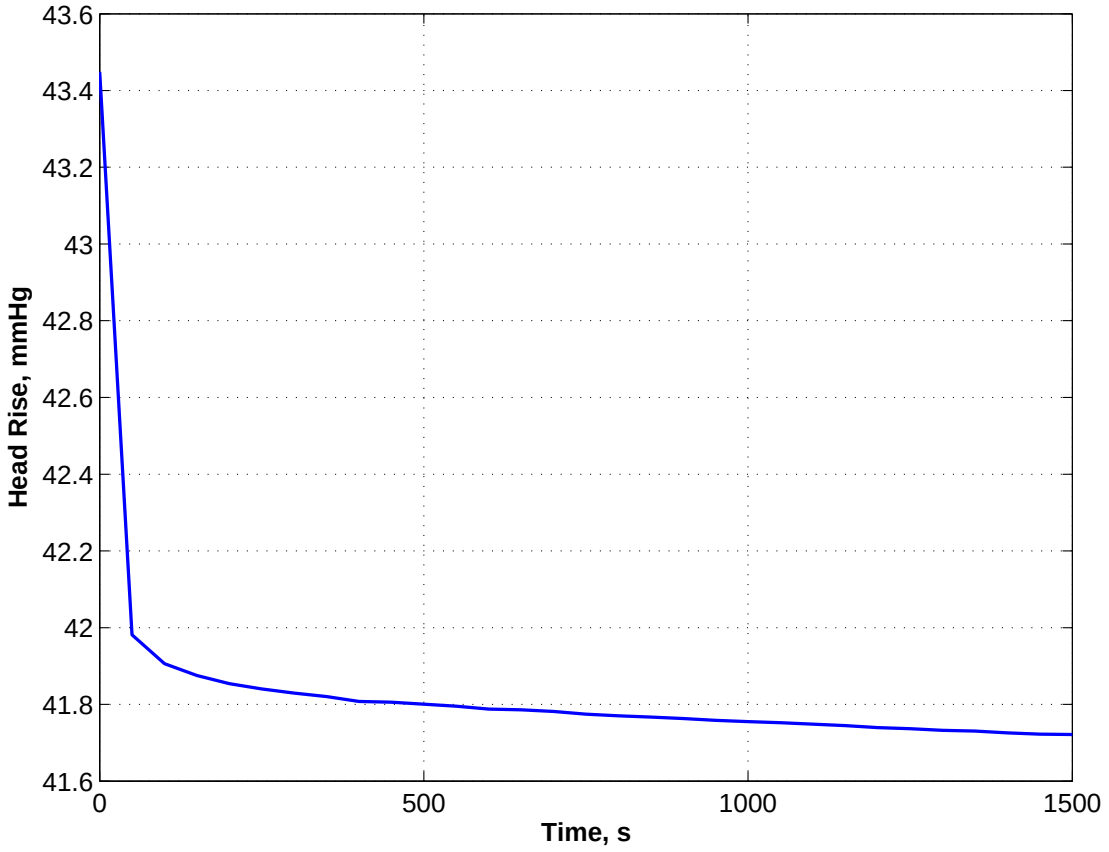

Fig. 7.15. Pump head rise variation with time due to impeller blade deformation; head rise at $t = 0$ s represents head rise prior to any impeller deformation

impeller FSI simulations. The inverted structural shape is shown in Figure 7.16 color-contoured by displacement magnitude. This shape corresponds to application of the design loads for a time period of 100 s. There is no significance to the use of 100 s here, other than it allows the initial elastic deformation and some viscoelastic relaxation and does not require substantial computer time to solve. Figure 7.17 shows the same inverted shape, but color-contoured by radial displacement. The maximum radial deformation shown in this figure occurs at the blade tips and represents a reduction of the design-shape tip clearance by approximately 57%.

The error in the inverted shape shown in Figure 7.18 indicates the inverted shape computed by `ifeanl` results in a design shape that differs from the intent with an error approximately four orders of magnitude less than the maximum inverse deflection amplitude. The error of the inverted shape is about three orders of magnitude lower than the manufacturing tolerance for such an impeller.

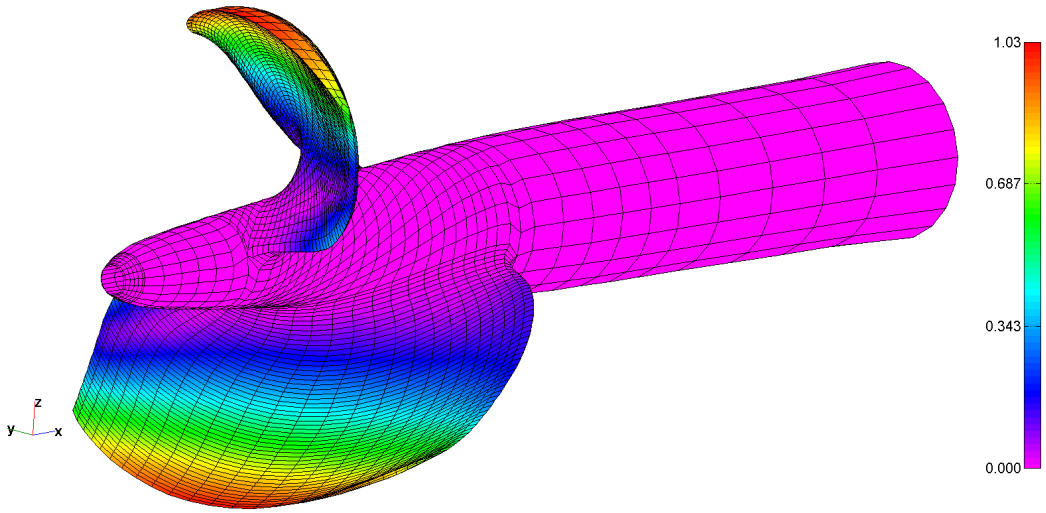

Fig. 7.16. Pump impeller inverted shape, color contoured by displacement magnitude as a percentage of the impeller diameter

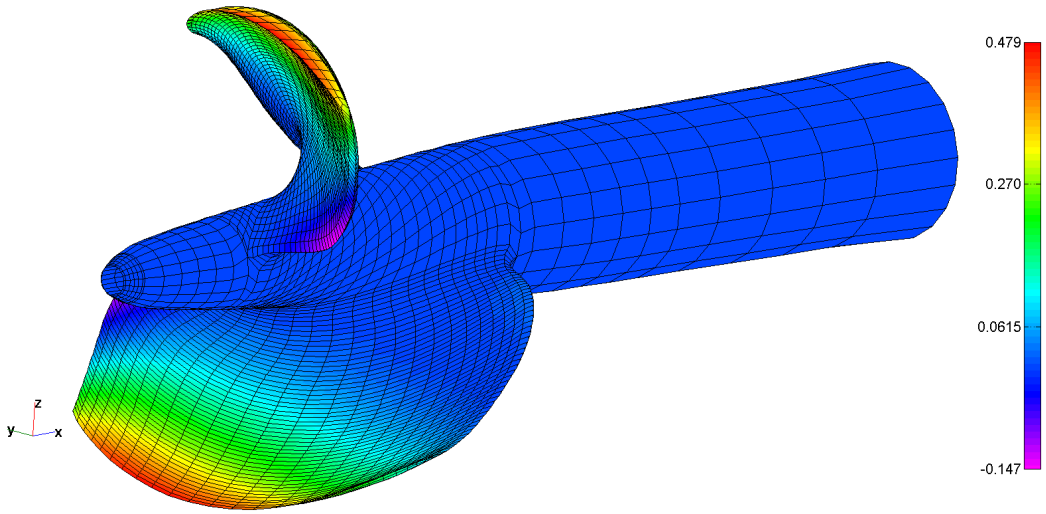

Fig. 7.17. Pump impeller inverted shape, color contoured by radial displacement as a percentage of the impeller diameter

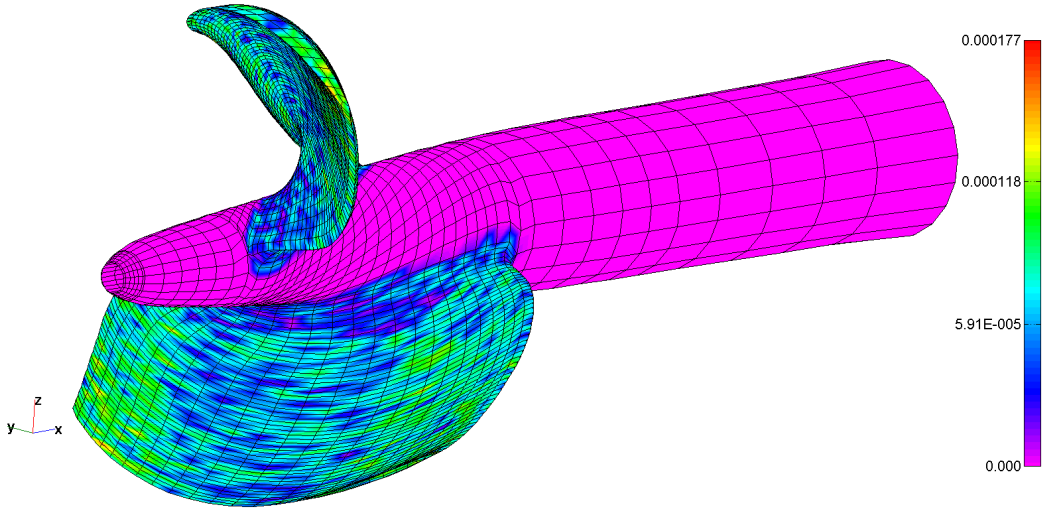

Fig. 7.18. Distance, as a percentage of the impeller diameter, from the design shape to inverted shape deformed by the design loads; the displacements shown here indicate an inverse error approximately three orders of magnitude smaller than the maximum inverse deformation shown in Figure 7.16

Next, the inverted shape is subjected to fluid loads using the FSI solver and simulated for 150 s, which is long enough to determine if the inverse shape deforms into the desired design shape at 100 s and shows how the impeller deforms shortly after this time. Comparisons are made between nodes of the inverse model and the design shape model at 100 s. The maximum distance between all nodes is shown in Figure 7.19 for various number of FSI subiterations. As shown by this figure, three or more sub-iterations provide nearly indistinguishable results. However, ten sub-iterations are used for all results reported in this chapter. Location errors are reported for the LE and TE points at 100% span in Figures 7.20 and 7.21, respectively. The net blade force for a single blade (summing over both blades would cancel all loads but the axial load because of the 2 x cyclic symmetry) is shown in Figure 7.22.

The minimum nodal location error of Figure 7.19 represents the time at which the inverted geometry best matches the design geometry. The minimum in this plot occurs at 95 s which differs slightly from the intended target time of 100 s. The location error has a maximum value for all nodes at this time of $6.823\text{x}10^{-7}$ m, which is 0.01% of the impeller design diameter and is more than an order of magnitude less than typical manufacturing tolerances for this type of part. However, the LE tip location error reaches a minimum at approximately 55 s while the TE tip reaches a minimum location error at 125 s. These discrepancies are attributed to differences between the blade loads used in the inverse FE analysis to the blade loads that actually occur during the simulation of the inverted shape, which is shown in Figure 7.22 for all load components and again in Figure 7.23 for the load magnitude and the load magnitude used during the inverse analysis. It is evident from this later plot that the net blade load during the inverse FE analysis differs from the blade load during the inverse FSI simulation. The slight geometry differences do impact the pump head rise, but not substantially. Figure 7.24 shows

a plot of the pump head rise versus time for the inverted impeller. The head rise in this figure at 95 s is only about 0.3% lower than the design shape head rise.

7.3 Conclusions

The FSI simulation tools developed and validated here are applied to the expandable impeller pump to compute the blade deformations and pump performance versus time. The performance decreases as the blade deformation increases the tip clearance and changes the blade incidence angle. To account for this deformation, the inverse analysis technique developed and implemented here for viscoelastic materials is applied to calculate an inverted shape of the impeller. Recall, the purpose of the inverse shape is to account for elastic and viscoelastic blade deformations due to the applied fluid stresses at operating conditions. The inverse shape corresponds to the manufactured shape of the impeller such that it will "pass through" the impeller design shape at a prescribed time. The FSI simulations of the inverted shape demonstrate the utility of the technique and show the pump performance and impeller shape closely match that of the design configuration, but slightly before the target time. Some shape discrepancies do occur, but it is speculated that these could be minimized by iterating on the load history employed by the inverse FE analysis.

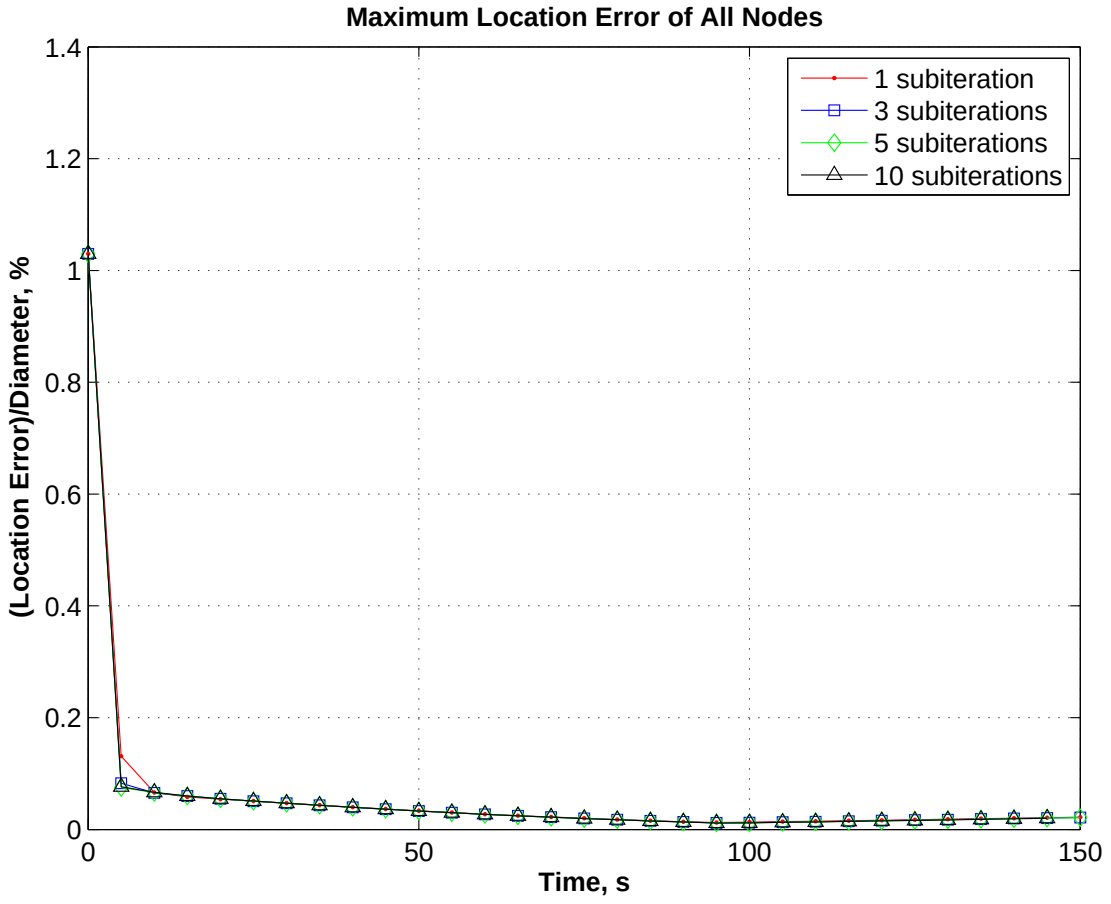

Fig. 7.19. Location error for the inverted impeller shape after being subjected to fluid loads for 100 s; all nodes of the impeller FE model are considered; results reported for various number of FSI subiterations

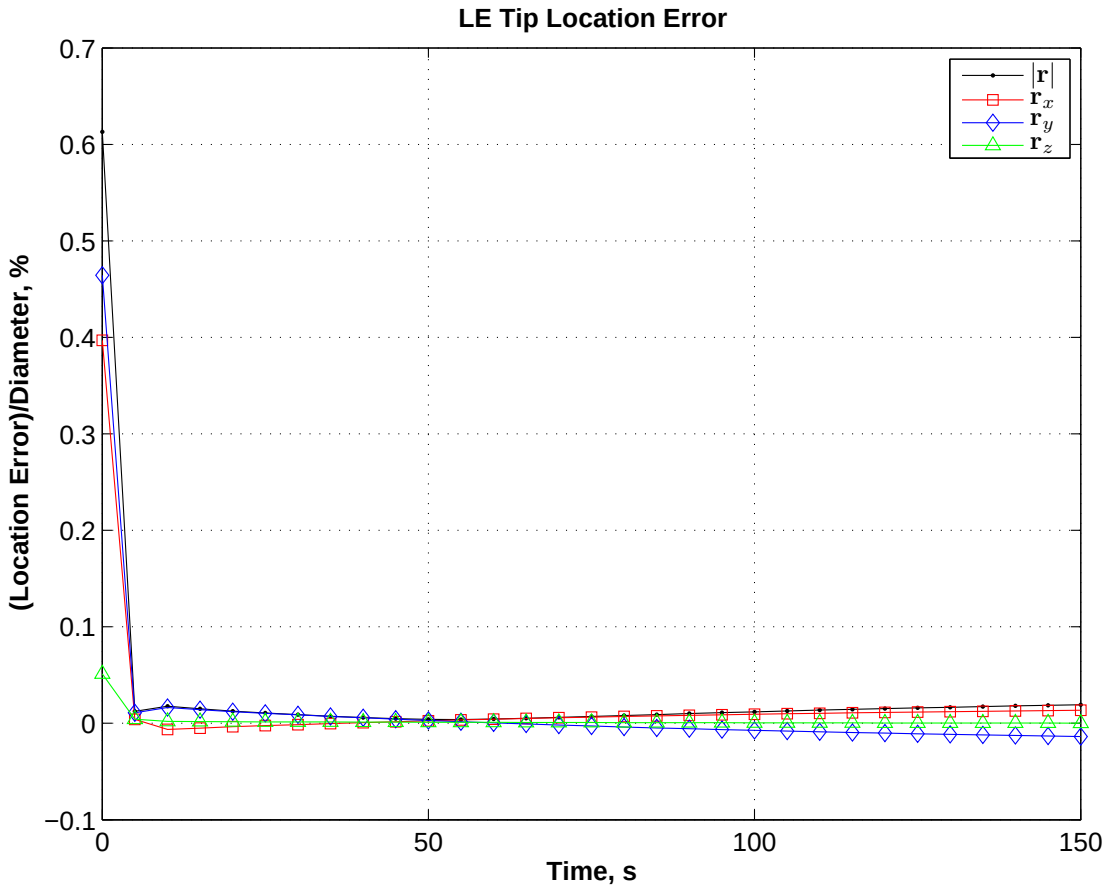

Fig. 7.20. Location error for the inverted impeller shape after being subjected to fluid loads for 100 s; leading edge tip nodes of the impeller FE model are considered

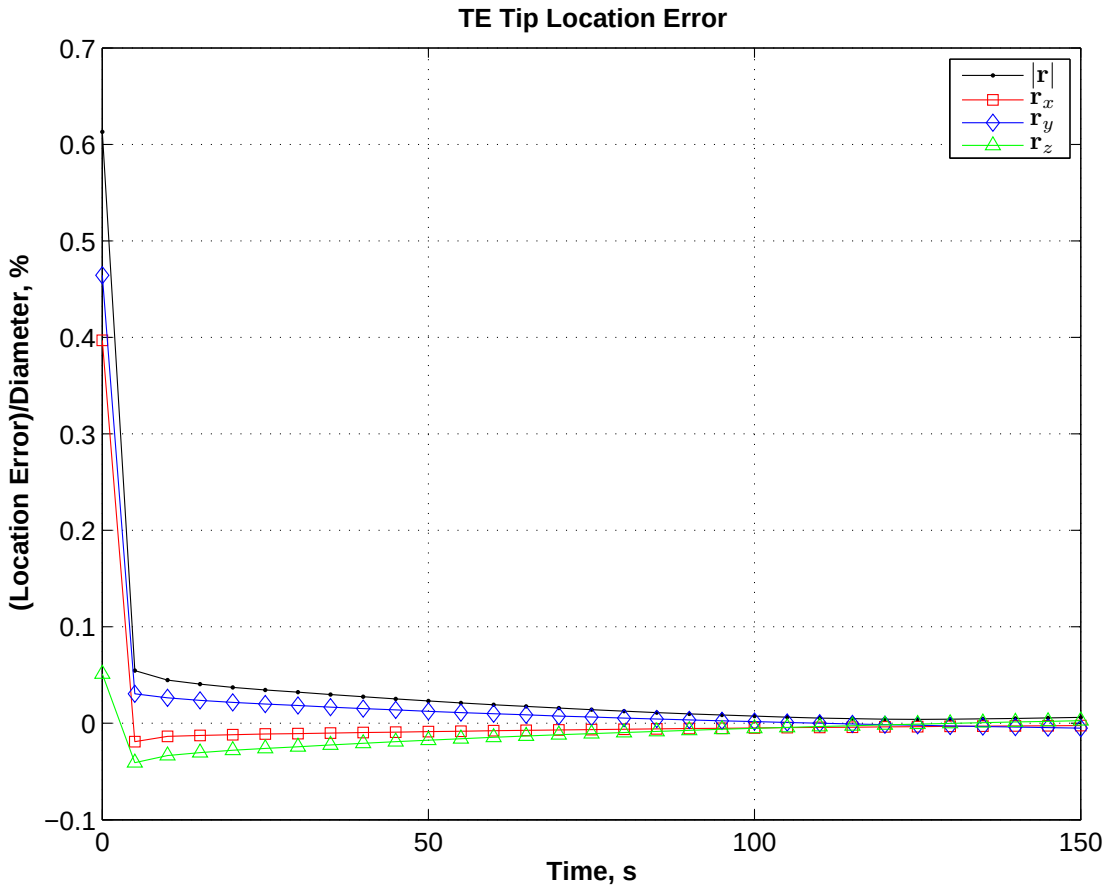

Fig. 7.21. Location error for the inverted impeller shape after being subjected to fluid loads for 100 s; trailing edge tip nodes of the impeller FE model are considered

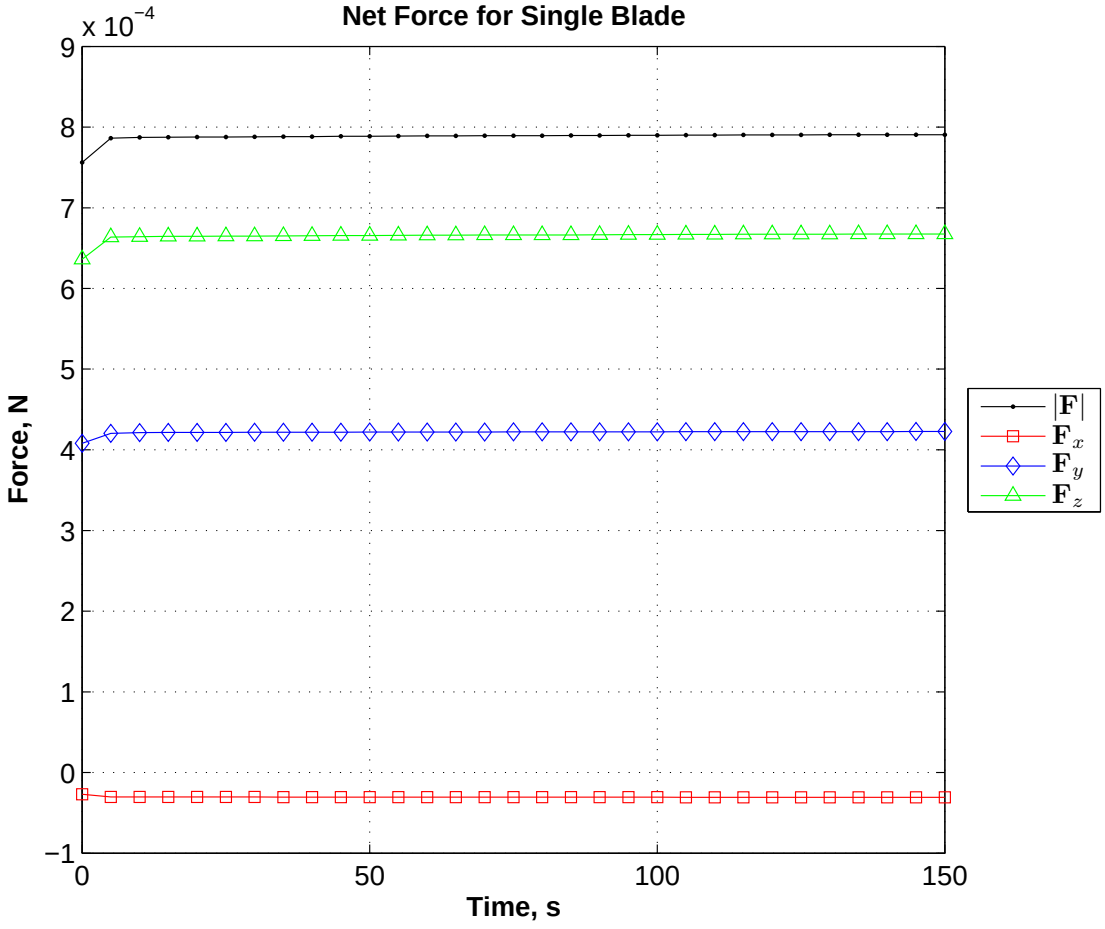

Fig. 7.22. Net blade loads for a single blade of the inverted impeller shape

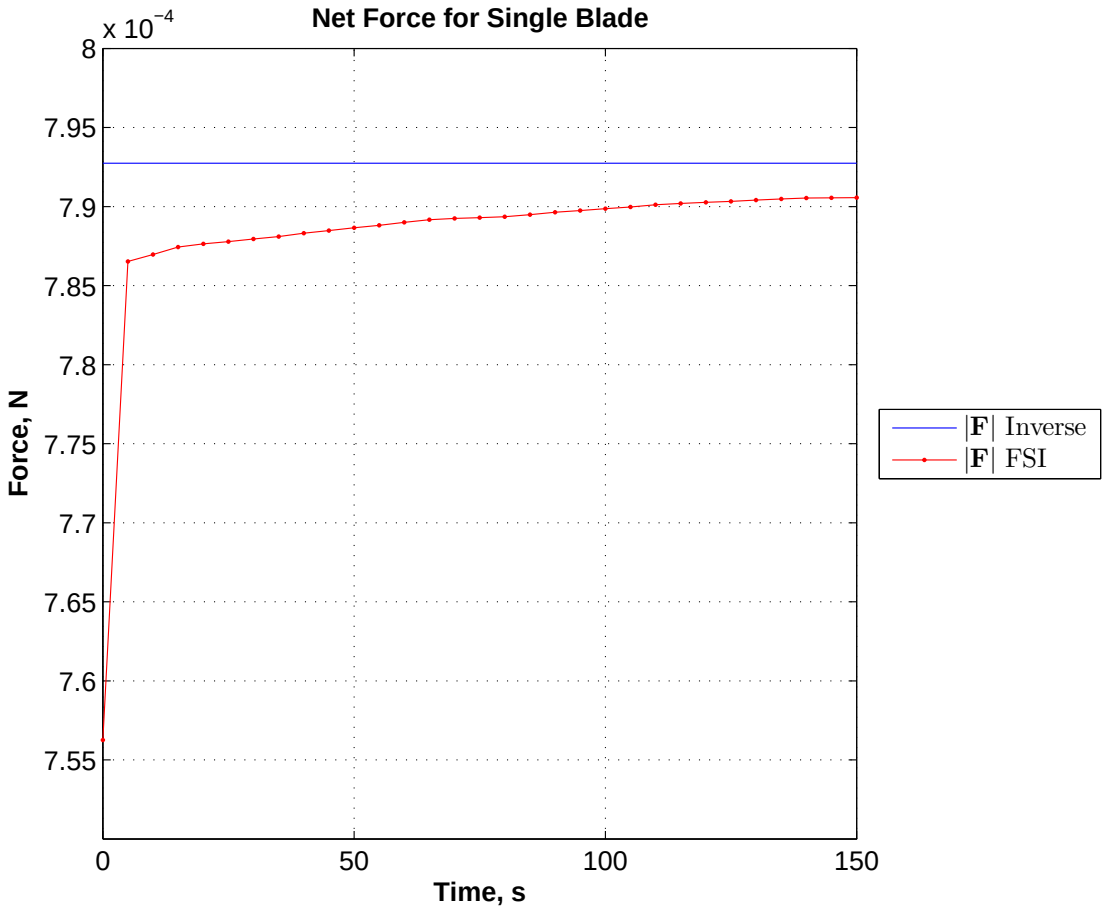

Fig. 7.23. Magnitude of net blade loads for a single blade of the inverted impeller shape

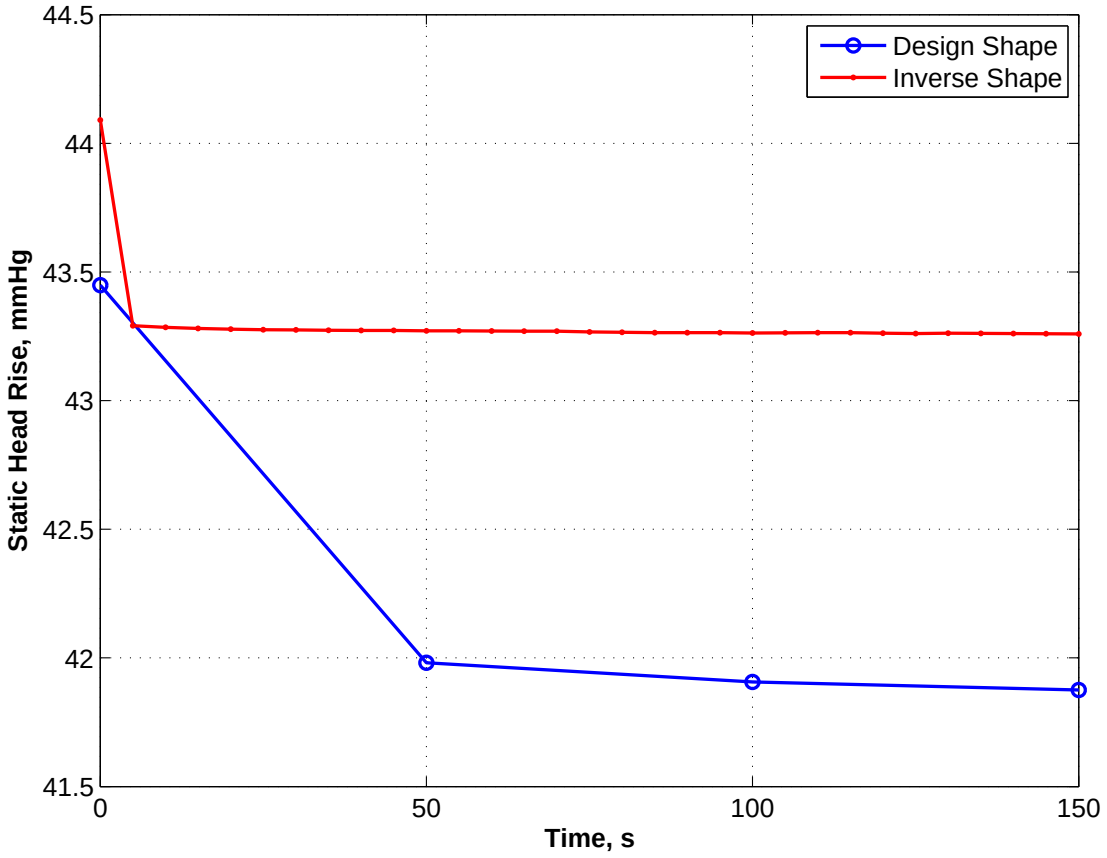

Fig. 7.24. Time-accurate pump head rise for the inverted and design-shape impeller

Chapter 8

Summary, Conclusions, and Future Work

8.1 Summary

The objective of this research was to explore the use of FSI for analyzing highly flexible turbomachinery, including a simple fin and an expandable impeller pump, and to evaluate the use of an inverse structural analysis approach to account for blade deformations during the design process. A partitioned FSI solver with capability for tightly coupled simulations was developed using OpenFOAM and author-developed mesh motion and structural solvers. Two FSI solvers were developed, `simpleFsiFoam` and `MRFSimpleFsiFoam`, for simulating fixed-reference frame FSI simulations and rotating-frame FSI simulations, respectively. The later solver differs from the former only in the inclusion of body force terms for the rotating frame of reference. `SimpleFsiFoam` was validated against water tunnel test results for a modified NACA 66 fin fabricated from Hapflex 598, which is the same viscoelastic material used by the expandable impeller pump. The validated FSI solvers were used to evaluate the inverse blade geometries for the viscoelastic material used in this research.

Substantial effort has been invested in this research to validate the FSI solver developed herein. The validation study used a modified NACA 66 fin subjected to quasi-steady flow in a water tunnel test facility. Comparisons between simulated and measured blade deformations were performed for multiple fin angle of attacks. The FSI solver simulations show good agreement with the experimental results. The viscoelastic material model used in the FSI simulations was validated

separately using existing material test data and refined with beam and fin bending tests.

A partitioned FSI solver was implemented, which means the flow and structural solvers can be maintained and advanced separately. OpenFOAM was used for the flow solver because it is freely available and has existing mesh motion capability. A structural FE solver was developed specifically for this research and integrated with OpenFOAM to create a single executable for the FSI solver. An interface class was implemented to perform the information exchange between the solvers while enabling independent solver updates so long as the class member function definitions of each solver do not change. This enables efficient exchange of interface information, something that is not currently possible with existing commercial software. The interface coupling between the flow and structural solvers to communicate stress and displacement supports disparate meshes at the interface. Fluid forces are computed by the flow solver and applied to structures to ensure consistent forces exist on both sides of fluid/structure interfaces. Displacement communication from structures to fluids occurs by interpolation of structural displacements by the structural solver to interface vertex locations. Development of interface coupling in this manner was facilitated greatly by having source code for each solver.

Motion of fluid meshes represents an integral part of the partitioned approach to FSI modeling. OpenFOAM's Laplace face decomposition solver was used initially for this research, but was later replaced by a custom mesh motion solver called the Radial Motion Function solver. The custom solver supports parallel execution of the solver and enables sliding constraints on cylindrical boundaries, both of which are not currently supported by the Laplace face decomposition solver. Additionally, the RMF solver computational requirements are far less than those of the Laplace face decomposition solver because there is no need to perform a matrix solution for each mesh motion. A second custom mesh motion solver was

implemented for large deformations in the fin simulations. This solver uses an auxiliary finite element mesh that deforms based on interface motion and interpolates displacements to compute motion of the underlying fluid mesh. This solver has not been fully evaluated and therefore should be further investigated for future endeavors.

An inverse finite element structural solver was implemented to enable the calculation of inverse shapes for viscoelastic components. Other researchers have implemented inverse FE solvers, but their work is not applicable to viscoelastic materials where response is dependent upon load history. Rather than implementing a solver specific to viscoelastic materials, a general solver was implemented that makes use of the underlying linear incremental formulation of the large deformation FE solver. The objective of this solver is to determine an undeformed (unknown) shape such that prescribed loads cause the unknown shape to deform into the existing (known/design) shape. The approach taken here to solve this problem uses a formulation that incrementally changes the reference configuration until the current (design) configuration can fully support the specified load. The solver is validated for all inverse simulations performed in this research and has demonstrated its accuracy for all models analyzed here.

Inverse analyses for the modified NACA 66 fin are performed for three angles of attack. The results show the inverted shapes deform into the design shape, but the time at which this occurs is not accurately predictable using only an estimate of the load-history during the model inversion. It is recommended that an iterative approach be employed to accurately define a load history for the inverse FE analysis.

Finally, simulations are performed for an expandable impeller pump to demonstrate the time-accurate pump performance changes caused by the impeller deformations. Plots of tip clearance and pump performance versus time are presented. The inverse analysis technique is applied to the pump and demonstrates

the ability to recover much of the lost performance due to impeller blade deformations. The resulting blade shapes and pump performance match well the design intent, but it is speculated that improvements are possible through an iterative approach similar to that recommended for the fin inverse simulations.

8.2 Conclusions

This research has demonstrated the following:

1. the time-accurate response of viscoelastic fins and turbomachinery subject to fluid flow forces can be accurately modeled using a partitioned FSI solver with subiterations to tightly couple the flow and structural response,

2. with the addition of body force terms for centrifugal and Coriolis acceleration, the same FSI solver can simulate the time-accurate performance of a pump comprised of a flexible, rotating impeller

3. inverse finite element analysis is possible for viscoelastic materials using the incremental formulation for large deformations, and

4. the inverse shapes obtained from the FE analysis technique deform into the design shape but the time at which this occurs deviates slightly from the prescribed time.

8.3 Future Work

While the FSI solver developed here was intended for quasi-steady simulations, it was implemented with the intent of applying it to unsteady simulations in the future. For instance, the solver's interface coupling scheme was developed to provide an energy-conserving exchange of forces and displacements between two disparate meshes with the intent of using the coupling scheme for fully unsteady simulations without the burden of the added-mass instability that is described in

the literature. Also, dynamic modeling capability has been implemented in the structural solver. A natural extension of the present work is therefore to validate the FSI solver for unsteady simulations. Included in this work should be a characterization of the interface coupling scheme developed here. The effects of discretization between donor and receiver solvers should be evaluated for accuracy and energy conservation. Effects of interpolation order of the structural solver should also be considered in this study. Different interpolation order elements would result in different interpolated displacement values at the same physical location on the interface. This should be quantified. Also, higher-order surface definition with, for example, a NURBS object could provide additional functionality at the interface, especially if the fluid mesh were to be remeshed to accommodate large deformations of the structure. Lastly, the structural solver would require validation for dynamic response simulations.

Motion of the fluid mesh has proved to be an area requiring additional development. Future work should formally compare existing mesh motion approaches and should further evaluate the overlayed mesh motion approach implemented here. The utility of the overlayed mesh motion is that it can range from a very coarse mesh that overlays only a small portion of the total flow domain with the cost of poor resolution, or it can coincide identically with the fluid mesh but at great computational expense. Alternatively, an overset mesh CFD solution [13, 100] could be used wherein the fluid mesh motion would occur only on the overset mesh without modification of the background mesh. Future work should extend the current FSI solver with the development by Boger et al. [13] to evaluate the use of overset meshes for FSI simulations.

Finally, the inverse finite element solver implemented for this research represents a far simpler approach relative to that used by other researchers. While the inverse solver performed well for all cases evaluated here, a more formal investigation of the solver is warranted. For example, convergence characteristics of

the solver relative to the rigorous inverse formulations used by others should be characterized.

Appendix A

Sample Structure Input File

```
** Comments identified by **
** Keywords identified by *
** Case insensitive, comma-separated entries
** Everything above *step keyword is model data section (defines nodes,
**  elements, materials, etc.)
** Step and beyond is considered history data
*NODE
       1,    0.050315,    -0.00289,          0.
       3,    0.049251,     -5.6E-5,          0.
      58,    0.024818,   -0.000783,          0.
...
    6344,    0.000654,     0.00023,      0.0996
*ELEMENT, TYPE=C3D8r, ELSET=P1
      5,      6,      7,     12,     11,    126,    127,    132,    131
      6,      7,      8,     13,     12,    127,    128,    133,    132
      7,      8,      9,     14,     13,    128,    129,    134,    133
...
   5100,   6114,   6115,   6120,   6118,   6234,   6235,   6240,   6238
** Map the elements to the material
*SOLID SECTION, ELSET=P1,
MATERIAL=M1
** Isotropic, linear-viscoelastic material
*MATERIAL, NAME=M1
```

```
*ELASTIC, TYPE=ISOTROPIC, MODULI=INSTANTANEOUS
 66.E6,      0.496,        0.
*DENSITY
     1020.
*VISCOELASTIC, TIME=PRONY
0.06, 0.,    4.13
0.20, 0.,   81.95
0.18, 0., 1610.
** Define a load ramp amplitude function
*amplitude,name=ramp2
0,0,100,1,10000,1
** History data section starts here (defines analysis type, output, and
**  boundary conditions)
*STEP, INC=400000, AMPLITUDE=STEP, NLGEOM
Force
*static
      1.0,        1000.,        0.,      1000.0
** Request output
*NODE FILE, FREQUENCY=100
    U, CF, RF
*El File, Frequency=100,position=averaged at nodes
E
** Specify boundary conditions
*BOUNDARY, OP=NEW
    1,    1
    1,    3
    3,    2
...
```

```
 6293,    3
*cload, op=new, amplitude=ramp2
 226, 1, 4.86589e-07
 226, 2, 1.58708e-06
 226, 3, -2.25339e-05
...
 6121, 3, -1.59597e-16
*END STEP
```

Appendix B

Matlab Edge Finding Routine

The following MATLAB program was used to automatically find test specimen edges during the material model validation testing.

```
function imageEdges(inFileName,timeFileName,dispFileName)

if(nargin < 1)
    inFileName = [];
end
if(isempty(inFileName))
    error('Requires at least one input argument');
end

if(nargin < 2)
    timeFileName = [];
end
if(isempty(timeFileName))
    timeFileName = sprintf('%s.time',inFileName);
end

if(nargin < 3)
    dispFileName = [];
end
if(isempty(dispFileName))
    dispFileName = sprintf('%s.disp',inFileName);
end

obj = mmreader(inFileName);

numFrames = get(obj, 'numberOfFrames');

frameRate = get(obj,'frameRate');

%videoFrames = read(obj);

% Get the reference length
```

```
h_image = imshow(read(obj,1));
h_axes = get(h_image,'parent');
set(h_axes,'nextPlot','add');
waitfor(msgbox('Pick two sets of two points for reference length (also ↩
    rotation angle)',...
    'Reference Length'));
[x,y,bttn] = ginput(1);
ptCnt = 0;
while(bttn ~= 3)
    if(~ptCnt);
        h = title('Left to pick, middle to remove, right to stop');
    end
    if(bttn==1)
        ptCnt = ptCnt + 1;
        refPt_x(ptCnt) = x;
        refPt_y(ptCnt) = y;
        h(ptCnt) = plot(h_axes,x,y,'.r');
    elseif(bttn==2)
        if(ptCnt)
            refPt_x(ptCnt) = [];
            refPt_y(ptCnt) = [];
            delete(h(ptCnt));
            ptCnt = ptCnt - 1;
        end
    end
    [x,y,bttn] = ginput(1);
end

% Get the length and its statistics
if(mod(ptCnt,2))
    error('Must have an even number of points for the reference length');
end
ind = [1:2:ptCnt];
for(ii=1:length(ind))
    refLength(ii) = norm(...
        [refPt_x(ind(ii)+1) - refPt_x(ind(ii)),...
        refPt_y(ind(ii)+1) - refPt_y(ind(ii))]);
end

ans = inputdlg('Enter reference length in m','Reference Length');
refLength_m = str2num(ans{:});
scaleFactor = refLength_m/mean(refLength);

meanRefLen = mean(refLength)*scaleFactor;
stdRefLen = std(refLength)*scaleFactor;
```

```
fprintf('Mean reference length is %g m with standard deviation of %g m\n'↩
    ,...
    meanRefLen,stdRefLen);

% Estimate the rotation angle
for(ii=2:length(ind))
    alpha_bttm(ii-1) = atan2(...
        refPt_y(ind(ii))-refPt_y(ind(ii-1)),...
        refPt_x(ind(ii))-refPt_x(ind(ii-1)));
    alpha_top(ii-1) = atan2(...
        refPt_y(ind(ii)+1)-refPt_y(ind(ii-1)+1),...
        refPt_x(ind(ii)+1)-refPt_x(ind(ii-1)+1));
end
rotAngle = cat(1,alpha_bttm(:),alpha_top(:));

meanRotAngle = mean(rotAngle);
stdRotAngle = std(rotAngle);

fprintf('Mean rotation angle is %g  with standard deviation of %g \n',...
    meanRotAngle*180/pi,stdRotAngle*180/pi);

% Pick the box to use for the edge detection
waitfor(msgbox('Pick the upper boundary for the edge detection',...
    'Upper Boundary'));
[xUpper,yUpper] = ginput(2);
waitfor(msgbox('Pick the lower boundary for the edge detection',...
    'Lower Boundary'));
[xLower,yLower] = ginput(2);

edgeBox = round([min(cat(1,xLower(:),xUpper(:))),...
    max(cat(1,xLower(:),xUpper(:))),...
    min(cat(1,yLower(:),yUpper(:))),...
    max(cat(1,yLower(:),yUpper(:)))]);

cols = [edgeBox(1):edgeBox(2)];
rows = [edgeBox(3):edgeBox(4)];

set(h_axes,'nextPlot','replace');

fidTime = fopen(timeFileName,'wt');
if(fidTime == -1)
    error('Failed to open %s',timeFileName);
end
fidDisp = fopen(dispFileName,'wt');
```

```
if(fidDisp == -1)
    error('Failed to open %s',dispFileName);
end

contSw = 1;
bwCutoff = 150;
while(contSw)
    videoFrame = read(obj,380);
    grayFrame = rgb2gray(videoFrame(rows,cols,:));
    ind = find(grayFrame<bwCutoff);
    grayFrame(ind) = 0;
    ind = find(grayFrame>=bwCutoff);
    grayFrame(ind) = 255;
    h_image = imshow(grayFrame);

    bwCutoff_ = questdlg('BW Separation Acceptable?', ...
        'BW Separation', ...
        'Yes', 'No', 'Yes');
    if(strcmpi(bwCutoff_,'No'))
       bwCutoff_ = inputdlg('Enter new bwCutoff value',...
           'bwCutoff',1,{sprintf('%d',bwCutoff)});
       try
            bwCutoff_ = str2num(bwCutoff_{:});
        catch
            bwCutoff_ = bwCutoff;
       end
        bwCutoff = bwCutoff_;
    else
        contSw = 0;
    end
end


for ii = 1 : numFrames
    videoFrame = read(obj,ii);

    grayFrame = rgb2gray(videoFrame(rows,cols,:));
    ind = find(grayFrame<bwCutoff);
    grayFrame(ind) = 0;
    ind = find(grayFrame>=bwCutoff);
    grayFrame(ind) = 255;
    h_image = imshow(grayFrame);
    h_axes = get(h_image,'parent');
    set(h_axes,'nextPlot','add');
```

```
    if(ii==1)
        % Choose the beam origin
        waitfor(msgbox('Pick the beam origin (center and leftmost point) ↩
            for image detection and rotation',...
            'Beam Origin'));
        [xOrigin,yOrigin] = ginput(1);
    end

    [boundaryPts,bottomPts,topPts] = findBoundaryPoints(...
        grayFrame,round([xOrigin,yOrigin]));

    %plot(h_axes,boundaryPts(:,1),boundaryPts(:,2),'.r');
    plot(h_axes,bottomPts(:,1),bottomPts(:,2),'.r',...
        topPts(:,1),topPts(:,2),'.m');
    set(h_axes,'nextplot','replace');

    title(sprintf('%d, %#0.4g',ii,ii/frameRate));
    pause(0.001);

    % Scale and rotate the points and write to an output file
    boundaryPts = bottomPts*scaleFactor;
%    bottomPts = bottomPts*scaleFactor

    % Move the origin
    boundaryPts(:,1) = boundaryPts(:,1) - boundaryPts(1,1);
    boundaryPts(:,2) = boundaryPts(:,2) - boundaryPts(1,2);

    % Rotate the points
    fprintf(fidDisp,'%g %g ',boundaryPts(1,:));
    for iP = 2 : size(boundaryPts,1)
        v = boundaryPts(iP,:);
        r = norm(v);
        alpha0 = atan2(v(2),v(1));
        alpha1 = alpha0 - meanRotAngle;

        % Compute the new point location
        v(1) = r*cos(alpha1);
        v(2) = r*sin(alpha1);

        % Write this to the output file
        fprintf(fidDisp,'%g %g ',v);
    end
    fprintf(fidDisp,'\n');

    fprintf(fidTime,'%g\n',ii/frameRate);
```

```
end

fclose(fidTime);
fclose(fidDisp);
```

Listing B.1. MATLAB edge detection program

Bibliography

[1] I. Abbott and A. Von Doenhoff. *Theory of wing sections: including a summary of airfoil data*. Courier Dover Publications, 1959.

[2] D. Abouri, A. Parry, A. Hamdouni, and E. Longatte. A stable fluid-structure-interaction algorithm: application to industrial problems. *Journal of Pressure Vessel Technology*, 128(4):516–524, 2006.

[3] P. Amestoy, A. Guermouche, J. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2):136–156, 2006.

[4] F. Baaijens. A fictitious domain/mortar element method for fluid-structure interaction. *International Journal for Numerical Methods in Fluids*, 35(7):743–761, 2001.

[5] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[6] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2009. http://www.mcs.anl.gov/petsc.

[7] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[8] K. Bathe. *Finite element procedures*. Englewood Cliffs, New Jersey, 1996.

[9] K. Bathe and H. Zhang. Finite element developments for general fluid flows with structural interactions. *International Journal for Numerical Methods in Engineering*, 60(1):213–232, 2004.

[10] A. Beckert and H. Wendland. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Aerospace Science and Technology*, 5(2):125–134, 2001.

[11] F. Benra. Numerical and experimental investigation on the flow induced oscillations of a single-blade pump impeller. *Journal of Fluids Engineering*, 128:783, 2006.

[12] F. Blom. Considerations on the spring analogy. *International Journal for Numerical Methods in Fluids*, 32(6):647–668, 2000.

[13] D. Boger, R. Noack, and E. Paterson. Dynamic overset grid implementation in OpenFOAM. In *5th OpenFOAM Workshop*, Chalmers, Gothenburg, Sweden, June 21-24 2010.

[14] F. Bos. *Numerical Simulations of flapping foil and wing aerodynamics mesh deformation using radial basis functions*. PhD thesis, Delft University of Technology, 2010.

[15] T. Brockett. Minimum pressure envelopes for modified NACA-66 sections with NACA a= 0.8 camber and BuShips type I and type II sections. Technical Report 1780, David Taylor Model Basin, U.S. Navy, 1966.

[16] H. Bui, M. Tanaka, M. Bonnet, H. Maigre, E. Luzzato, and M. Reynier, editors. *Inverse problems in engineering mechanics*. Balkema Rotterdam, 1994.

[17] F. Casadei and J. Halleux. An algorithm for permanent fluid-structure interaction in explicit transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 128(3-4):231–289, 1995.

[18] F. Casadei, J. Halleux, A. Sala, and F. Chillè. Transient fluid–structure interaction algorithms for large industrial applications. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3081–3110, 2001.

[19] P. Causin, J. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194(42-44):4506–4527, 2005.

[20] P. Chadwick. Applications of an energy-momentum tensor in non-linear elastostatics. *Journal of Elasticity*, 5(3):249–258, 1975.

[21] R. Clark, D. Cox, H. C. Curtiss, J. W. Edwards, K. C. Hall, D. A. Peters, R. Scanlan, E. Simiu, F. Sisto, and T. W. Strganac. *A modern course in aeroelasticity*. Springer Netherlands, 4 edition, 2004.

[22] R. Cook, D. Malkus, and M. Plesha. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 3 edition, 1989.

[23] R. Cook, D. Malkus, M. Plesha, and R. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 4 edition, 2001.

[24] M. Crisfield. *Non-linear finite element analysis of solids and structures*. John Wiley & Sons Inc, 1991.

[25] J. De Hart. *Fluid–structure interaction in the aortic valve: a three-dimensional computational analysis*. PhD thesis, Eindhoven University of Technology, 2002.

[26] J. De Hart, F. Baaijens, G. Peters, and P. Schreurs. A computational fluid-structure interaction analysis of a fiber-reinforced stentless aortic valve. *Journal of Biomechanics*, 36(5):699–712, 2003.

[27] J. De Hart, G. Peters, P. Schreurs, and F. Baaijens. A two-dimensional fluid–structure interaction model of the aortic valve. *Journal of Biomechanics*, 33(9):1079–1088, 2000.

[28] J. De Hart, G. Peters, P. Schreurs, and F. Baaijens. A three-dimensional computational analysis of fluid–structure interaction in the aortic valve. *Journal of Biomechanics*, 36(1):103–112, 2003.

[29] C. Degand and C. Farhat. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers & Structures*, 80(3-4):305–316, 2002.

[30] I. Demirdžić and D. Martinović. Finite volume method for thermo-elasto-plastic stress analysis. *Computer Methods in Applied Mechanics and Engineering*, 109(3-4):331–349, 1993.

[31] S. Deparis, M. Discacciati, and A. Quarteroni. A domain decomposition framework for fluid-structure interaction problems. In *Proceedings of the Third International Conference on Computational Fluid Dynamics (ICCFD3)*. Springer, 2004.

[32] S. Deparis, M. Fernandez, L. Formaggia, and F. Nobile. Acceleration of a fixed point algorithm for fluid-structure interaction using transpiration conditions. *Mathematical Modelling and Numerical Analysis*, 37(4):601–616, 2003.

[33] W. Dettmer and D. Perić. A computational framework for fluid–structure interaction: Finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5754–5779, 2006.

[34] W. G. Dettmer and D. Perić. On the coupling between fluid flow and mesh motion in the modelling of fluid–structure interaction. *Computational Mechanics*, 43(1):81–90, 2008.

[35] E. Di Martino, G. Guadagni, A. Fumero, G. Ballerini, R. Spirito, P. Biglioli, and A. Redaelli. Fluid–structure interaction within realistic three-dimensional models of the aneurysmatic aorta as a guidance to assess the risk of rupture of the aneurysm. *Medical Engineering and Physics*, 23(9):647–655, 2001.

[36] V. Fachinotti, A. Cardona, and P. Jetteur. Finite element modelling of inverse design problems in large deformation anisotropic hyperelasticity. *International Journal for Numerical Methods in Engineering*, 74:894–910, 2008.

[37] N. Fallah, C. Bailey, M. Cross, and G. Taylor. Comparison of finite element and finite volume methods application in geometrically nonlinear stress analysis. *Applied Mathematical Modelling*, 24(7):439–455, 2000.

[38] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer methods in applied mechanics and engineering*, 163(1-4):231–245, 1998.

[39] C. Farhat, P. Geuzaine, and G. Brown. Application of a three-field nonlinear fluid–structure formulation to the prediction of the aeroelastic parameters of an F-16 fighter. *Computers and Fluids*, 32(1):3–29, 2003.

[40] C. Farhat, M. Lesoinne, and P. Le Tallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. *Computer Methods in Applied Mechanics and Engineering*, 157(1-2):95–114, 1998.

[41] C. Felippa and K. Park. Staggered transient analysis procedures for coupled mechanical systems: formulation. *Computer Methods in Applied Mechanics and Engineering*, 24(1):61–111, 1980.

[42] C. Felippa, K. Park, and C. Farhat. Partitioned analysis of coupled mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3247–3270, 2001.

[43] C. Figueroa, I. Vignon-Clementel, K. Jansen, T. Hughes, and C. Taylor. A coupled momentum method for modeling blood flow in three-dimensional deformable arteries. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5685–5706, 2006.

[44] W. Findley, J. Lai, and K. Onaran. *Creep and relaxation of nonlinear viscoelastic materials: With an introduction to linear viscoelasticity*. North Holland Publishing Company, 1976.

[45] Fluent. *Fluent 6.2 UDF manual*. Fluent Inc., Centerra Resource Park, 10 Cavendish Court, Lebanon, NH 03766, USA, January 2005.

[46] Fluent. *Fluent 6.3 user's guide*. Fluent, Inc., Centerra Resource Park, 10 Cavendish Court, Lebanon, NH 03766, USA, 2006.

[47] C. Förster, W. Wall, and E. Ramm. Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 196(7):1278–1293, 2007.

[48] J. Gerbeau, M. Vidrascu, and P. Frey. Fluid–structure interaction in blood flows on geometries based on medical imaging. *Computers and Structures*, 83(2-3):155–165, 2005.

[49] M. Glück, M. Breuer, F. Durst, A. Halfmann, and E. Rank. Computation of fluid–structure interaction on lightweight structures. *Journal of Wind Engineering & Industrial Aerodynamics*, 89(14-15):1351–1368, 2001.

[50] V. Gnesin and R. Rządkowski. A coupled fluid–structure analysis for 3-D inviscid flutter of IV standard configuration. *Journal of Sound and Vibration*, 251(2):315–327, 2002.

[51] J. Gomes and H. Lienhart. Experimental study on a fluid-structure interaction reference test case. *Lecture Notes in Compuational Science and Engineering*, 53:356–370, 2006.

[52] S. Govindjee and P. Mihalic. Computational methods for inverse finite elastostatics. *Computer Methods in Applied Mechanics and Engineering*, 136(1-2):47–57, 1996.

[53] S. Govindjee and P. Mihalic. Computational methods for inverse deformations in quasi-incompressible finite elasticity. *International Journal for Numerical Methods in Engineering*, 43(5):821–838, 1998.

[54] C. Greenshields and H. Weller. A unified formulation for continuum mechanics applied to fluid–structure interaction in flexible tubes. *International Journal for Numerical Methods in Engineering*, 64:1575–1593, 2005.

[55] M. Heil. Stokes flow in an elastic tube–a large-displacement fluid-structure interaction problem. *International Journal for Numerical Methods in Fluids*, 28:243–265, 1998.

[56] M. Heil. An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(1-2):1–23, 2004.

[57] Hibbitt, Karlsson, and Sorensen. *ABAQUS user's manual, version 6.6*. Hibbitt, Karlsson & Sorensen, Inc., 1080 Main Street, Pawtucket, RI 02860-4847, USA, 2006.

[58] H. Hibbitt, P. Marcal, and J. Rice. A finite element formulation for problems of large strain and large displacement. *International Journal of Solids and Structures*, 6(8):1069–1086, 1970.

[59] W. Hosford. *Mechanical behavior of materials*. Cambridge University Press, 2005.

[60] J. Hron and S. Turek. A monolithic FEM/multigrid solver for an ALE formulation of fluid-structure interaction with applications in biomechanics. *Lecture Notes in Compuational Science and Engineering*, 53:146–170, 2006.

[61] B. Hübner, E. Walhorn, and D. Dinkler. A monolithic approach to fluid–structure interaction using space–time finite elements. *Computer Methods in Applied Mechanics and Engineering*, 193(23-26):2087–2104, 2004.

[62] T. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Prentice-Hall Englewood Cliffs, NJ, 1987.

[63] T. Hughes, W. Liu, and T. Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29(3):329–249, 1981.

[64] K. Hunter, C. Lanning, S. Chen, Y. Zhang, R. Garg, D. Ivy, and R. Shandas. Simulations of congenital septal defect closure and reactivity testing in patient-specific models of the pediatric pulmonary vasculature: a 3D numerical study with fluid-structure interaction. *Journal of Biomechanical Engineering*, 128:564, 2006.

[65] S. Idelsohn and E. Oñate. Finite volumes and finite elements: two 'good friends'. *International Journal for Numerical Methods in Engineering*, 37(19):3323–3341, 1994.

[66] Intel. *Intel Math Kernel Library Reference Manual*. Intel, September 2007.

[67] D. Ishihara and S. Yoshimura. A monolithic approach for interaction of incompressible viscous fluid and an elastic body based on fluid pressure Poisson equation. *International Journal for Numerical Methods in Engineering*, 64:167–203, 2005.

[68] G. Jacquet-Richardet and P. Rieutord. A three-dimensional fluid–structure coupled analysis of rotating flexible assemblies of turbomachines. *Journal of Sound and Vibration*, 209(1):61–76, 1998.

[69] H. Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows*. PhD thesis, Imperial College of Science, Technology, and Medicine, 1996.

[70] H. Jasak and Z. Tuković. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30(2):1–20, 2006.

[71] H. Jasak and H. Weller. Application of the finite volume method and unstructured meshes to linear elasticity. *International Journal for Numerical Methods in Engineering*, 48:267–287, 2000.

[72] M. Kaliske and H. Rothert. Formulation and implementation of three-dimensional viscoelasticity at small and finite strains. *Computational Mechanics*, 19(3):228–239, 1997.

[73] R. Kamakoti and W. Shyy. Fluid–structure interaction for aeroelastic applications. *Progress in Aerospace Sciences*, 40(8):535–558, 2004.

[74] A. Karac. *Drop impact of fluid-filled polyethylene containers*. PhD thesis, Department of Mechanical Engineering, Imperial College London, 2003.

[75] R. Kulak. Some aspects of fluid-structure coupling. In *ASME Pressure Vessel and Piping Conference*, Orlando, FL, USA, 27 June 1982.

[76] U. Küttler and W. Wall. Fixed-point fluid–structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72, 2008.

[77] J. Lan, X. Dong, and Z. Li. Inverse finite element approach and its application in sheet metal forming. *Journal of Materials Processing Tech.*, 170(3):624–631, 2005.

[78] G. Lauchle, M. Billet, and S. Deutsch. High-Reynolds number liquid flow measurements. In *Frontiers in experimental fluid mechanics (A90-26059 10-34)*, pages 95–157. Berlin and New York, Springer-Verlag, 1989.

[79] P. Le Tallec and J. Mouro. Fluid structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3039–3067, 2001.

[80] J. Leishman. *Principles of helicopter aerodynamics*. Cambridge University Press, 2 edition, 2006.

[81] J. Lemmon and A. Yoganathan. Three-dimensional computational model of left heart diastolic function with fluid–structure interaction. *Journal of Biomechanical Engineering*, 122:109, 2000.

[82] M. Lesoinne, M. Sarkis, U. Hetmaniuk, and C. Farhat. A linearized method for the frequency analysis of three-dimensional fluid/structure interaction problems in all flow regimes. *Computer Methods in Applied Mechanics and Engineering*, 190(3121):3146, 2001.

[83] J. Leung, A. Wright, N. Cheshire, J. Crane, S. Thom, A. Hughes, and Y. Xu. Fluid structure interaction of patient specific abdominal aortic aneurysms: a comparison with solid stress models. *BioMedical Engineering OnLine*, 5:33, 2006.

[84] A. Leuprecht, K. Perktold, M. Prosi, T. Berk, W. Trubel, and H. Schima. Numerical study of hemodynamics and wall mechanics in distal end-to-side anastomoses of bypass grafts. *Journal of Biomechanics*, 35(2):225–236, 2002.

[85] X. S. Li and J. W. Demmel. SuperLU_DIST: a scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Transactions on Mathematical Software*, 29(2):110–140, June 2003.

[86] Z. Li and C. Kleinstreuer. Fluid-structure interaction effects on sac-blood pressure and wall stress in a stented aneurysm. *Journal of Biomechanical Engineering*, 127:662, 2005.

[87] H. Lin and J. Lin. Nonlinear hydroelastic behavior of propellers using a finite-element method and lifting surface theory. *Journal of Marine Science and Technology*, 1(2):114–124, 1996.

[88] R. Löhner, J. Cebral, C. Yang, J. Baum, E. Mestreau, and S. Orlando. Extending the range and applicability of the loose coupling approach for FSI simulations. *Lecture Notes in Compuational Science and Engineering*, 53:82–100, 2006.

[89] E. Longatte, V. Verreman, and M. Souli. Time marching for simulation of fluid–structure interaction problems. *Journal of Fluids and Structures*, 25(1):95–111, 2008.

[90] E. Lund, H. Møller, and L. Jakobsen. Shape design optimization of stationary fluid-structure interaction problems with large displacements and turbulence. *Structural and Multidisciplinary Optimization*, 25(5):383–392, 2003.

[91] X. Lv, Y. Zhao, X. Huang, G. Xia, and X. Su. A matrix-free implicit unstructured multigrid finite volume method for simulating structural dynamics and fluid–structure interaction. *Journal of Computational Physics*, 225(1):120–144, 2007.

[92] H. Matthies and J. Steindorf. Efficient iteration schemes for nonlinear fluid-structure interaction problems. *Computational Mechanics: Techniques and Developments*, pages 263–267, 2000.

[93] H. Matthies and J. Steindorf. Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction. *Computers and Structures*, 80(27-30):1991–1999, 2002.

[94] H. Matthies and J. Steindorf. Partitioned strong coupling algorithms for fluid–structure interaction. *Computers and Structures*, 81(8-11):805–812, 2003.

[95] H. Matthies, J. Steindorf, and G. Brunswick. Strong Coupling Methods. *Analysis and Simulation of Multifield Problems*, 2003.

[96] M. McBride, T. M. Mallison, G. P. Dillon, R. L. Campbell, D. A. Boger, S. A. Hambric, R. F. Kunz, J. P. Runt, J. M. Walsh, and B. Leschinsky. Expandable impeller pump. United States Patent 7,393,181 B2, July 2008.

[97] C. Michler, S. Hulshoff, E. van Brummelen, and R. de Borst. A monolithic approach to fluid–structure interaction. *Computers and Fluids*, 33(5-6):839–848, 2004.

[98] K. Miller, G. Joldes, D. Lance, and A. Wittek. Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. *Communications in Numerical Methods in Engineering*, 23(2):121, 2007.

[99] MpCCI. *MpCCI 3.0.6-21 Documentation*. Fraunhofer Institute for Algorithms and Scientific Computing, Schloss Birlinghoven, 53754 Sankt Augustin, Germany, May 2008.

[100] R. Noack. SUGGAR: a general capability for moving body overset grid assembly. In *17th AIAA Compuational Fluid Dynamics Conference*, 2005-5117, Toronto, Ontario, Canada, June 6-9 2005.

[101] F. Nobile. *Numerical approximation of fluid-structure interaction problems with application to haemodynamics*. PhD thesis, Politecnico di Milano, 2001.

[102] OpenCFD. *OpenFOAM user guide*. OpenCFD Limited, 9 Albert Road, Caversham, Reading, Berkshire RG4 7AN, UK, version 1.5 edition, July 2008.

[103] K. Park, C. Felippa, and R. Ohayon. Partitioned formulation of internal fluid–structure interaction problems by localized Lagrange multipliers. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):2989–3007, 2001.

[104] E. Paterson, J. Poremba, L. Peltier, and S. Hambric. A physics based simulation methodology for predicting hydrofoil singing. In *Proceedings of the 25th Symposium on Naval Hydrodynamics*, St. John's, Newfoundland and Labrador, CANADA, 8-13 August 2004.

[105] J. Penrose and C. Staples. Implicit fluid–structure coupling for simulation of cardiovascular problems. *International Journal of Numerical Methods in Fluids*, 40:469–480, 2002.

[106] H. Petrie, S. Deutsch, T. Brungart, and A. Fontaine. Polymer drag reduction with surface roughness in flat-plate turbulent boundary layer flow. *Experiments in Fluids*, 35(1):8–23, 2003.

[107] L. Piegl and W. Tiller. *The NURBS book*. Springer Verlag, 1997.

[108] S. Piperno and C. Farhat. Partitioned procedures for the transient solution of coupled aeroelastic problems–part II: energy transfer analysis and three-dimensional applications. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3147–3170, 2001.

[109] S. Piperno, C. Farhat, and B. Larrouturou. Partitioned procedures for the transient solution of coupled aeroelastic problems part I: Model problem, theory and two-dimensional application. *Computer Methods in Applied Mechanics and Engineering*, 124(1-2):79–112, 1995.

[110] Pointwise, Inc. *Gridgen user manual, version 15*, 2007.

[111] T. Sawada and T. Hisada. Fluid-structure interaction analysis of a two-dimensional flag-in-wind problem by the ALE finite element method. *JSME International Journal Series A*, 49(2):170–179, 2006.

[112] R. Shield. Inverse deformation results in finite elasticity. *Zeitschrift für Angewandte Mathematik und Physik (ZAMP)*, 18(4):490–500, 1967.

[113] J. Sigrist and D. Broc. Homogenisation method for the dynamic analysis of a complete nuclear steam generator with fluid–structure interaction. *Nuclear Engineering and Design*, 238(9):2261–2271, 2008.

[114] A. Slone, K. Pericleous, C. Bailey, and M. Cross. Dynamic fluid–structure interaction using finite volume unstructured mesh procedures. *Computers and Structures*, 80(5-6):371–390, 2002.

[115] A. Slone, K. Pericleous, C. Bailey, M. Cross, and C. Bennett. A finite volume unstructured mesh approach to dynamic fluid–structure interaction: an assessment of the challenge of predicting the onset of flutter. *Applied Mathematical Modelling*, 28(2):211–239, 2004.

[116] M. Souli, A. Ouahsine, and L. Lewin. ALE formulation for fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 190(5-7):659–675, 2000.

[117] K. Stein, R. Benney, V. Kalro, T. Tezduyar, J. Leonard, and M. Accorsi. Parachute fluid–structure interactions: 3-D computation. *Computer Methods in Applied Mechanics and Engineering*, 190(34):373386, 2000.

[118] K. Stein, R. Benney, T. Tezduyar, and J. Potvin. Fluid–structure interactions of a cross parachute: numerical simulation. *Computer Methods in Applied Mechanics and Engineering*, 191(6-7):673–687, 2001.

[119] K. Stein, T. Tezduyar, and R. Benney. Computational methods for modeling parachute systems. *Computing in Science & Engineering*, 5(1):39–46, 2003.

[120] K. Stein, T. Tezduyar, and R. Benney. Mesh moving techniques for fluid-structure interactions with large displacements. *Journal of Applied Mechanics*, 70:58, 2003.

[121] J. Steindorf and H. Matthies. Numerical efficiency of different partitioned methods for fluid-structure interaction. *Journal of Applied Mathematics and Mechanics*, 2(80):557–558, 2000.

[122] B. Stroustrup. *The C++ programming language*. Addison-Wesley Reading, MA, 3 edition, 1997.

[123] E. Swim. *Nonconforming finite element methods for fluid-structure interaction*. PhD thesis, Texas Tech University, 2005.

[124] E. Swim and P. Seshaiyer. A nonconforming finite element method for fluid-structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 195(17-18):2088–2099, 2006.

[125] G. Taylor, C. Bailey, and M. Cross. A vertex-based finite volume method applied to non-linear material problems in computational solid mechanics. *International Journal for Numerical Methods in Engineering*, 56:507–529, 2003.

[126] P. Teixeira and A. Awruch. Numerical simulation of fluid–structure interaction using the finite element method. *Computers and Fluids*, 34(2):249–273, 2005.

[127] T. Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics*, 28:1–44, 1991.

[128] T. Tezduyar, S. Sathe, M. Schwaab, and B. Conklin. Arterial fluid mechanics modeling with the stabilized space-time fluid-structure interaction technique. *International Journal for Numerical Methods in Fluids*, 57(5):601–629, 2008.

[129] T. E. Tezduyar and S. Sathe. Modelling of fluid-structure interactions with the space-time finite elements: solution techniques. *International Journal for Numerical Methods in Fluids*, 54(6-8):855–900, 2007.

[130] T. E. Tezduyar, S. Sathe, T. Cragin, B. Nanna, B. S. Conklin, J. Pausewang, and M. Schwaab. Modelling of fluid-structure interactions with the space-time finite elements: arterial fluid mechanics. *International Journal for Numerical Methods in Fluids*, 54(6-8):901–922, 2007.

[131] T. E. Tezduyar, S. Sathe, J. Pausewang, M. Schwaab, J. Christopher, and J. Crabtree. Interface projection techniques for fluid-structure interaction modeling with moving-mesh methods. *Computational Mechanics*, 43(1):39–49, 2008.

[132] T. E. Tezduyar, S. Sathe, M. Schwaab, J. Pausewang, J. Christopher, and J. Crabtree. Fluid–structure interaction modeling of ringsail parachutes. *Computational Mechanics*, 43(1):133–142, 2008.

[133] P. Thomas and C. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, 1979.

[134] A. Timperi, T. Pättikangas, J. Niemi, and M. Ilvonen. Fluid-structure interaction analysis of a water pool under loading caused by steam injection. Technical Report Research Report TUO72-056662, VTT Industrial Systems, Espoo, Finland, 2006.

[135] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. Tezduyar. Computer modeling of cardiovascular fluid–structure interactions with the deforming-spatial-domain/stabilized space–time formulation. *Computer Methods in Applied Mechanics and Engineering*, 195(13-16):1885–1895, 2006.

[136] R. Torii, M. Oshima, T. Kobayashi, K. Takagi, and T. Tezduyar. Fluid–structure interaction modeling of aneurysmal conditions with high and normal blood pressures. *Computational Mechanics*, 38(4):482–490, 2006.

[137] Z. Tuković and H. Jasak. Updated Lagrangian finite volume solver for large deformation dynamic response of elastic body. *Transactions of FAMENA*, 31(1):55–70, 2007.

[138] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. *Lecture Notes in Compuational Science and Engineering*, 53:371, 2006.

[139] UGS Corporation. *FEMAP user guide version 9.3*, 2007.

[140] A. van Zuijlen, S. Bosscher, and H. Bijl. Two level algorithms for partitioned fluid–structure interaction computations. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1458–1470, 2007.

[141] J. Vierendeels. Implicit coupling of partitioned fluid-structure interaction solvers using reduced-order models. *Lecture Notes in Compuational Science and Engineering*, 53:1–18, 2006.

[142] J. Vierendeels, K. Dumont, and P. Verdonck. A partitioned strongly coupled fluid-structure interaction method to model heart valve dynamics. *Journal of Computional and Applied Mathematics*, 215(2):602–609, 2008.

[143] J. Vierendeels, K. Riemslagh, E. Dick, and P. Verdonck. Computer simulation of intraventricular flow and pressure gradients during diastole. *Journal of Biomechanical Engineering*, 122:667, 2000.

[144] W. Wall and T. Rabczuk. Fluid-structure interaction in lower airways of CT-based lung geometries. *International Journal for Numerical Methods in Fluids*, 57(5):653, 2008.

[145] W. Wall and E. Ramm. Fluid-structure interaction based upon a stabilized (ALE) finite element method. In *Computational Mechanics – New Trends and Applications*, 1998.

[146] H. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12:620, 1998.

[147] M. Wheel. A mixed finite volume formulation for determining the small strain deformation of incompressible materials. *International Journal for Numerical Methods in Engineering*, 44:1843–1861, 1999.

[148] F. White. *Fluid Mechanics*. McGraw-Hill, New Jersey, 3 edition, 1994.

[149] B. Wolters, M. Rutten, G. Schurink, U. Kose, J. de Hart, and F. van de Vosse. A patient-specific computational model of fluid–structure interaction in abdominal aortic aneurysms. *Medical Engineering and Physics*, 27(10):871–883, 2005.

[150] G. Xia and C. Lin. An unstructured finite volume approach for structural dynamics in response to fluid motions. *Computers and Structures*, 86(7-8):684–701, 2008.

[151] T. Yamada. Finite element procedure of initial shape determination for rubber-like materials. Technical Report Report No 20, Research Laboratory of Engineering Materials, Tokyo Institute of Technology, 1995.

[152] Y. Young. Fluid–structure interaction analysis of flexible composite marine propellers. *Journal of Fluids and Structures*, 24(6):799–818, 2008.

[153] S. Zhao, X. Xu, and M. Collins. The numerical analysis of fluid-solid interactions for blood flow in arterial structures part 2: development of coupled fluid-solid algorithms. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 212(4):241–252, 1998.

Vita
Robert L. Campbell

EDUCATION:

Ph.D. in Mechanical Engineering January 2004 to August 2010
The Pennsylvania State University, University Park, PA, 16802
THESIS: "Fluid–Structure Interaction and Inverse Design Simulations for Flexible Turbomachinery"
GPA: 4.0/4.0

M.S. in Mechanical Engineering August 1999 to December 2002
The Pennsylvania State University, University Park, PA, 16802
THESIS: "Nuclear Grid Spacer Thermal–Hydraulic Modeling Using Computational Fluid Dynamics"
GPA: 4.0/4.0

B.S. in Mechanical Engineering *summa cum laude* August 1994 to May 1998
The Pennsylvania State University, University Park, PA, 16802
SENIOR PROJECT: "Automated Material Handling for Medical-Grade Molybdenum"
GPA: 4.0/4.0

PROFESSIONAL WORK EXPERIENCE:

Senior Research Assistant April 1999 to Present
Penn State Applied Research Laboratory, State College, PA, 16804
Perform computational and experimental structural-acoustic and hydro-acoustic evaluations for projects funded by the U.S. Navy and private industry.

Safety Analysis Engineer June 1998 to April 1999
Westinghouse Electric Company, Monroeville, PA 15146
Performed transient and steady-state computer simulations for postulated accident scenarios to support licensing of commercial nuclear power plants worldwide.

Research Assistant to Dr. Michael F. Modest May 1997 to April 1998
The Pennsylvania State University, University Park, PA, 16802
Developed equipment to measure combustion gas transmissivities and developed computer program to post-process the measured data.

Mechanical Engineering Intern May 1996 to April 1997
SKF USA, Incorporated, Altoona, PA, 16601
Designed tooling for various assembly-line machines for the manufacturing of ball bearings.