Towards a Fluid Solid Interaction Model of a Dynamic Lung

Justin C. Jacobs
University of Denver

Follow this and additional works at: https://digitalcommons.du.edu/etd

Part of the Mechanical Engineering Commons

Recommended Citation
https://digitalcommons.du.edu/etd/313

This Thesis is brought to you for free and open access by the Graduate Studies at Digital Commons @ DU. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ DU. For more information, please contact jennifer.cox@du.edu,dig-commons@du.edu.
Towards a Fluid Solid Interaction Model of a Dynamic Lung

Abstract
Because the pulmonary system is a site for both environmental particulate contamination, as well as drug delivery into the body, numerous research groups have focused on precisely understanding its inner-workings. Past research has demonstrated the need to realistically model the lung walls in order to accurately capture the complex airflow profile throughout all of the branches. Since this is paramount to properly replicating particulate transport in the lung, computational fluid dynamics simulations on their own are inadequate, as they cannot account for lung wall dilation. Only by coupling the fluid and solid domains can natural lung behavior can be effectively modeled.

The goal of this work was to develop and validate the methods required to create a reliable computational fluid-solid interaction pulmonary simulation. To validate the proposed technique, a balloon was both experimentally and computationally modeled. A small pulmonary system was then presented to demonstrate the simulation capabilities.

Document Type
Thesis

Degree Name
M.S.

Department
Mechanical Engineering

First Advisor
Corinne Lengsfeld, Ph.D.

Second Advisor
Cynthia V. Fukami

Third Advisor
Paul Rullkoetter

Keywords
Fluids, Fluid-solid interaction (FSI), Interaction, Lung, Modeling, Solid

Subject Categories
Engineering | Mechanical Engineering

Publication Statement
Copyright is held by the author. User is responsible for all copyright compliance.

This thesis is available at Digital Commons @ DU: https://digitalcommons.du.edu/etd/313
TOWARDS A FLUID SOLID INTERACTION MODEL OF A DYNAMIC LUNG

A Thesis

Presented to

the Faculty of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Justin C. Jacobs

November 2012

Advisor: Dr. Corinne Lengsfeld
Abstract

Because the pulmonary system is a site for both environmental particulate contamination, as well as drug delivery into the body, numerous research groups have focused on precisely understanding its inner-workings. Past research has demonstrated the need to realistically model the lung walls in order to accurately capture the complex airflow profile throughout all of the branches. Since this is paramount to properly replicating particulate transport in the lung, computational fluid dynamics simulations on their own are inadequate, as they cannot account for lung wall dilation. Only by coupling the fluid and solid domains can natural lung behavior can be effectively modeled.

The goal of this work was to develop and validate the methods required to create a reliable computational fluid-solid interaction pulmonary simulation. To validate the proposed technique, a balloon was both experimentally and computationally modeled. A small pulmonary system was then presented to demonstrate the simulation capabilities.
Acknowledgements

The author wishes to thank Dr. Corinne Lengsfeld for her continuous support and advisement, Dr. Paul Rullkoetter for his guidance with the solid mechanics necessary for this research, and all of the family and friends who agreed to the arduous task of editing this paper.
# Table of Contents

Chapter One: Introduction ........................................................................................................ 1  
  Motivation ................................................................................................................................. 1  
  Lung Modeling .............................................................................................................................. 3  
  Scope of Research ....................................................................................................................... 7  

Chapter Two: Methods .................................................................................................................. 8  
  Simulation Sequence ................................................................................................................... 9  
  Computational Fluid Dynamics ................................................................................................... 10  
    Equations ................................................................................................................................ 11  
    Gambit ...................................................................................................................................... 13  
    Fluent ....................................................................................................................................... 14  
    Convergence Study .................................................................................................................. 16  
    CFD Pitfalls .............................................................................................................................. 18  
  Finite Element Analysis ............................................................................................................. 19  
    HyperMesh ............................................................................................................................. 19  
    Abaqus .................................................................................................................................. 22  
    Element Type .......................................................................................................................... 24  
    Convergence Study .................................................................................................................. 26  
    Hyperelastic/Elastic .................................................................................................................. 28  
    FEA Pitfalls ............................................................................................................................. 29  
  Fluid-Solid Interaction .............................................................................................................. 31  
    Pitfalls ..................................................................................................................................... 40  

Chapter Three: Validation .......................................................................................................... 41  
  Overview ................................................................................................................................... 41  
  Experimental Methods .............................................................................................................. 41  
    Gas volume vs. Diameter ......................................................................................................... 44  
  Numerical Methods .................................................................................................................... 46  
    Shell vs. Membrane Elements ................................................................................................. 47  
    Validation of Hyperelastic Behavior ....................................................................................... 48  
    Mass Scaling as Method to Increase Computational Efficiency ............................................ 51  
  Results ....................................................................................................................................... 52  
    Gas volume vs. Diameter ......................................................................................................... 52  
  Discussion .................................................................................................................................. 57  

Chapter Four: Lung Model .......................................................................................................... 59  
  Overview ................................................................................................................................... 59  
  Numerical Methods .................................................................................................................... 59  
  Preliminary Results .................................................................................................................... 62  

Chapter Five: Summary and Future Work .................................................................................. 66  

References .................................................................................................................................. 68
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A</td>
<td>75</td>
</tr>
<tr>
<td>Fluent Journal File Example</td>
<td>75</td>
</tr>
<tr>
<td>Fluent Mass Flow Rate Inlet UDF</td>
<td>75</td>
</tr>
<tr>
<td>Abaqus Input File Examples</td>
<td>76</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1: The FSI computational simulation sequence....................................................... 10
Figure 2.2: Tetrahedral Eulerian mesh for a balloon generated in Gambit. ......................... 14
Figure 2.3: A convergence study was completed to determine the optimum number of cells to use in the Eulerian mesh. .............................................................................................. 17
Figure 2.4: Convergence study showing the total wall pressure as the Fluent continuity residual lowers. ....................................................................................................................... 18
Figure 2.5: Quadrilateral elements from a 1/8 section of the balloon geometry. .............. 21
Figure 2.6: Convergence study completed in Abaqus showing a decrease in deflection error as mesh density increased. ........................................................................................................... 28
Figure 2.7: Hyperelasticity was experimentally measured before running the computational model of the balloon. ...................................................................................................................... 29
Figure 2.8: A balloon model with a poorly designed quadrilateral mesh was used in an FSI simulation ........................................................................................................................................ 30
Figure 2.9: Showing sequential frames of an elliptical balloon during an FSI simulation as it became unstable over time. .................................................................................................................. 31
Figure 2.10: The weak coupling algorithm used in MpCCI to transfer data back and forth between different computational solver codes.......................................................... 32
Figure 2.11: A typical strong coupling algorithm used for multiphysics simulations that are too unstable for the weak-coupling scheme.............................................................. 33
Figure 2.12: A graph showing how the FSI stability value from equation (17) changes for a balloon simulation as it grows................................................................. 35
Figure 2.13: A reference table showing equation (17) applied to several common FSI simulation geometries .......................................................................................................................... 36
Figure 2.14: Methodology of how two non-conformal meshes exchange data through a neighborhood search association and interpolation .............................................. 37
Figure 2.15: A lung simulation model that suffered from orphaned nodes/elements...... 38
Figure 3.1: Data from three hyperelasticity trials. ................................................................ 43
Figure 3.2: Hyperelastic trials shown in a stress versus strain curve.............................. 44
Figure 3.3: Schematic showing inner workings of a Precision Scientific wet test meter.

Figure 3.4: Graph showing experimental measurements of the balloon volume versus diameter.

Figure 3.5: A balloon model was run in Abaqus with shell and membrane elements and the results are compared.

Figure 3.6: A screenshot of a simulated rubber strip in Abaqus before (left) and after (right) a force of .02 N has elongated it.

Figure 3.7: A graph showing the computational to experimental correlation of the force versus deflection curve when using the Neo Hookean hyperelastic model.

Figure 3.8: A graph showing the computational to experimental correlation of the engineering stress versus engineering strain curve when using the Neo Hookean hyperelastic model.

Figure 3.9: A graph showing the unwanted effects of mass scaling in the computational FSI balloon simulation.

Figure 3.10: A graph showing the comparison of computational and experimental simulation results.

Figure 3.11: A more detailed version of figure 3.9. This graph shows the percent error between experimental and computational simulations as the balloon grew in size.

Figure 3.12: Abaqus CAE model showing the balloon growing over time.

Figure 3.13: A graph showing the balloon velocity profile in one plane, as time progresses.

Figure 3.14: A graph showing the balloon pressure profile in one plane, as time progresses.

Figure 4.1: Possible iterative flow structure of an idealized lung geometry.

Figure 4.2: Schematic showing method for the idealized lung geometry created in Gambit.

Figure 4.3: Showing the lung simulation air pressure contours on the walls and through slices at a time of 7.0554e-3 seconds.

Figure 4.4: Showing the lung simulation velocity vectors inside and through slices at a time of 7.0554e-3 seconds.

Figure 4.5: Showing lung nodal displacements at time 7.0554e-3 seconds.
Figure 4.6: Showing the lung simulation velocity vectors through a vertical slice........ 65
CHAPTER ONE: INTRODUCTION

Motivation

The respiratory system is unique in that it is internal to the human body but is continually exposed to external environmental agents with each breath. Foreign contaminants can easily enter the body through the lungs and quickly affect the ability to function properly. Approximately 35 million people in the US currently have some form of chronic lung disease. Lung disease here is defined as any disease or disorder in which lung performance is reduced [1].

Lung disease is a broad term used for many different types of disorders involving the pulmonary system. The different types of lung disease can be categorized into the following groups: obstructive diseases such as asthma, chronic bronchitis, and emphysema; infectious diseases such as pneumonia and influenza; all types of lung cancer; respiratory failure such as pulmonary edema and pulmonary embolism; stiffening and scarring disorders such as pulmonary fibrosis and other occupational diseases such as mesothelioma and asbestosis [1].

Although there is a considerable body of literature that documents how these diseases impact the elasticity of the lung tissue, tidal volume, breath rate and other frequently monitored breathing parameters, few to no tools are available to understand how these changes impact therapeutic response to a disease. For example, pulmonary care strategies, especially aerosol drug delivery systems, fail to provide reliable intra and inter patient dosing. Patient variability is a major factor in the inability to accurately
capture or predict therapeutic deposition (i.e., delivered dose). Modern numerical simulations used to predict the deposition efficiency of drug aerosols use static, steady state models for air flow in the extrathoracic region (mouth or nose), trachea, and upper airways derived from CT scans [65, 66] or empirically derived statistical deposition modeling based on cadaver casts of the entire lung [67]. Recently, Dr. Lengsfeld’s group used statistical deposition modeling techniques to examine if the characteristics of an optimal aerosol were a function of age [68]. They found that optimal particle size for maximum delivered drug volume varies dramatically with age from 2.5 µm for infants to 6 µm for adults. They also determined using a formal sensitivity analysis that the optimal breath rates for maximum dose delivery are below normal breathing rates, with the greatest discrepancy and sensitivity exhibited in young children (i.e., one fourth the normal breathing rate). Breath rate was found to exert a greater influence on deposition efficiency than particles size, potentially explaining the large variation observed in delivered dose from person to person or from day to day. Moreover, computational efforts aimed at the administration of a reliable dose over healthy to severe asthma breath rates produced optimal particle size distributions that were found to be far from the original optimums [78].

These recent findings call for better numerical modeling approaches that capture the dilation and contraction of each lung generation as a function of disease state. The dilation and contraction is a response to both the pressure in the thoracic cavity, the localized fluid dynamics within the lung, and local tissue properties. For example, the elasticity and thickness of the lung wall change as a function of location, age, and health state. Thus, the lung structure and aerosol drug deposition efficiency are strongly coupled
to how the pressure drop in the thoracic cavity translates to lung volume, flow rate, inspiratory time, and expiratory time. It is only by coupling computational fluid dynamic modeling to solid modeling of the soft tissue that more natural lung behavior can be replicated.

This thesis takes the critical steps necessary to make such a model a reality by establishing and validating the tools and methods required. Primarily the work will focus on a simple inflating balloon, whose geometry is similar to a lung. A spherical balloon shares the same computational instabilities, material hyperelasticity, thin walled geometry, and fluid flow interaction as the lower lung generations, making it an ideal system to develop and validate the tools and methods. Finally, simple lower lung geometries are explored to demonstrate the application and identify the critical issues in this next phase of the problem.

_Lung Modeling_

The modern age of quantitative study of the pulmonary system was established by Ewald R. Weibel in his historic book _Morphometry of the Human Lung_, written in 1963. He examined and outlined the morphometry, the structural measurements, of the human lung using an entirely new methodology. His major contributions to the understanding of the pulmonary system include: alveolar and capillary dimensions, capillary blood volume, alveolar to capillary tissue barrier thickness, and general dimensions of all generations of airways [58]. The Weibel lung models are still used today in statistical particle deposition models of the lung and give a good generalization for relevant lung dimensions for emergent computational models. After Weibel, the work of K. Horsfield
and T. Pedley took the understanding of the geometry of the lung further, examining the branching angles and the energy and pressure drop throughout the lung [4, 5, 9]. Weibel, Horsfield and others like Finlay all have worked to produce better geometric models of a generalized lung [59, 70], however, many people have moved towards taking dimensions from actual cadaver models [39, 71] and micro CT imaging to add finer detail to the geometries under evaluation [56].

Computational fluid dynamic (CFD) models have been extensively used to understand flow through the pulmonary system. The term fluid is defined as a substance that has no fixed shape and will continually deform in response to a constantly applied external pressure. In this regard, both liquids and gases are considered “fluids”. Finlay et al. did considerable modeling of the mouth and throat to understand the impact of turbulence on particle deposition [70]. Lin et al. pushed the understanding of how adding the complex geometry seen in the oropharynx and larynx to a CFD model changes the characteristics of the flow through the throat, and thus completely alters the flow patterns entering the bronchioles [6]. Large CFD sections of the lung have been simulated to show the nature of the secondary vortices that develop in realistically modeled asymmetric airways. The largest lung model published is from Gemci et al. containing the first 17 generations of the 25-generation lung [7]. In addition to exploring specific sections of the lung, several groups have focused on improving the accuracy of algorithms used to describe particle transport to the lung surfaces. One of the main factors that influence particle flow is turbulence, which can be modeled using large eddy simulations (LES), direct numerical simulations (DNS), and Reynolds-averaged Navier-Stokes (RANS) models [25, 32, 38, 57]. All of this previous work, however, ignores the dynamic dilation
behavior of the lung. A dilating lung model will be able to capture local fluctuations that interact with the fluid and particle flow, thereby enhancing deposition rates by sedimentation, impaction, and diffusion. Fluid-solid interaction (FSI) simulations, such as these, would add significant value to all of these previous efforts.

There has been an effort to model the dilating behavior of a lung, but mostly from the solid mechanics aspect. These efforts incorporated the structural geometry with the hyperelastic properties of lung tissue mainly to improve upon our understanding of pulmonary mechanics. Alveolar material properties have been extensively studied with finite element analysis (FEA) to better understand the reduced surface tension effect from surfactant in the wall lining [27, 47]. Additional studies have been completed to model the effects of cancerous lung tissue on tidal volume and energy requirements for breathing [36, 72]. These transient solid models fail to capture the fluid dynamics in the lung. FSI capabilities would add solid material behavior due to changes in total pressure in high velocity regions compared to low velocity regions, as well as an ability to track particle deposition with time. In the future, lung model validation could possibly be obtained from experimental models that document the hyperelasticity behavior of different portions of the lung under different health conditions.

Fluid-solid interaction simulation strategies largely began with custom code development. An example is the Team for Advanced Flow Simulation and Modeling (T*AFSM), in development since 1994. Applications have included descent of a porous parachute, a piece of cloth falling through air onto a rigid rod, the flow in a tube constricted with a flexible diaphragm, inflation of a balloon, flow through and around a windsock, and even a patient-specific cerebral aneurysm [12, 41]. Key aspects to FSI
simulations reside in how to deal with the contact between structural surfaces, remeshing frequency, strength of coupling between the fluid and solid algorithms, and potential for artificial stiffening due to remeshing near a boundary [12, 37, 40, 41, 62].

However, only one group (W. Wall et al.) has worked to apply FSI techniques to a lung under tidal breathing. A simulation of the upper portions of the lung was completed and showed that minute changes to the structure of a bronchial tube would drastically alter the airflow results. While the lung wall dilation in the stiff upper generations is relatively small to non-existent, the lung walls become more elastic further in and the cross-sectional deformation increases. Using patient specific biomedical studies as validation, this group is pushing the boundaries in pulmonary FSI simulations. A method of validating FSI alveolar simulations by using a novel endoscopic apparatus for investigating living alveoli was introduced that allows the in vivo measurement and control of alveoli for the first time. Simulations of very flexible elastic structures such as a single alveoli and the flow through abdominal aortic aneurysms were completed to show their computational modeling capabilities. Currently their efforts deal with investigating the complex local stress and strain behavior in alveoli and improving the material modeling in the tracheobronchial region to include the effects of adding fibrous cartilage [2, 8, 11, 19].

Considering how much the wall structure affects the flow characteristics, it can easily be reasoned that even small deformation of the bronchial tubes could greatly change the flow and particle deposition characteristics. Although custom FSI software is computationally efficient, it does not allow for the rapid inclusion of the latest advances in solid mechanic simulations, and fluid-particle interaction and tracking. The current
work seeks to utilize commercial software programs, so as to allow for the maximum amount of versatility in subsequent simulations.

Scope of Research

The long-term objective of this work is to develop a lung simulation capable of tidal breathing while allowing for multiphase flow (particle deposition, pneumonia etc). The scope of the current effort is to develop and validate the methodologies required to create this simulation. Therefore, significant effort is dedicated to describing the intricacies of the methods employed, problems encountered, and the work-around solutions. A simplified balloon geometry is explored computationally and experimentally to validate the tools and methods. Finally an example of a small pulmonary system is presented.

Computational modeling at best is always only an approximation to a solution. Because of the complexity required in developing any numerical or computational model, physical experimental data must be consulted for validation. Ideally, FSI data from an actual human or animal lung would have been used to validate the methods presented in this thesis. Unfortunately, a lung was not available, nor was the relevant data required to complete the validation. The next best thing was to devise an experimental FSI simulation that involved materials of similar properties and deformations. To accomplish this, a hyperelastic inflating balloon was both experimentally and computationally modeled.
CHAPTER TWO: METHODS

Research in the area of FSI has matured significantly over the last decade. Multiple dedicated research groups have made considerable advances to the field in recent years [37]. Three general FSI algorithm options are available: solving all FSI equations together in a monolithic approach each time step; solving the fluid and solid equations separately once and transferring data before each time step (explicit or weak coupling); or allowing the solvers multiple data transfers to ensure an exact balance of energy before stepping forward in time (implicit or strong coupling) [42]. Because the equations are customized to fit the specific application for the first option, this results in a specialized code that limits versatility. To avoid these restrictions, typically the weak or strong coupling options are used. FSI strong coupling is a relatively new development to computational modeling, and many different methods are currently being investigated [16, 19, 37, 42, 48, 55]. Because the strong coupling method is so immature, this thesis explored only the explicit weak coupling method. The tradeoffs between the two will be explained in detail later.

A single commercial software program was initially explored but it lacked the capability to do either enhanced solid behavior simulation or advanced fluid-particle interaction simulation (Abaqus). Instead this work coupled the premier solid (Abaqus) and fluid (Fluent) programs through the use of a third-party software (MpCCI).
**Simulation Sequence**

A precise sequence was required (Figure 2.1) before a simulation could be initiated, starting with the creation of the geometry for the fluid and solid domains. For the current work, both the fluid and solid domains were generated within Gambit. This program generated an Eulerian mesh of the fluid domain, while HyperMesh was used to generate the Lagrangian mesh of the tissue/membrane domain. Next, a CFD solution to the initial geometry must be completed in the absence of dynamic solid behavior. The coupling software, MpCCI (Mesh-based parallel Code Coupling Interface), would start the simulations by taking fluid pressure nodal values from Fluent and interpolating to determine the pressure at each node in the solid model. Using boundary conditions and material property information contained in input files, Abaqus used this pressure during one time step to find nodal displacements and MpCCI transferred this data back to the fluid mesh. The CFD program’s dynamic meshing capabilities were then used to fit the fluid mesh to this new domain. Thus a cyclical exchange of information was generated to drive towards a solution. In the following sections, details regarding CFD and FEA analytical procedures are discussed in relation to the simplified geometry for experimental comparison in chapter 3.
Computational Fluid Dynamics

CFD is a branch of fluid dynamics that uses numerical methods to approximate a solution for fluid flow problems. The fluid flow volume (or area if 2-D) is divided into a discrete number of cells, termed the mesh. Using the finite volume method, the series of partial differential equations (PDEs) are linearized across these finite sized cells to create solvable linear equations. Each calculation point in the volume, or node, has a finite volume surrounding it. The finite volume method uses conservation such that the flux entering one volume is equal to the flux exiting the previous volume. One main advantage to this method is that it is easily used with unstructured meshes. Various numerical methods can be used to implicitly solve this system of equations, such as Gaussian Elimination or the Jacobi Relaxation Technique [53].
Equations

The governing equations used in Fluent for this CFD analysis are shown as equations (1) through (10), and are generally referred to as the Navier-Stokes equations. Equation (1) is the Conservation of Mass equation for compressible flow shown for three dimensions where \( u, v, \) and \( w \) are the velocities in the \( x, y, \) and \( z \) directions respectively.

\[
\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z} = 0
\]

(1)

Because the flow is slow enough to allow for the incompressible assumption, density throughout the system is assumed constant at each time step. This assumption does not introduce any significant error into the system, and allows for a simplification of equation (1). Because density will not change spatially or over time, equation (1) is closely approximated by equation (2):

\[
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0
\]

(2)

The momentum equation in tensor form is shown in equation (3) and is expanded into \( x, y, \) and \( z \) components shown in equations (4) through (6).

\[
\frac{\partial}{\partial t}(\rho \vec{v}) + \nabla \cdot (\rho \vec{v} \vec{v}) = -\nabla p + \nabla \cdot \left( \tau \right) + \rho \vec{g} + \vec{F}
\]

(3)

\[
\rho \left[ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right] = -\frac{\partial P}{\partial x} + \mu \left[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right] - \frac{\partial (u'\nu')}{\partial x} - \frac{\partial (u'\nu')}{\partial y} - \frac{\partial (u'w')}{\partial z}
\]

(4)

\[
\rho \left[ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right] = -\frac{\partial P}{\partial y} + \mu \left[ \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right] - \frac{\partial (v'\nu')}{\partial x} - \frac{\partial (v'\nu')}{\partial y} - \frac{\partial (v'w')}{\partial z}
\]

(5)
\[
\rho \left[ \frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \right] = -\frac{\partial P}{\partial z} + \mu \left[ \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right] - \frac{\partial (w w')}{\partial x} - \frac{\partial (w v')}{\partial y} - \frac{\partial (w' w')}{\partial z} \tag{6}
\]

where \( P \) is the pressure, \( \mu \) is the viscosity, and the final three terms of each equation are the turbulence terms.

To solve for these turbulence terms, the k-omega method was used. Equation (7) shows \( k \), or the turbulent kinetic energy equation and equation (8) solves for \( \omega \), or the turbulent specific dissipation. While it looks like nine of each of these equations would need to be solved with every computational fluid iteration in all cells, each of these equations needs to be solved only once for each iteration per cell. Because turbulence becomes isotropic on a very small scale, both \( k \) and \( \omega \) only need to be considered for one directional component.

\[
\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_i}{\partial x_j} - \beta^* k \omega + \frac{\partial}{\partial x_j} \left[ (\nu + \sigma^* \nu_T) \frac{\partial k}{\partial x_j} \right] \tag{7}
\]

\[
\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha \frac{\omega}{k} \frac{\partial U_i}{\partial x_j} - \beta \omega^2 + \frac{\partial}{\partial x_j} \left[ (\nu + \sigma \nu_T) \frac{\partial \omega}{\partial x_j} \right] \tag{8}
\]

where \( k \) is the turbulent kinetic energy, \( U \) is the velocity, \( \tau \) is the stress tensor, \( \nu \) is the kinematic viscosity, \( \nu_T \) is the kinematic eddy viscosity or \( \frac{k}{\omega^2} \), and \( \alpha, \beta, \beta^*, \sigma, \sigma^* \) are the standard closure coefficients experimentally found to be:

\[
\alpha = \frac{5}{9}, \quad \beta = \frac{3}{40}, \quad \beta^* = \frac{9}{100}, \quad \sigma = \frac{1}{2}, \quad \sigma^* = \frac{1}{2}
\]

The Ideal Gas Law (9) is the most basic equation of state while still allowing for variable temperature, pressure and density. Although most liquids can be modeled as
incompressible fluids, air and other gases are much more susceptible to compression and require an equation of state.

\[
Z = \frac{PV}{RT}
\]  

(9)

where \(Z\) is the compressibility constant, \(P\) is pressure, \(V\) is volume, \(R\) is the ideal gas constant or 8.314472 (J/mol-K), and \(T\) is the absolute temperature.

Because the equation of state allows temperature to vary, the incompressible energy equation must also be used in Fluent:

\[
\frac{\partial}{\partial t} \left( \rho C_p T \right) + \nabla \cdot \left( \bar{u} \left( \rho C_p T \right) \right) = \nabla \cdot \left( k_{eff} \nabla T \right) + S_h
\]  

(10)

where \(\nabla\) is the gradient, \(\bar{u}\) is the velocity vector, \(k_{eff}\) is the effective thermal conductivity, and \(S_h\) is the source term. The source term introduces contributions from radiation and any other volumetric heat sources. The energy equation is unique in that it is the only equation that links the energy transfer due to conduction, diffusion, and viscous dissipation to the spatial temperature distribution [23].

**Gambit**

The Fluent preprocessor Gambit can be used to model the geometry, create the initial mesh, and define boundary conditions. The current work used 3-D models but there is a potential to enhance computational time by using 2-D axisymmetric models. The created solid model was used as the foundation from which all subsequent drawings and models were developed simply because this was the easiest method to completely align the solid and fluid domains. The Eulerian mesh was created from tetrahedral cells.
(if it is 2-D use triangular cells), shown in Figure 2.2. Both a MESH and STEP file are exported, one for use in Fluent and the later to be imported into HyperMesh to create the solid mesh.

**Figure 2.2.** Tetrahedral Eulerian mesh for a balloon generated in Gambit.

*Fluent*

The commercial CFD software used for this project was Fluent version 13.0 employing an incompressible, turbulent, transient fluid flow analysis. Incompressible flow is acceptable for this situation seeing as the fluid flow rate remains much below Mach 0.3, which is the threshold where the incompressible assumption becomes invalid. While the density inside the system would never vary spatially like in a compressible flow, the density was allowed to change with each time step using the Ideal Gas Law (9). This means that as air was blown into the balloon, stress was added to the balloon walls, and thus the air pressure and air density increased with each time step. Density is,
therefore, solved for by the Ideal Gas Law equation of state (9) and a pressure-based solver with energy modeling enabled was acceptable for the simplified geometry. Turbulence required the addition of two additional partial differential equations (PDEs). The turbulence equations selected were the k-ω equations. These equations were specifically designed for confined flows with swirling, which would relate very well to either a balloon blowing up, or the lung dilating.

Fluent must look at the fluid flow during a specific time step. A general rule of thumb to determine a valid time step is to set the time step less than the smallest cell size in the grid divided by the maximum velocity in the flow \( t \leq \frac{x}{u} \). This rule works well for helping convergence of the momentum equations, but often the time step size needs to be lowered by as much as a factor of 100 for convergence of the turbulence equations [54]. With FSI simulations, the smallest time step was driven by the solid modeling program to gain greater stability and a stronger FSI coupling.

Standard air properties were used in all simulations. The operating pressure was set to 101325.4 Pascal, or the standard air pressure at sea level. As a result of the solid material instability, a pressure inlet boundary condition was established for slowly ramping up the inlet pressure. Linear, sine wave, and exponential increase of the mass flow rate were all evaluated, but the initial slope needed to be very slow to maintain stability. Both the sine wave and exponential increase were found to be suitable. The UDF and example journal file can be seen in Appendix A.

A dynamic mesh was allowed with smoothing, layering, and remeshing capabilities. This was necessary because as the balloon expands, more cell volumes
would need to be created at the interface. Finally simulations were initialized off of inlet conditions and residual convergence was set to the standard $1 \times 10^{-3}$. Figure 2.4 shows a convergence study that confirms that these residuals were sufficient.

**Convergence Study**

A convergence study is necessary to confirm that a simulation’s mesh density is sufficient to accurately approximate real-world behavior. A general rule to get a starting point for a fluid mesh density is to always have at least 6 cells across a region of interest such as a constriction or inlet/outlet. As the number of cells at the inlet might be critical to flow behavior, especially in a dynamic lung, a convergence study was necessary to show that the cell size was small enough to accurately report velocities and pressure distributions. Figures 2.3 and 2.4 show the convergence studies endorsing reliable results.

The maximum velocity was used as the convergence parameter in the first study. The graph shows that as the number of cells increase, the error in maximum velocity in the model decreases. The graph is normalized so that the simulation with the highest number of cells (534,942) is assumed to be the correct solution. Not shown in figure 2.3 is the amount of time required to solve a simulation with each mesh density. The number of cells used in the mesh determined the fluid computational time. The red dot represents the mesh density chosen at 311,487 cells, with six across the inlet. This Fluent simulation solved in 300 seconds, versus the next larger mesh evaluated (390,758 cells) which took 543 seconds to converge. Because of this large increase in computational time with a more dense mesh, using a model with six cells across the inlet was found to be sufficient.
Figure 2.3. A convergence study was completed to determine the optimum number of cells to use in the Eulerian mesh. The maximum velocity was used as the convergence parameter. The red dot represents the mesh density used in the balloon model (311487 cells in all, with 6 across inlet).

A second convergence study was completed to verify the correct air pressure distribution within the balloon. As wall pressure is the quantity transferred from Fluent to Abaqus in an FSI simulation, it is important to know that this value has converged. Looking at only the mesh density case used (6 across inlet), figure 2.4 shows that the total wall pressure force converged as the continuity residuals hit about $1 \times 10^{-2}$. Since lowering residuals to at least $1 \times 10^{-3}$ is recommended for all implicit fluid flow simulations, this was chosen.
Figure 2.4. Convergence study showing the total wall pressure as the Fluent continuity residual lowers. The wall pressure was the quantity to be transferred from Fluent to Abaqus in an FSI simulation.

**CFD Pitfalls**

The Fluent case and data files were saved at regular intervals during the FSI simulation. As these could be a fairly large file format, it was necessary to limit the number saved. Because the data file was usually much larger than the case file, it was possible to save only case files. Later when looking at the flow results in Fluent, it was then necessary to recreate the data file by running the case file until convergence just like the FSI model did.

The initial Fluent time step would sometimes have trouble converging the continuity residual because the mass flow rate at that time was so small. This was due to truncation error associated with the mass flow rate. Ignoring this issue for the first few
time steps had no impact on the results but enabled the FSI simulation to continue. Subsequent time step residuals converged without issue.

**Finite Element Analysis**

Finite element analysis (FEA) is another numerical method for finding approximate solutions to partial differential equations. The standard form for FEA uses Lagrangian meshing. In a Lagrangian calculation, the finite mesh coordinates correspond to specific points on the material. As the material is stressed and deforms, the mesh exactly follows the progression of the material [48]. The material boundaries are therefore automatically updated as the edges of the mesh volume move with the material. This method works well for solid materials because the boundary of the solid material is usually very well defined and deformations are relatively small and slow compared to fluid flows.

*HyperMesh*

Many commercial programs are available for FEA mesh generation. The program HyperMesh was used for this study. For consistency between the fluid and solid models, all initial drawings were completed in the solid modeling program Gambit. HyperMesh was used to convert this drawing to a useable form for Abaqus solid modeling. This 3-D model was used to create the FEA elements used as the balloon material. The surface of the model, essentially the balloon walls, was meshed with the best quality quadrilateral elements possible. Later on after the balloon model validation was completed, a simple
geometry was designed for the lung. This lung model was idealized to ensure that no sharp edges would cause excessive skewness in the quadrilateral elements.

It was found that the FEA mesh quality was a large factor in determining how the FSI model would run. Triangular elements were initially investigated but were found to not work for this simulation. Because of the low density and thinness of the balloon walls, very minor skewness in the triangular elements lead to a complete failure of the simulation.

HyperMesh was used to create the solid mesh file that would be incorporated in the input files used for Abaqus. While each FSI simulation would have different critical parameters, it was found that for modeling very thin and low-density materials, the solid mesh quality is of the utmost importance. For the balloon material, a 2-D mesh of shell or membrane elements was created around the 3-D surface. Any small non-uniformity in the mesh would cause the thin material to deform in strange locations and would result in oscillations that would crash a simulation. Instead a high-quality quadrilateral solid mesh was used (figure 2.5). This figure shows a 1/8 section of the balloon mesh. For a spherical shaped model, this was the largest region for which HyperMesh could automatically generate a high quality mesh. This mesh was then reflected across the axes to create a very uniform mesh. A similar partitioning and reflecting technique was used when creating the solid mesh for sections of a lung. When mesh sections were reflected, copied, or translated, it was necessary to equivalence them before using the mesh in a simulation. When a mesh section was reflected across a plane there was always overlapping nodes at the intersections, which were corrected through equivalency.
It could be reasoned that for a more complicated solid structure, it would range from very difficult to completely impossible to create quality quadrilateral meshes. While investigating this was not within the scope of this project, it may well be true that if the solid material was modeled as thicker or denser, the FSI simulation might be less sensitive to mesh quality.

Figure 2.5. Quadrilateral elements from a 1/8 section of the balloon geometry.

After the solid mesh was created, it was necessary to define all of the entity sets to be used by Abaqus. For a balloon or lung model, a bounded node set was always defined as a ring of nodes around the inlet. This was necessary so that when forces are put on the model, the whole model would not transplant unbounded. Element sets for were created to define the interfacial FSI layer and to allow for contact. Lastly, the element type was defined as either a shell or membrane. This mesh input file was exported and combined with the other input files created for Abaqus.
Abaqus

The commercial program Abaqus was used for all of the FEA completed in this project. Abaqus calculations were solved to converge on an approximate (finite element) solution for displacement, deformation, stress, strain, forces, or other state variables involved in the partial differential equations (PDEs). In general, as a solid system was subjected to a load, an exact solution to the variables required that the force and momentum equilibrium needed to be solved continuously throughout the system. The finite element assumption only required that this equilibrium be preserved linearly or quadratically throughout a finite number of elements inside the system. If the number of elements representing the system was sufficient, the approximated solution approached the exact answer [21].

By continually solving for the dynamic equilibrium in the system, the explicit solver used in Abaqus used the central difference rule to integrate all of the equations of motion explicitly through time. At each time step, Newton’s second law of motion was solved to determine nodal accelerations using equation (11):

$$a(t) = M^{-1}[P(t) - I(t)]$$

(11)

where $a(t)$, $M$, $P$, and $I$ were the current nodal accelerations, the lumped mass matrix, and external and internal element forces respectively. The main assumption taken by using the explicit method was that by using small enough time steps ($\Delta t$), all nodal accelerations stay constant within each time step. From equation (11), the change in velocity at the middle of the current time step can be determined using equation (12):
\[ v_{(i+\Delta t/2)} = v_{(i-\Delta t/2)} + \left( \Delta t (i+\Delta t) + \Delta t (i) \right) / 2 \cdot a(t) \] (12)

The new found velocity was multiplied by the time step and added to the previous time step’s displacement to determine the new nodal displacement \( d_{(i+\Delta t)} \) shown in equation (13):

\[ d_{(i+\Delta t)} = d_{(i)} + \Delta t (i+\Delta t) \cdot v_{(i+\Delta t/2)} \] (13)

Using each element’s area, degrees of freedom, constitutive elasticity matrix, and the element type shape function \([N]\), the strain-displacement matrix \([B]\) could be determined:

\[ [B] = [\partial][N] \] (14)

The strain-displacement matrix \([B]\) could then be multiplied by the nodal displacements \([D]\) to find the new nodal strains \(\{\varepsilon\}\):

\[ \{\varepsilon\} = [B]\{D\} \] (15)

Using the constitutive matrix \([E]\) and the nodal strains \(\{\varepsilon\}\), the new nodal stresses \(\{\sigma\}\) were calculated:

\[ \{\sigma\} = [E]\{\varepsilon\} \] (16)

Equation (13) determined the new nodal displacement only based on the displacement from the previous increment; hence this was defined as an “explicit” method. Alternatively, “implicit” methods would iteratively converge on a nodal displacement by only looking at the future time step data. When the new nodal displacement values stopped changing by a low enough residual, the simulation would move on to the next time step [21].
Element Type

The type of element used in an FEA simulation determined the way in which stress and strain were distributed throughout each element. Unlike a fluid mesh, which only had one central node per element, a solid element could have different numbers of nodes, depending on the application and the shape function required. The number of nodes in an element determined if the distribution of stress and strain throughout the element would be constant, linear, or quadratic.

Many different types of elements were investigated to represent the lung material. Solid continuum elements, membrane elements, and shell elements were investigated for their accuracy and ability to accurately mimic lung tissue. Using quadrilateral versus triangular shaped mesh was also investigated along with using full versus reduced integration methods. The mesh quality was also looked into and found to be of great importance to having the model run in entirety.

Full integration for a 2-D nine-node quadrilateral element meant that the PDEs were still represented by enough nodes such that the stress and strain vary by a quadratic function across them. Reduced integration would cause stress and strain to vary linearly across each element. A 2-D four node quadrilateral element with full integration allowed for a linear relationship, whereas the reduced integration gave a constant stress and strain across the element [50].

Because of the complexity of the lung branches, it was decided to model the lung in three dimensions to be able to accurately model non-symmetric flows through
junctions. Two dimensional shell elements were initially chosen because of their simplicity. Shell elements have a very simple formulation and are defined as only being good for planar stress and strain. These elements exhibit an artificial bending stiffness that they implement when out of plane forces were applied. Because of this, problems are anticipated with these elements when utilized in a balloon or lung model.

Membrane elements were more robust in that they allowed for out of plane stresses, while still holding accurate. One main issue encountered with membrane elements is how to deal with element to element contact. Many different options were available for how to define contact: general contact, contact pairs, contact inclusions, and many more. All options were investigated for the simulation with membrane elements and none of the definitions were found to correctly define contact. While the “contact pair” method was found to work well with shell elements in the lung simulations, shell elements were not desired. Membrane elements that came into contact in the simulation would always over-penetrate and caused the fluid solver to crash because of overlapping nodes. Both of these element types are investigated for the balloon validation simulation in chapter 3.

Finite element calculation algorithms could be classified into two separate categories: implicit or explicit integration. The implicit algorithm solved a system of equations multiple times each time step where the entire mesh structure was coupled. Since these equations described the entire system, the solution at each node was accurate. While this resulted in a product whose accuracy was not solely dependent on specifying the correct time step, it inherently took more computational power and time. The explicit
algorithm solved the solution without using a storage matrix or global system of equations. With explicit integration, each node was solved independently of all of the others as explained with equations (11) through (16). While this resulted in a faster convergence, effort had to be put into specifying a correct time step. If too large of a time step was applied to an explicit integration scheme, the nodal results would be inaccurate compared to the same simulation with an implicit scheme [49].

MpCCI only allowed the use of the Abaqus Explicit integration scheme. While this made for a much faster convergence within each time step, it required shorter time steps to ensure accurate answers. In the case of modeling either a balloon or the lung, too large of a time step resulted in violent oscillations in the model that eventually caused it to crash.

For organizational purposes, separate input files were created to define the mesh, material properties, interactions, and contact. A main input file was then created to combine all of these input files and specify the step parameters for Abaqus. An example of each of these input files can be seen in Appendix A.

Convergence Study

Because all FEA is only an approximation of the exact solution, convergence studies were regularly required to be assured that the solution was accurate. A typical FEA convergence study would be to vary mesh density and see if the displacement converges on a single value. In practice, this would mean starting with a coarse mesh and computing a rough solution. As there would be fewer elements in a coarse mesh, the
solution would be reached relatively quickly. The convergence study required that the mesh density increase with each computation until the solution converged on a single answer, regardless of number of cells in the mesh. Essentially a convergence study is completed to understand the required mesh density, and allow for a tradeoff between accuracy and speed of the simulation.

An FEA convergence study was completed with a balloon to determine optimal mesh density. This study was completed solely in Abaqus with uniform wall pressures over an arbitrary amount of time to give a sufficient balloon wall deflection to be used only for mesh density comparison. Figure 2.6 shows the deflection percent error of the point on the balloon with maximum deflection, as mesh density increases. A tradeoff between accuracy and computational run time needed to be considered when determining the best mesh density. A mesh consisting of 6784 quadrilateral elements (132 elements around balloon circumference) was selected as it was shown to have an error of only 0.8% in regards to deflection, while it still solved in only 341 seconds. The next denser mesh of 10,880 elements solved in 741 seconds, which was determined to be too slow for the increase in accuracy to be used for the more complicated FSI simulation. By using this convergence data, a good solid mesh density starting point could also be determined for a lung model.
Figure 2.6. Convergence study completed in Abaqus showing a decrease in deflection error as mesh density increased. The red point in the graph (6784 quadrilateral elements) was determined to be a sufficient mesh density and was used in the FSI balloon validation experiments.

Hyperelastic/Elastic

The modulus of elasticity is an intensive property and can be defined for a region of material. A constant elastic modulus was initially used for the entire model, but both the balloon and lung were known to be hyperelastic. A material that is hyperelastic has a stress versus strain curve that does not produce a linear line, whereas normally the slope of this line is the modulus of elasticity. Hyperelastic models either use tabular data or a quadratic curve fit to meet the changing slope of the stress versus strain curve. The experimentally measured hyperelasticity of the balloon used in the experiments are shown in figure 2.7.
Figure 2.7. Hyperelasticity was experimentally measured before running the computational model of the balloon. The polynomial trend line used for the model is shown here in the stress versus strain curve.

**FEA Pitfalls**

As discussed earlier, with a very thin and low-density solid material such as the lung walls, the quality of the solid mesh was found to be particularly important. The mesh created for the geometry was always 2-D elements folded in 3-D, always with either membrane or shell elements. Figure 2.8 shows a model of a balloon simulation with a quadrilateral mesh with imperfections. The mesh was created without sectioning the balloon into smaller sections, therefore resulting in some regions with higher cell skewness. It was found that if any triangular mesh or a non-uniform quadrilateral mesh were used for the FSI simulation, hot spots would appear and cause deformation oscillations that would continue building until the simulation crashed.
Figure 2.8. A balloon model with a poorly designed quadrilateral mesh was used in an FSI simulation. Colors show amount of displacement. A deformation hot spot occurred around a highly skewed cell. Other imperfections can be seen throughout the mesh (compared to the ideal mesh being drawn in figure 2.5).

Oscillations in the model were also found to arise from stepping up the pressure too violently. Even if the mesh quality were perfect for the balloon structure, if the pressure was not sufficiently ramped up over time, the model would crash. This could be seen when testing in Abaqus by putting a constant pressure on the inside of the balloon to blow it up, or in the actual FSI simulation if the time step between the fluid and solid computation was too large. While the mass flow rate could be ramped smoothly over time in Fluent, if the time step between the two programs were too large, the Abaqus model would essentially see a saw tooth increase in pressure inside the balloon. This ended up being the driving factor for computational time with this simulation. Sequential screenshots are shown in figure 2.9 as an example of the simulation instabilities seen here as oscillations.
A seemingly obvious solution to the problem of uncontrollable oscillations would be to introduce damping. This was investigated with the balloon simulation, but proved not to be beneficial. A simple damping command could usually be used with 3-D elements to reduce oscillations and improve stability. With the shell and membrane elements used in these simulations, damping either caused the stable time increment to lower too far to an unusable time step or would not have any effect at all.

Physically it could be reasoned that it could be difficult to dampen very thin, low density structures. As with the balloon model, all forces were perpendicular to the 2-D elements. Because the elements essentially did not have any structure in the direction of the motion, damping could not be completed.

**Fluid-Solid Interaction**

MpCCI is a multiphysics commercial software program that interpolates nodal information from one software program to another. Because the fluid Eulerian mesh relied on being fixed in space with fluid flow through it and the solid mesh deformed as the solid moves, the interaction of the two mesh geometries were taken into account with FSI. Since it would be unreasonable to have the fluid and solid nodes exactly align at the
interface, the data from one to the other needed to be interpolated. The calculations for each of the mesh geometries only happened at the nodal locations (or integration points for FEA), but interpolation techniques could be used to pass FSI results from the fluid model to the solid model and back. This interpolation was the primary job of the interaction software called MpCCI.

MpCCI solved the fluid and solid simulations each time step separately and used a weak-coupling method [24]. The weak coupling algorithm used for the simulation is shown in figure 2.10. Fluent was the starting code, or code A, shown in the figure, while Abaqus was code B. After it received data, one solver computed a solution and passed the resultant values to the other solver. After a solution was reached, the simulation took a time step and repeated the process.

![Figure 2.10. The Weak Coupling Algorithm used in MpCCI to transfer data back and forth between different computational solver codes [24].](image)

As opposed to the weak-coupling scheme used by MpCCI, a strong-coupling algorithm could sometimes be used for multiphysics pairing. Recently, a lot of effort has been put into developing these more robust algorithms. The basic idea of the strong-
coupling algorithm is shown in figure 2.11. The main difference here was that the solvers now had the ability to pass data back and forth multiple times before reaching convergence and stepping forward in time [42]. While at first glance this might look like it would always involve more computational time than the weak algorithm, it is possible that the time step could be increased because of the added stability gained from using this method. As these methods are still immature, commercial coupling programs such as MpCCI have yet to fully implement them.

![Diagram of strong coupling algorithm](image)

**Figure 2.11.** A typical strong coupling algorithm used for multiphysics simulations that are too unstable for the weak-coupling scheme. Here data is exchanged between the two solvers multiple times until a convergence criterion is reached before moving on to the next time step [42].

An enlightening equation that related the geometry of an FSI simulation to its tendency towards instability is shown in equation (17):

\[
\frac{\rho_s h \pi^2 R}{\rho_f 2L^2} > 1
\]  

(17)

where \( \rho_s \) is the density of the solid material, \( \rho_f \) the density of the fluid, \( h \) the thickness of the solid material, \( R \) the radius of the fluid passageway between the solid walls, and \( L \) is the length of the fluid passageway. This equation had to be satisfied or instabilities
were shown to occur in an FSI simulation [42]. This explained why simulations such as arterial blood flow could be especially difficult to model. If the density of the fluid was close to the density of the solid, the so-termed “added-mass effect” occurred where fluid flowing into a solid wall acted as an extra mass, which is added to the structural degrees of freedom at the fluid-solid boundary. Because of the similarity of the fluid and solid densities, biomechanical FSI simulations have been shown to be most inflicted by these instabilities [55].

While there is a 1000:1 density ratio for the solid to fluid components when modeling the lung, the lung wall thickness was so small that equation (17) comes dangerously close to instability in the lung. As the model geometry changed with each time step, this equation showed that the simulation almost always got closer to instability.

Chapter 3 will discuss the FSI simulation of a spherical balloon with a diameter of 0.1 meters, which started at a stability value of 4.4 as shown in figure 2.12. As the simulation progressed and the balloon grew, it would have dropped below a value of one near a diameter of 0.43 meters. This simulation went unstable at a diameter of 0.2 meters with a stability value of 2.2 and is shown in red in figure 2.12. As equation (17) was intended to be used as a general rule to determine if the FSI simulation is anywhere close to instability, it was difficult to know at exactly what size an instability would occur in this case. As shown earlier in figure 2.8, solid mesh quality was found to be another factor that determined FSI simulation stability. It was found that if the simulation geometry was close to a stability value of one, any problems with mesh quality were enough to drive the simulation towards instability. Because a perfect mesh was not
achievable with the balloon simulation, the simulation crashed before the equation (17) prediction.

\[
\text{Figure 2.12. A graph showing how the FSI stability value from equation (17) changes for a balloon simulation as it grows. While the equation states that the value must be greater than 1 to be stable, the balloon validation experiment completed in Chapter 3 went unstable at a value of 2.2 when it was at a diameter of 0.2 meters as shown in red.}
\]

Using equation (17) and some common values for each setup, the reference table in figure 2.13 was created to summarize the expected stability value of several common FSI simulation geometries. Depending on the actual case or even the time step at which the FSI simulation is at, the stability value for a particular simulation could be anywhere along the shown bars [69, 73].
Figure 2.13. A reference table showing equation (17) applied to several common FSI simulation geometries. Instability theoretically occurs at less than a value of one, with increased instability the lower the value gets. As can be seen, most biomechanical FSI simulations will experience instabilities for some of their geometries.

As is shown in figures 2.12 and 2.13, a balloon FSI simulation becomes unstable with certain geometries. Figure 2.13 shows that simulations of parts of the lower lung generations can also come very close to instability as well. Because the lung generations become less slender with thicker walls in the upper portions, they will tend to be more stable. FSI simulations of arteries are known to be some of the hardest to model, and this figure has confirmed that. It should be noted that the data in this figure assumes that the FSI solvers are explicitly coupled and instability dominates the model below a value of one. Implicitly coupled solvers might be able to successfully run these models to a lower stability value, but specific data backing this is not available.

MpCCI uses an interpolation scheme when transferring data from one model to another. They state that during this process, the mesh geometries, the data distribution and the conservation of flux must all be considered [24]. The method used by MpCCI to
transfer data between two non-conformal meshes involves both association and interpolation. Association refers to how the elements or nodes of each mesh create partners through a neighborhood search to determine where the transferred data goes. The neighborhood search is based on a Kd-tree implementation where a bounding box is created around the element or node and all points satisfying the specific conditions are selected as partners. Shape functions are used to interpolate how the transferred data varies across the new element it was assigned to [24]. Figure 2.14 shows two non-conformal mesh geometries that MpCCI would use association and interpolation to transfer data back and forth between.

![Figure 2.14](image)

**Figure 2.14.** Methodology of how two non-conformal meshes exchange data through a neighborhood search association and interpolation. The quadrilateral mesh represents the solid interface and the triangular mesh represents the fluid mesh. Even though shown separated for clarity, the meshes occupy the same spatial domain [24].

One downside to having non-conformal meshes was that it was sometimes possible to have orphaned nodes or elements. This was caused by the association bounding-box around the source node being too small to see the correct nodes in the interfacial layer of the other mesh. These nodes or elements wouldn’t receive the
quantities being transferred and are called “orphans”. They no longer had the ability to transfer or receive data and the simulation usually suffered from this. MpCCI allowed for two methods to deal with orphaned nodes. One was to specify default values along with a ramping function to be assigned to orphaned nodes to slowly try to get them back to normal. The other was to allow MpCCI to extrapolate data out to orphaned nodes from nearby nodes not suffering from the same dilemma. A lung simulation inflicted with orphaned nodes is shown in figure 2.15. This simulation did not use either correction method, so orphaned nodes permanently remained so.

![Image](image.png)

**Figure 2.15.** A lung simulation model that suffered from orphaned nodes/elements. The colors represent magnitude of displacement and the orphaned nodes/elements can easily be seen by the red dimples created in the lung walls.

Orphaned nodes could be caused by too large of a time step, allowing for one mesh to move too much for the bounding-box to account for it. Using smaller time steps could possibly solve this problem. Another method of overcoming the problem of
orphaned nodes would be to enlarge the bounding-box used by the neighborhood search. The main problem with enlarging the bounding box was the effect this would have on the computational speed. Because a bounding box is used for every node in the FSI model, increasing this search parameter would critically increase the amount of time necessary to complete the simulation.

Before starting the MpCCI program, it was necessary to have created a case and a data file in Fluent, and an input file in Abaqus. To see the sequence of programs required to obtain these files, reference Figure 2.1 and discussions earlier in this section. When opening MpCCI, the GUI starts in the “Models” step. It was required that the Fluent 3DDP (double precision) program be used to avoid decimal truncation errors with the small size of the models. It was also necessary to define the unit system used when designing the model in Abaqus.

The next step encountered in MpCCI was the “Coupling” step. Here the coupling surfaces from each program are selected, and the variables that are to be transferred back and forth are defined. For the FSI simulations, the quantities to exchange were defined as RelWallForce (wall pressure from Fluent nodes) and NPosition (nodal displacement from Abaqus).

In the “Go” step, the coupling configuration was defined to show which program drives the coupled simulation. Because the pressures from Fluent initially drove Abaqus to displace the nodes, under the Fluent coupling configuration tab, the initial quantity transfer was set as “exchange”. Abaqus was set up to only “receive” on the initial quantity transfer, as it needed to wait for fluid pressures before moving ahead. When
MpCCI was initialized, the Fluent graphical user interface (GUI) would open completely populated with everything saved in the case file. If everything was ready in Fluent, the calculations would begin.

For post processing or checking results during the simulation, it was most useful to use Abaqus CAE. The (*.odb) file that gets saved with displacement results could be opened to see the displacements at the specified time step intervals. If the fluid flow was of more concern, Fluent could be opened to look at saved case and data files.

Pitfalls

The major items to be concerned with when dealing with MpCCI were to correctly define the time steps to avoid oscillations in the model and to not allow for orphaned nodes. Of course making the time step larger made the simulation run much faster, but this should be weighed against the probability of these problems occurring. Unfortunately, the ideal FSI time step was usually found to be equal to the stable time increment computed by Abaqus, which was based on the element size and density. During the course of this thesis, the solid material elements were small and had low density, so the simulation always had a small time step and required a long computation time.
CHAPTER THREE: VALIDATION

Overview

Computational models are only as accurate as the algorithms, boundary conditions and initial conditions forming the analytical foundation. A validation step sometimes called a calibration step is required to ensure that the simulation outcome is physically accurate. Validation via lung experiments is difficult to impossible currently, and computationally it is a poor choice to calibrate these tools. Instead a spherical hyperelastic balloon was selected as a simplified geometry for comparison with experimental measurements. A balloon is an excellent model system because it represents a thin membrane system similar to a lung, the material behaves in a hyperelastic fashion similar to lung tissue, and the shape is driven by internal fluid flow like a human lung.

A comparison is made between the diameter of the solid system and volume of gas within the system as calculated from an FSI model and experimentally measured values.

Experimental Methods

*Hyperelastic Material Behavior*

It was necessary before completing the FEA to determine the hyperelastic behavior of the balloon material, which was obtained through a force versus displacement experiment. A known force was applied, and the displacement was measured. From this a
stress (load / cross sectional area) versus strain (elongation) relationship was determined. A normal elastic material would have a large portion of the stress versus strain curve that was linear, but for a hyperelastic material such as this, a polynomial was needed to describe the relationship [75, 76].

Two strips of material from one balloon were used for these experiments to confirm that the material was isotropically hyperelastic. Rectangles approximately 0.025m x 0.05m were cut from a balloon. Precision weights in increments of 0.05 kilograms were hung from the rubber strips, while the new material length was measured with a micrometer. This experiment was repeated three times to determine the variation. A one-way Analysis of Variance (ANOVA) was used to test the variation among the hyperelastic trials. The trials were found to have no statistically significant differences between group means ($F (3, 53) = 1.251, P = 0.301$). A graph containing data points from each of the three tests is shown in figure 3.1.
Figure 3.1. Data from three hyperelasticity trials. A one-way ANOVA was used to test the variation among the hyperelastic trials. The trials were found to have no statistically significant differences between group means ($F(3, 53) = 1.251, P = 0.301$).

For the hyperelastic data to be used in a computational model, the stress versus strain was calculated as shown in figure 3.2. The engineering strain was calculated as the change in length divided by the initial length. Stress was calculated as the load per area, where the load is the weight divided by gravity (9.81 m/s$^2$) and the area is the cross-sectional area being stretched. As the computational model would never exceed element expansion of 120%, data points beyond this are unnecessary. A $4^{th}$ order polynomial trendline shown in equation (18) was created for this region. Points were plotted every 5% strain using this polynomial to be used as the computational uniaxial test data shown in the material properties input file in Appendix A.

$$y = -4396.274268x^4 + 14055.377231x^3 - 19121.616892x^2 + 19246.108256x - 16.343944 \quad (18)$$
Figure 3.2. Hyperelastic trials shown in a stress versus strain curve. A trendline was created to define the balloon material in the computational model.

Gas volume vs. Diameter

A wet test meter from Precision Scientific was used to measure the volume of gas within the balloon for a measured diameter. The wet test meter allowed for very accurate and repeatable measurements of the volume of air contained within the balloon. Prior to using this equipment, the meter equilibration and calibration was completed [61].

Valves and tubes were attached to the meter so that a blown up balloon could be held at a certain volume while the circumference was measured. The valve could then be slowly opened such that the flow rate into the meter was controlled and did not exceed the rate required by the meter. This slow rate was necessary so that the air in the meter
being measured could be assumed to be at standard temperature, density, and pressure.

The wet test meter setup is shown in figure 3.3.

![Figure 3.3](image)

**Figure 3.3.** Schematic showing inner workings of a Precision Scientific wet test meter [61].

The wet test meter has a rotor that split the interior into four compartments shown in figure 3.3 labeled $C_1$, $C_2$, $C_3$, and $C_4$. The inlet is denoted as Center Pipe A and the outlet is shown on the top of the figure. The meter was precisely half filled with water, which acted like a seal as the rotor moves. At the point shown in figure 3.3, chambers $C_1$ and $C_4$ were being filled by air from the center pipe. As the rotor moved counterclockwise and water was removed from these chambers, they were filled with air. On the other hand, the center pipe was not filling chambers $C_2$ and $C_3$. Air from these chambers was allowed to escape through the top exit pipe. During this process a dial counted the number of revolutions. If the flow was slow enough, standard temperature and pressure could be assumed and the revolutions dial directly correlated to the volume of air passing through the wet test meter.
The volume of eight balloons all starting at a circumference of 0.695 +/- 0.001 meters were measured with the Wet Test Meter to determine its accuracy. The volumetric measurements were very repeatable with an average of 0.005765 m$^3$ with a standard deviation of $3.83 \times 10^{-5}$ m$^3$. The balloon volume was varied to produce figure 3.4. The error bars are not shown on the graph, as they are smaller than the data points.

![Balloon Diameter vs. Volume](image)

**Figure 3.4.** Graph showing experimental measurements of the balloon volume versus diameter. Because the balloon became elliptical at certain volumes, the reported average diameter is derived from orthogonal circumference measurements. The data points are larger than the error standard deviation that was found to be $3.83 \times 10^{-5}$ m$^3$. A quadratic polynomial shows the smoothness of the data.

**Numerical Methods**

The same numerical methods as discussed in Chapter 2 were utilized.
Shell vs. Membrane Elements

A set of spherical balloon simulations was run with both shell and membrane elements to observe differences. It was expected that because forces are acting on these elements out of plane, the shell elements would be overly stiff, and the membrane elements would be more accurate. As stated earlier, the formation of the shell element would create a bending stiffness that should not occur in the real material.

The comparison of a balloon model run in Abaqus with shell and membrane elements is shown in figure 3.5. For this simulation, a pressure on the interior walls of the balloon was slowly ramped up over time. While it was expected that the shell elements would be stiffer than the membrane elements, they both produced exactly the same results. This is useful knowledge as shell elements were found to work with contact, a necessary condition for the lung model. For the FSI balloon simulation, either element was sufficient. While this verification does not confirm that either element will work for the lung model, it still allows either as a possibility. As the lung involves both tension and compression, a similar test would be necessary before using shell elements.
Figure 3.5. A balloon model was run in Abaqus with shell and membrane elements and the results are compared. In the case of blowing up a balloon, either a shell or membrane elements work equally as well.

Validation of Hyperelastic Behavior

Numerical simulations not involving the balloon were conducted to verify that the hyperelastic material definition was computing the solution correctly. A rubber strip of similar size to the experiment was simulated. A force was equally distributed across the bottom of the rubber strip made up of 2-D membrane elements, and a force versus deflection curve was produced as different forces were used to pull the strip. To keep the experiment similar to the other methods to be used with the balloon and lung model, the model was set as dynamic, explicit, with membrane elements. It was necessary to set the hyperelastic model as Neo Hookean with $A_{10}=100$ psi and $D_1=5\times10^5$ psi to simulate the rubber as nearly incompressible [60]. All available hyperelastic models were attempted
and this setup was found to give the closest force versus displacement output compared to the experimental data. The Neo Hooke model equation (19) is shown as:

\[
U = C_{10}(I - 3) + \frac{1}{D_1}(J^{el} - 1)^2
\]  

where \( U \) is the strain energy potential per reference volume, \( C_{ij} \) and \( D_i \) are temperature-dependent material parameters, \( J^{el} \) is the elastic volume ratio, \( I_1 \) is the first stress-strain invariant defined in equation (20) as:

\[
I_1 = \lambda_1^{-2} + \lambda_2^{-2} + \lambda_3^{-2}
\]  

where \( \lambda_i = J^{-\frac{1}{3}} \lambda_i \) and \( J \) is the total volume ratio and \( \lambda_i \) are the principle stretches [21].

Figure 3.6 shows a screenshot of a simulated rubber strip in Abaqus before (left) and after (right) a force of 0.02 N has elongated it. The detached node below the strip was attached to all of the bottom nodes of the strip with beam elements so that the downward force put on this node would create an equally distributed load across the bottom of the strip. All of the top nodes on the strip were bounded only in the y-direction. This model was a success and verified that the hyperelastic model force versus deflection curve and stress versus strain curve accurately represent the experimental results as shown in figures 3.7 and 3.8.
Figure 3.6. A screenshot of a simulated rubber strip in Abaqus before (left) and after (right) a force of .02 N has elongated it.

Figure 3.7. A graph showing the computational to experimental correlation of the force versus deflection curve when using the Neo Hookean hyperelastic model.
Figure 3.8. A graph showing the computational to experimental correlation of the engineering stress versus engineering strain curve when using the Neo Hookean hyperelastic model. The model was verified for up to 100% engineering strain.

Mass Scaling as Method to Increase Computational Efficiency

The computational time of FSI models are significant. To model the complete lung geometry, efforts need to be focused on reducing computational times. Although the CFD computations require the largest computational effort, it is the stable time increment in the FEA model that hinders the efficiency in modeling an entire tidal breath. Mass scaling has been used to accelerate computational times by increasing the stable time increment. It has been shown that a moderate artificial increase in mass might not significantly change the response of an FEA system, while allowing for a more realistic solution time. A drastic amount of mass scaling should not be applied as it could introduce unwanted structural effects [77]. A numerical study was undertaken to compare the difference in balloon inflation behavior (deflection) as a function of mass scaling.
Figure 3.9 shows the unwanted stiffening effect introduced by using mass scaling with this type of simulation. In this case, a mass scaling of 100 and 10 corresponded to a computational efficiency increase of 10 and 3 times respectively. Because the rate of material deflection changed by such a drastic amount when any significant amount of mass scaling was added, it was determined that mass scaling would not be an effective solution to speed up the computational time.

Figure 3.9. A graph showing the unwanted effects of mass scaling in the computational FSI balloon simulation. Mass scaling was investigated as a way to increase computational efficiency but was found to not be effective for this specific FSI case.

Results

Gas volume vs. Diameter

Figures 3.10 and 3.11 provide a comparison between the experimental and numerical simulations of a balloon inflating. To incorporate both the fluid and solid
computational results in the validation, the balloon diameter versus volume of air inside the balloon is examined. This was found to be an appropriate validation parameter because as the balloon grows in size, the resultant diameter is completely dependent upon both the solid material properties of the rubber and the fluid flow characteristics and air material properties. Accurately matching the experimental and computational results in this fashion will ensure that all three computational programs are working correctly.

Figure 3.10 shows that the computational results are correctly trending along the experimental data. The computational balloon simulation was started as a perfect sphere with a diameter of about 0.159 meters (circumference of 0.5 meters) with no stresses in the rubber and standard air pressure inside the balloon. This was necessary as any initial stresses or air pressure shocked the fragile simulation and caused oscillations that lead to a crash. Because the initial conditions in the computational simulation could not exactly match up with experimental, a small offset error was expected. After the simulation was allowed enough run time to settle out, an offset error of about 4-5% was seen (figure 3.11).

Because the growing computational balloon was bounded at the inlet, it slightly strayed from being perfectly spherical as air was pushed into it. The balloon diameter was measured in orthogonal directions (X, Y, Z) and averaged to find the diameter used for the comparison.
Figure 3.10. A graph showing the comparison of computational and experimental simulation results. Although there is an offset (see figure 3.11), the computational results shown in this graph are correctly trending along the experimental data. A run time of 11 days was required to increase the diameter by 0.04m.
Figure 3.11. A more detailed version of figure 3.10. This graph shows the percent error between experimental and computational simulations as the balloon grew in size. The offset error shown in the graph was expected as the computational simulation started with no stress in the material and standard pressure air inside the balloon.

Figure 3.12 shows the results of the FSI balloon simulation. The color in the diagram represents the amount of displacement. Because the inlet had to be bounded, the spherical balloon shows the maximum deflection at the bottom of the balloon as red. A computational runtime of 11 days was required to increase the balloon diameter by 0.04 meters.
**Figure 3.12.** Abaqus CAE model showing the balloon growing over time. Color denotes the amount of displacement. Because the inlet was bounded, it deflected less than the rest of the model and is shown as staying blue.

Figure 3.13 shows the balloon velocity flow profile as the simulation time progresses. Because the mass flow rate is increasing over time, the flow velocity increases over time as seen in the graph. Figure 3.14 is a graph showing the balloon pressure profile over time. The initial pressure hot spot at 1 second is at the edge of the inlet flow and as the flow hits the bottom of the balloon, the highest pressure follows. The average pressure inside the balloon also increases over time as expected.

**Figure 3.13.** A graph showing the balloon velocity profile in one plane, as time progresses.
Figure 3.14. A graph showing the balloon pressure profile in one plane, as time progresses.

Discussion

The experimental and numerical simulations were determined to be in good agreement as shown by figure 3.11. The computational to experimental error leveled off at only 4%. While this error is acceptable, this data actually relates to a much lower error as an offset error was expected because of the differences in initial conditions. In computational simulations such as these, an absolute error under 7% is considered acceptable. In summary, the balloon simulation performed exactly as expected and matched the experimental data with high precision. This validation therefore confirmed the legitimacy of all of the methods described above. These methods can now confidently be applied to lung FSI simulations.

To acquire the necessary validation data, the balloon simulation completed calculations for 11 days. While many techniques were investigated to speed up the computational efficiency, none were found to be effective for this specific FSI case. None of the investigated techniques are expected to apply to lung FSI simulations. For lung simulations, the more complicated geometries and smaller FEA elements will drive the
computational time to be longer than the balloon simulation with the currently available resources.
CHAPTER FOUR: LUNG MODEL

Overview

The main objective of this study was to explore all possible FSI simulation methods and validate a set of methods useful for computational biomechanical FSI simulations. This can be difficult to complete without attempting an actual bio-FSI simulation. While having a fully functioning computational lung model was not a necessity for this work to be considered a success, several permutations of lung simulations were explored and the findings are discussed in this chapter.

Numerical Methods

As a first attempt at modeling the pulmonary system, an idealized version of the geometry was used. Dimensions from a lung cast from a 14.08-year-old female lung were used [69, 73]. The human lung consists of 25 distinct generations or bronchial tubes, terminating in small spherical sacs called alveoli. Because modeling all 25 generations of the lung is currently not feasible with the computational processing power available, the lung was split into discrete regions for the model. Although the model is physically broken up into sections, results from each of the sections will drive the others. As shown in figure 4.1, the mass flow rate during inspiration is initially known only at the entrance to the lung. As the lung bronchial tubes branch apart, it is impossible to analytically know the flow rate or pressure drop in specific lung geometries. The mass flow rate out of
section 1 would be fed into section 2 and so forth. After section 3 has been fed the residual mass flow rate from the section 2 outlet for one time step, a backpressure at the inlet can be determined from the model. This backpressure is fed back through the models until section 1 is reached. Explicit time stepping most likely will not be sufficient for this setup as it would allow for only one iteration of each case. Implicit integration could help stability, as it would allow for model convergence before stepping ahead in time. While figure 4.1 outlines this method for three sections, it is possible that splitting the lung into more sections could optimize the computational time.

Figure 4.1. Possible iterative flow structure of an idealized lung geometry. Sectioning the lung into distinct sections is a way to increase computational efficiency. To link the entire model together, it is necessary to transfer the mass flow rate and backpressure between the sections.

Available computing power was the driving factor for initially utilizing three sections for the lung model. Six 64-bit double Quad-Core Intel Xeon Processor computers with 8.0 Gb of RAM were used for these experiments. Each of these lung models was generated in Gambit by starting at the lower generations and building up.
With this method, each additional lung generation more than doubled the cells required to accurately resolve the complex flow characteristics.

Although not shown in the flow chart shown in figure 4.1, symmetry could have been applied twice with each of the model sections to quarter the computational cells used. This would only be possible for an idealized version of a lung, whereas lines of symmetry would make too many assumptions with a realistic geometry.

Diameter and length dimensions were given for each lung generation. To make this geometry in Gambit, tubes were united to spheres at each branching angle as shown in figure 4.2. Real lung branching angles are complex and impossible to reproduce without knowing the exact geometry. For this study, branching angles of 32 degrees were chosen [4]. Each of the sections required slightly larger branching angles with the larger generations to keep the lower generations from overlapping.

![Figure 4.2](image)

**Figure 4.2.** Schematic showing method for creating the idealized lung geometry in Gambit. Cylinders and spheres were united at specific angles to create the lung models.
Preliminary Results

Three generations of the lung terminating in alveoli were modeled and small preliminary lung simulation was completed. This lung model was created using the methods and knowledge learned from the balloon validation simulation. Because of the size of the FEA elements needed to model this miniscule part of the lung, a stable time increment of 1e-7 seconds was required, which was also used as the FSI exchange time step. The lung simulation was allowed to run for 7 days to attain the following results.

The lung walls were modeled as one cell thick at 1.5 x 10^-6 meters, with a density of 1000 kg/m³. A simple elastic material model was used with a modulus of 50 x10^6 Pa and a Poisson’s ratio of 0.41. In future experiments, different hyperelastic models should be investigated for their compatibility with lung tissue. Because the fragile simulation required the mass flow rate to slowly ramp up over time, and the time step shown in the following figures was still early and had a low flow rate, it is expected that more meaningful results would have been obtained further into the simulation.

Figures 4.3 and 4.4 show the pressure contours and velocity vectors for the furthest time step reached in the lung simulation at 7.0554e-3 seconds. While it looks like there are pressure hotspots throughout the model, the pressure only varied by 10 Pascal. The velocity vectors, on the other hand, show significant variation throughout the model, including unexpected low spots like one of the low alveoli picture as blue. It is possible that the model was still in a settling phase before more uniform flow would occur and inflate the system. If more time was allocated to running the simulation, this could be confirmed.
**Figure 4.3.** Showing the lung simulation air pressure contours on the walls and through slices at a time of $7.0554 \times 10^{-3}$ seconds. The pressure only varies by 10 Pa throughout the simulation at this time step.

**Figure 4.4.** Showing the lung simulation velocity vectors inside and through slices at a time of $7.0554 \times 10^{-3}$ seconds.
Figures 4.5 and 4.6 investigate a strange occurrence of dimples seen on the lung walls. The nodal displacements at the same time step are shown in figure 4.5. Earlier in this research it was postulated that orphaned nodes caused the dimples, but most likely this is not the case. This simulation was run with the addition of nodal extrapolation for any orphaned nodes. Because the dimples still showed, they most likely occur because of small non-uniformities in the mesh that create small pressure spikes in the material. Figure 4.6 shows the vortex airflow that resulted from the occurrence of dimples, most likely perpetuating them by in turn creating a pressure difference. Like mentioned earlier, it might be possible that the model was still in a settling phase and the dimples would have disappeared in later time steps.

![Figure 4.5](image)

**Figure 4.5.** Showing lung nodal displacements at time 7.0554e-3 seconds. Unexplained dimples cover the lung walls. As the simulation was not allowed enough runtime to reach a large mass flow rate, it is possible that these are symptoms of the model settling out.
Figure 4.6. Showing the lung simulation velocity vectors through a vertical slice. One of the wall dimples is examined to show the resultant swirling air. This vortex created pressures that added to the dimpling phenomenon.
CHAPTER FIVE: SUMMARY AND FUTURE WORK

The purpose of this research was to validate the methods required to create a reliable computational FSI pulmonary simulation. The main contributions from this research reside in the outlined methods and how they were verified by physical experiments. When attempting to break new ground with research in the field of computational modeling, the validation is always the most important component.

The computational fluid and solid modeling programs Fluent and Abaqus were linked with the interpolation software MpCCI and simulations were shown to reliably mimic a balloon blowing up. Extensive comparisons were shown to verify the experimental and computational models of a balloon. A method for modeling the pulmonary system in discrete sections was discussed and a small portion of the lower lung was modeled with FSI.

A parallel direction could be envisioned for the next step in this program. First, particle deposition efficiency as a function of particle size should be computed for the small lung section presented and compared to classical static modeling. Second, the methodology to link multiple sections of the lung, passing the mass flow rate and back pressure between them each time step should be undertaken. When the capability becomes available with the program Abaqus, the MpCCI implicit FSI solver should be evaluated for its stability, and compared to the explicit solver used for this research. It would also be useful to investigate a completely different FSI modeler such as ADINA,
which already has the capability to solve FSI simulations implicitly. A similar computational balloon model could be used to validate the methods.
REFERENCES


N/a.


### APPENDIX A

**Fluent Journal File Example**

; Thesis Fluent Journal File
file read-case Balloon.msh
grid check

define models unsteady yes
define materials change-create air air ideal-gas , , , , ,
define user-defined interpreted-functions "MassFlowNew.c", , ,
report reference-values pressure 101325.4
report reference-values temperature 310.15
; define operating-cond gravity yes , -9.81 ,
define boundary-conditions mass-flow-inlet , yes yes yes yes , , 310.15 , , no yes

solve monitors residual plot yes
solve initialize set-defaults temperature 310.15
solve initialize initialize-flow
solve set time-step .001

; Set auto-save details
file auto-save case-frequency if-case-is-modified
file auto-save data-frequency 200
file auto-save retain-most-recent-files yes
file auto-save root-name balloon-iter

; iterate # of time steps, iterations per time step
solve dual-time-iterate 1 200

; save file
; file write-case-data , yes
; exit

**Fluent Mass Flow Rate Inlet UDF**

#include "udf.h"

DEFINE_PROFILE(inlet_mf, thread, position)
{
    face_t f;
    real t = CURRENT_TIME;
}
begin_f_loop(f, thread)
{
    F_PROFILE(f, thread, position) = 1.99*3*(t*t);
}
end_f_loop(f, thread)
Material Properties Input File

** Balloon Properties
**
** MATERIALS
**
*Shell Section, elset=BalloonStart, material=Balloon
1.905e-4,
**
*Material, name=Balloon
*Density
940.,
*Hyperelastic, Test Data Input, Poisson=0.49
*Uniaxial Test Data, smooth=3
0,0
899.8868718,0.05
1730.666462,0.1
2485.547198,0.15
3173.42201,0.2
3802.524387,0.25
4380.428376,0.3
4914.048584,0.35
5409.640177,0.4
5872.798879,0.45
6308.460973,0.5
6720.903301,0.55
7113.743265,0.6
7489.938824,0.65
7851.788496,0.7
8200.93136,0.75
8538.347051,0.8
8864.355766,0.85
9178.618257,0.9
9480.135839,0.95
9767.250382,1
10037.64432,1.05
10288.34064,1.1
10515.70288,1.15
10715.43517,1.2
**
Interactions Input File

** INTERACTION PROPERTIES
**
*Surface Interaction, name=GenInteraction, Pad Thickness=1.905e-4
1.905e-4
**
** INTERACTIONS
**
*Surface, type=ELEMENT, name=Interface
Interface, SPOS
*Surface, type=ELEMENT, name=BalloonStart
BalloonStart, SPOS
*Surface, type=ELEMENT, MAXRATIO, name=ContactFace
ContactFace, SPOS
**

Contact Input File

**
*CONTACT PAIR, MECHANICAL CONSTRAINT=KINEMATIC
ContactFace
**

Main Input File

*Heading
**
** Balloon Geometry File created by Justin Jacobs
**
** LOAD THE GEOMETRY FILE
*Include, Input = BalloonStartHM.inp
**
** MATERIAL DEFINITIONS
*Include, Input = BalloonMaterial.inp
**
** INTERACTION DEFINITIONS
*Include, Input = BalloonInteractions.inp
**
*Step, name=Dilation
*Dynamic,explicit
,10.
**
**Fixed Mass Scaling, Factor=1
**
** CONTACT DEFINITIONS
*Include, Input = BalloonContact.inp
**
** APPLY THE BC'S
**
*BOUNDARY, OP=NEW
  Top, 1, 3, 0.0
**
** LOADS TO BE USED FOR TESTING IN ABAQUS WITHOUT FSI:
**
**Amplitude, Definition=Smooth Step, Name=SmoothStep
**0,0,10,1.
**
**Dsload, Amplitude=SmoothStep
**Interface, P, -0.0033
**
*Output, Field, variable=preselect, time interval=1e-3
**
*End Step
**