Flexible Slosh Diaphragm Modeling and Simulation in Propellant Tanks

by

Darren Vaughn Levine

Bachelor of Science
Aerospace Engineering
Florida Institute of Technology
2012

A thesis
Submitted to Florida Institute of Technology
in partial fulfillment of requirements
for degree of

Master of Science
In
Aerospace Engineering

Melbourne, Florida
May, 2014
© Copyright 2014 Darren Vaughn Levine

All Rights Reserved

The author grants permission to make single copies
We, the undersigned committee, hereby recommend that the attached document be accepted as fulfilling in part the requirements of the degree of Master of Science in Aerospace Engineering

Flexible Slosh Diaphragm Modeling and Simulation in Propellant Tanks

A thesis by Darren Vaughn Levine

Daniel R. Kirk, Ph.D., Major Advisor
Professor
Mechanical and Aerospace Engineering

Hector M. Gutierrez, Ph.D.
Professor
Mechanical and Aerospace Engineering

James R. Brenner, Ph.D.
Assistant Professor
Chemical and Biomedical Engineering

Hamid Hefazi, Ph.D.
Professor and Department Head
Mechanical and Aerospace Engineering
Abstract

Flexible Slosh Diaphragm Modeling and Simulation in Propellant Tanks

By

Darren Vaughn Levine

Advisor: Daniel R. Kirk, Ph. D.

Liquid sloshing and propellant distribution is an important field of research for aerospace applications such as launch vehicles and spacecraft. The propellant mass behavior can greatly influence vehicle dynamics, altering trajectories and structural loading distributions. In order to combat these concerns some propellant tanks employ thin elastomeric diaphragms to separate the fuel from the gas volumes and restrict the fluid's motion. The diaphragm’s flexible behavior is generally highly nonlinear, and various propellant fill levels and acceleration conditions can cause large deflections resulting in complex buckling and folding patterns. When complex and non-uniform deformations occur, there is potential for diaphragm wear and damage to occur at contact and bending regions. Therefore the problem of simulating a diaphragm coupled with a fluid is essential to mass distribution prediction.
and damage prediction. This thesis explores and implements several fluid coupled diaphragm modeling techniques. Additionally, a comparison with experimentally obtained 3D scans of a diaphragm is made. The results from these analyses show that by using a soft body deformation model, membrane displacement behavior can be predicted with an average error of 7.1% with a standard deviation of 3.3%.
TABLE OF CONTENTS

Abstract...........................................................................iii

TABLE OF CONTENTS.......................................................v

LIST OF FIGURES.........................................................ix

LIST OF TABLES............................................................xvi

LIST OF ABBREVIATIONS...............................................xvii

LIST OF SYMBOLS.........................................................xviii

ACKNOWLEDGEMENT.....................................................xxi

CHAPTER 1: INTRODUCTION..............................................1

1.1 BACKGROUND .......................................................1

1.2 MOTIVATION ....................................................... 11

1.3 OBJECTIVES ....................................................... 12

1.4 APPROACH ..........................................................13

1.5 THESIS OVERVIEW ............................................ 14

CHAPTER 2: SLOSH DIAPHRAM OVERVIEW AND THEORY...........15

2.1 INTRODUCTION ..................................................... 15

2.2 VEHICLE DYNAMICS COUPLING ............................... 15

2.3 FLUID AND DIAPHRAGM COUPLING .......................... 19

2.4 STRUCTURAL INTERACTIONS AND FOLDING ................. 25
4.2.4 MATERIAL BEHAVIOR SIMULATION ..................... 106

4.2.5 MATLAB RESULTS ..................................... 113

CHAPTER 5: EXPERIMENTALLY OBTAINED 3D SCANS OF INVERTING SLOSH DIAPHRAGM ..................................... 116

CHAPTER 6: RESULTS OF COMPARISON ............................... 119

CHAPTER 7: CONCLUSIONS AND FUTURE WORK ......................... 131

7.1 CONCLUSIONS ........................................... 131

7.2 FUTURE WORK ............................................ 133

REFERENCES .................................................... 136

APPENDIX A: ANSYS MODEL DEFORMATION RESULTS ............. 146

APPENDIX B: MATLAB MODEL – MAIN PROGRAM DRIVER ........ 155

APPENDIX C: MATLAB MODEL – GRID GENERATOR ................ 160

APPENDIX D: MATLAB MODEL – NODE AREA ...................... 164

APPENDIX E: MATLAB MODEL – GRID LENGTH MAPPING ......... 165

APPENDIX F: MATLAB MODEL – STRUCTURAL FORCES .......... 167

APPENDIX G: MATLAB MODEL – IN PLANE STRAIN LIMITING CONSTRAINTS .................................................. 169

APPENDIX H: MATLAB MODEL – OUT OF PLANE STRAIN LIMITING CONSTRAINTS .............................................. 173

APPENDIX I: MATLAB MODEL – COLLISION HANDLING ......... 175
APPENDIX J: MATLAB MODEL - DATA PROCESSING .............177

APPENDIX K: MATLAB MODEL - SIMULATED DIAPHRAGM INVERSION
  TOPOLOGY RESULTS........................................187

APPENDIX L: TANK AND SCANNING APPARATUS .................193
LIST OF FIGURES

Fig. 1.1 Spherical satellite propellant tank with wicking vane propellant management devices [6]-[7] ............ 2

Fig. 1.2 An elastomeric diaphragm tank schematic of standard components [8] ............................... 3

Fig. 1.3 An AF-E-332 elastomeric diaphragm and retaining ring for a composite hydrazine propellant tank [9] .... 4

Fig. 1.4 “Tensile Strain (ε) versus time to rupture (t_b) in hydrazine as a function of temperature for AF-E-332 elastomer” [11] ........................................... 7

Fig. 1.5 Experimental modulus of elasticity (MPa) data and curve fit as a function of diaphragm thickness (mm) at the glass transition temperature (T_g) and T=20°C [12]... 8

Fig. 2.1 Schematic for multiple slosh mode model [17].... 17

Fig. 2.2 Schematic for a gradient based static equilibrium model of a discretized diaphragm element ............ 22

Fig. 2.3 Fold types with representative symbols: A) pipe fold; B) diaper fold; C) half-lock fold; D) zigzag fold; E) drop fold; F) spiral fold; G) inert fold [19] ..... 29

Fig. 2.4 Sketched examples of the seven fold types [19].. 30

Fig. 2.5 Diaphragm in a 40 inch diameter tank, (a) 50% fill fraction, (b) 75% fill fraction, (c) and (d) show
examples of diaphragm folding, with fold types labeled.

Images courtesy of NASA. ................................. 31

Fig. 2.6 Double fold in a thin elastomeric member [11]... 32

Fig. 2.7 Schematic of a nine node section for a spring-mass n-particle body system using all three spring modes .. 46

Fig. 3.1 Displacement profile trends for circular diaphragms of varying diameters with clamped edges, ethylene-propylene terpolymer material, constant thickness = 0.07 in. ................................. 55

Fig. 3.2 Displacement profile trends for flat circular diaphragms in spherical tanks of 16.5" and 40" diameters with clamped edges, ethylene-propylene terpolymer material, constant thickness = 0.07 in., water fill fraction is measured by volume ....................... 59

Fig. 3.3 Nondimensional displacement profile trends for flat circular diaphragms in spherical tanks of 16.5" and 40" diameters with clamped edges, ethylene-propylene terpolymer material, constant thickness = 0.07 in., water fill fraction is measured by volume ............. 61

Fig. 3.4 Maximum displacement (f) trends for flat circular diaphragms in spherical tanks with clamped diaphragm edges, ethylene-propylene terpolymer, constant thickness = 0.07 in., water fill fraction is measured by volume. 63
Fig. 4.1 ANSYS Transient Structural model block schematic 69
Fig. 4.2 Example of a rough surface contact model, A is the undeformed geometry, B is the displaced geometry .... 74
Fig. 4.3 Grid generated for ANSYS Simulations ............ 77
Fig. 4.4 Results of ANSYS Transient Structural model numbered in order of occurrence, maximum deflection is 0.4263 m, lower scale bar uses 0.1 m increments ...... 82
Fig. 4.5 ANSYS Transient Structural and Fluent coupled model block schematic .......................... 85
Fig. 4.6 Cross section of FSI geometry including Fluent pressure result for a static tank in gravity, A) membrane and tank geometry (pressure is applied here), B) tank geometry, C) fluid outlet or inlet, D) pressure in tank .............................................. 88
Fig. 4.7 Azimuthal equidistant projection (left), gnomonic projection of an inscribed cube (right) ............ 96
Fig. 4.8 A single patch from the gnomonic projection method (left), expanded into a hemisphere (right) ............ 97
Fig. 4.9 Array of grid points in spiral pattern for angle equalization ................................................. 99
Fig. 4.10 Unaltered grid (left), corrected grid (right). 100
Fig. 4.11 Two examples of complex topologies for simulated diaphragms with high allowable strain rates ...... 101
Fig. 4.12 Stress Strain trends for linear elastic and hyperelastic materials [47] .......................... 109

Fig. 5.1 3D scan of diaphragm at 70% fill volume........ 117

Fig. 6.1 Dissection schematic of elastomer diaphragm's connection to inside of propellant tank [12] .......... 120

Fig. 6.2 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 40% fill volume ......................... 122

Fig. 6.3 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 50% fill volume .......................... 123

Fig. 6.4 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 60% fill volume .......................... 124

Fig. 6.5 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 70% fill volume .......................... 125

Fig. 6.6 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 80% fill volume .......................... 126

Fig. 6.7 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 90% fill volume .......................... 127
Fig. 6.8 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 100% fill volume ....................... 128

Fig. 6.9 Simulated topology and experimental topology measurement error, as a function of fill volume percent, with an average of 7.1% error ....................... 129

Fig. A.1 ANSYS Transient Structure model deformation, max deformation = 0.0072623 m ........................... 146

Fig. A.2 ANSYS Transient Structure model deformation, max deformation = 0.022054 m ............................ 147

Fig. A.3 ANSYS Transient Structure model deformation, max deformation = 0.045602 m ............................ 148

Fig. A.4 ANSYS Transient Structure model deformation, max deformation = 0.06871 m ............................. 149

Fig. A.5 ANSYS Transient Structure model deformation, max deformation = 0.13522 m .............................. 150

Fig. A.6 ANSYS Transient Structure model deformation, max deformation = 0.31408 m ................................. 151

Fig. A.7 ANSYS Transient Structure model deformation, max deformation = 0.36669 m ................................. 152

Fig. A.8 ANSYS Transient Structure model deformation, max deformation = 0.40974 m ................................. 153
Fig. A.9 ANSYS Transient Structure model deformation, max deformation = 0.4263 m ........................................ 154

Fig. K.1 Soft body deformation simulation of diaphragm
inverting at 0% Fill Volume .............................. 187

Fig. K.2 Soft body deformation simulation of diaphragm
inverting at 10% Fill Volume ............................. 187

Fig. K.3 Soft body deformation simulation of diaphragm
inverting at 20% Fill Volume .............................. 188

Fig. K.4 Soft body deformation simulation of diaphragm
inverting at 30% Fill Volume .............................. 188

Fig. K.5 Soft body deformation simulation of diaphragm
inverting at 40% Fill Volume .............................. 189

Fig. K.6 Soft body deformation simulation of diaphragm
inverting at 50% Fill Volume .............................. 189

Fig. K.7 Soft body deformation simulation of diaphragm
inverting at 60% Fill Volume .............................. 190

Fig. K.8 Soft body deformation simulation of diaphragm
inverting at 70% Fill Volume .............................. 190

Fig. K.9 Soft body deformation simulation of diaphragm
inverting at 80% Fill Volume .............................. 191

Fig. K.10 Soft body deformation simulation of diaphragm
inverting at 90% Fill Volume .............................. 191
Fig. K.11 Soft body deformation simulation of diaphragm inverting at 100% Fill Volume ....................... 192

Fig. L.1 Side view of slosh diaphragm 16.5 inch diameter fuel tank, A) tank, B) gas port, C) liquid port, D) 80-20 aluminum frame for center of gravity testing, E) LVDT displacement sensors with spring mounts ............. 193

Fig. L.2 Front view of slosh diaphragm 16.5 inch diameter fuel tank, A) tank, B) gas port, C) liquid port, D) 80-20 aluminum frame for center of gravity testing, E) LVDT displacement sensors with spring mounts ............. 194

Fig. L.3 Xbox 360 Kinect infrared 3D sensor with an electric motor mounted to the sensor .................. 195

Fig. L.4 Sample photo of slosh diaphragm 16.5 inch diameter tank at 70% fill volume during the 3D scanning experiment fuel tank, A) gas port, B) cinder block support for Xbox Kinect 3D sensors, C) tank, D) diaphragm, E) 80-20 aluminum support frame ......... 195

Fig. L.5 Photo of the entire 3D scanning experiment setup, A) 6 Xbox Kinects, B) 3D scan processing workstation, C) cinderblocks sensor mounts, D) propellant tank sitting on 80-20 aluminum support frame ..................... 196
LIST OF TABLES

Table 1.1 PSI AF-E-332 elastomeric diaphragm tank usage history [8] ........................................... 6

Table 4.1 Matlab model input conditions.................... 114
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMD</td>
<td>Propellant Management Device</td>
</tr>
<tr>
<td>FEA</td>
<td>Finite Element Analysis</td>
</tr>
<tr>
<td>PSI</td>
<td>Pressure Systems, Inc.</td>
</tr>
<tr>
<td>ATK</td>
<td>Alliant Techsystems, Inc.</td>
</tr>
<tr>
<td>FSI</td>
<td>Fluid Structure Interaction</td>
</tr>
<tr>
<td>EPR</td>
<td>Ethylene-Propylene Terpolymer Rubber (AF-E-332)</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>SLE</td>
<td>System of Linear Equations</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS

\[ z \] Subscript: Transverse Component

\[ x \] Subscript: Axial Component

\[ a \] Acceleration of the center of the fuel tank, radius, or constant variable

\[ i \] Slosh mode index value

\[ \varepsilon_i \] Slosh mode damping constant for \( i^{th} \) mode

\[ v \] Velocity of the center of the fuel tank

\[ l_i \] Pendulum length for \( i^{th} \) mode pendulum

\[ m_i \] Pendulum mass for \( i^{th} \) mode pendulum

\[ h_i \] Distance from the tank center to the pendulum pivot point for \( i^{th} \) mode pendulum

\[ m_f \] Total fuel mass

\[ m \] Mass of the spacecraft

\[ \theta \] Attitude angle between the spacecraft’s coordinate system and the fixed reference frame

\[ \psi_i \] Angle between the \( i^{th} \) mode pendulum arm and the transverse axis of the spacecraft

\[ I_i \] Moment of inertia for the \( i^{th} \) mode pendulum mass

\[ I \] Moment of inertia for the spacecraft or area moment of inertia for a bending member

\[ F \] Force or thrust from a gimbaled engine
\( \delta \) Angle between the thrust vector of a gimbaled engine and the transverse axis of the spacecraft

\( M \) Moment force around the center of mass of the spacecraft

\( d \) Distance to the gimbaled engine from the center of mass of the spacecraft, flexural rigidity, or deflection of a bending member

\( b \) Distance from the center of mass of the spacecraft to the center of the fuel tank

\( T_{\text{end}} \) End time

\( C \) Easing function constant

\( E \) Young's modulus

\( t \) Plate thickness or thickness of the diaphragm

\( v \) Poisson's ratio

\( w \) Deflection

\( r \) Radial position or local radius

\( P \) Pressure

\( P_{\text{max}} \) Maximum pressure

\( f \) Maximum deflection

\( D \) Diameter

\( \varepsilon_i \) Residual strain

\( \alpha, \beta, \gamma \) Deflection formula coefficients

xix
\( R \)  Radius

\( X, Y, Z \)  Global coordinates

\( x, y, z \)  Local coordinates

\( \alpha \)  Grid line angle around the y axis

\( \beta \)  Grid line angle around the x axis

\( A \)  Area

\( i \)  The x direction index in the computational domain

\( j \)  The y direction index in the computational domain

\( F_b \)  Bending force

\( k \)  Equivalent spring stiffness
ACKNOWLEDGEMENT

I would like to thank my advisor, faculty, staff, family, and friends for their support and encouragement throughout my educational career.
CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Liquid sloshing and propellant distribution is an important field of research for aerospace applications such as launch vehicles and spacecraft due to microgravity and launch conditions [1]-[2]. The propellant behavior can greatly influence vehicle dynamics - altering trajectories and structural loading distributions - as well as influencing cyclic wear, draining problems and tank overpressure events. In order to combat these concerns, Propellant Management Devices (PMDs) are often employed to manage propellant behavior [3]-[5]. PMDs redistribute propellant in order to maintain adequate propellant flow to an intake aperture; additionally, PMDs may provide various levels of slosh damping during vehicle maneuvers. In these ways, fuel tank dynamics are influenced by PMD design, and therefore prediction of fuel tank dynamics when a PMD is employed is necessary to sufficiently account for the propellant volume’s influence on vehicle dynamics. There are many different types of propellant tank PMDs, each relying on a variety of principles such as wicking geometries and surfaces, as well as baffles, diaphragms,
and slosh suppressing inner tank structures. An example of a PMD designed for a geosynchronous satellite with wicking vanes and a sponge trap is shown in Fig. 1.1 [6]-[7].

![Diagram of a PMD](image)

**Fig. 1.1 Spherical satellite propellant tank with wicking vane propellant management devices [6]-[7]**

Vehicle applications for PMD technologies occasionally require high levels of propellant distribution control and propellant environmental isolation; in these cases, a flexible elastomeric diaphragm is often employed to enclose the liquid. These elastomeric diaphragms are often called "Slosh Diaphragms" because they passively restrict fuel sloshing dynamics by acting as both a spring and baffle on the fluid surface. An elastomeric diaphragm is inherently better at propellant slosh damping than other PMDs such as vanes; this allows the center of gravity in a tank to be controlled for to a much higher degree of confidence than
the alternatives. An example diagram for a spherical propellant tank with a diaphragm is shown in Fig. 1.2 [8].

![Diagram of a spherical propellant tank with a diaphragm.](image)

**Fig. 1.2 An elastomeric diaphragm tank schematic of standard components [8]**

A diaphragm, in addition to damping slosh and ensuring fuel ingestion through propellant redirection, can separate fuel and gas media that may prevent chemical reactions, dilution, contamination, or phase changes from occurring. This isolation of propellant is especially important for highly reactive fuels such as monopropellants. Unlike most rigid PMDs, propellant diaphragms need to be flexible in order to conform to the surface of the propellant volume as it drains or fills. The diaphragm’s flexible behavior is highly nonlinear, and various propellant fill levels and acceleration conditions can cause large deflections resulting in complex buckling and folding patterns. Elastomeric diaphragms are typically used in spherical or
cylindrical tanks with hemispherical caps due to geometric constraints in an effort to minimize boil-off of cryogenic liquid. A sample picture of a less typical cylindrical diaphragm with a hemispherical cap shape is shown in Fig. 1.3 [9].

![Fig. 1.3 An AF-E-332 elastomeric diaphragm and retaining ring for a composite hydrazine propellant tank [9]](image)

The design of a slosh diaphragm is simplistic. It is typically made from a 0.07 inch thick elastomeric material, in the shape of the propellant hemisphere of the fuel tank. Small ribs made from the same material are also included in order to add extra stiffness for additional slosh damping.
These ribs are included in the mold of the entire diaphragm, which is made as a single continuous piece.

Due to their simple nature, elastomeric diaphragms have been employed in fuel tanks since the start of the space industry. The majority of elastomeric tanks produced and designed have been contracted by the company: Pressure Systems, Inc. (PSI). PSI is a company specializing in the design and production of spacecraft pressure and propellant tanks and is the world's largest independent supplier of welded titanium tanks. In 2004 PSI was acquired by Alliant Techsystems, Inc. (ATK). The majority of the PSI tanks produced use the elastomer "ethylene-propylene terpolymer" also known as "AF-E-332" for the diaphragm material and are made for the monopropellant hydrazine. A history of these tanks use is given in Table 1.1, a history in which no failure was ever recorded [8].
Table 1.1 PSI AF-E-332 elastomeric diaphragm tank usage history
[8]

<table>
<thead>
<tr>
<th>Tank Size</th>
<th>QTY.</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.4&quot; Sphere</td>
<td>28</td>
<td>AEROS, IUS, HCMM, SCATHA, SHUTTLE</td>
</tr>
<tr>
<td>12.9&quot; Sphere</td>
<td>94</td>
<td>CTS, OTS, GPS, APPLE, ITV, DMSP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(TIROS), GEOSAT, MSTI, (WORLDVIEW)</td>
</tr>
<tr>
<td>15.4&quot; Sphere</td>
<td>11</td>
<td>OTS</td>
</tr>
<tr>
<td>16.5&quot; Sphere</td>
<td>219</td>
<td>PIONEER, ATS, NOVA, BSE, SEASAT, MMS,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOPEX, P-80, ECS, TELECOM, GPS, SKYNET 4,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NATO IV, CRRES, TOS, TITAN II, LANDSAT, MESUR PATHFINDER,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(MUPS)</td>
</tr>
<tr>
<td>17.4 Sphere</td>
<td>9</td>
<td>ISPM, SAX</td>
</tr>
<tr>
<td>19&quot; Sphere</td>
<td>3</td>
<td>EXOSAT</td>
</tr>
<tr>
<td>20.8&quot; Sphere</td>
<td>53</td>
<td>IUS</td>
</tr>
<tr>
<td>22.1&quot; Sphere</td>
<td>231</td>
<td>P-95, VIKING, HEAO, FLTSATCOM, DSCS, (CENTAUR),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RADARSAT, (STEP), TOMS, NEAR</td>
</tr>
<tr>
<td>23.1&quot; x 25.7&quot; Long</td>
<td>8</td>
<td>EURECA</td>
</tr>
<tr>
<td>28&quot; Sphere</td>
<td>54</td>
<td>VOYAGER, SHUTTLE, MMS-ATK, UARS, TOPEX, ERBS, VRM,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TITAN III, CASSINI, (INDOSTAR)</td>
</tr>
<tr>
<td>36&quot; x 47&quot; Long</td>
<td>5</td>
<td>GRO</td>
</tr>
<tr>
<td>40&quot; Oblate Spheroid</td>
<td>28</td>
<td>TDRSS, SOHO, P80, COBE, TRMM, EOS, (EST-VII)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>743</td>
<td></td>
</tr>
</tbody>
</table>

Other less common elastomeric diaphragm materials that have been used in older systems are "MIL-R-83412" and the "Royal Ordnance’s D11". It has been observed that these elastomers have the potential to leach some of the material’s primary constituents into the hydrazine over long periods of time, inducing performance degradation. In order to address this issue development of other compatible elastomeric materials that do not degrade over time are ongoing. One such material that has been developed with some success is the elastomer "SIFA 35" [10].
The experimental portion of this thesis employed a diaphragm made from AF-E-332. The material's time to failure is a function of temperature and strain, this function is shown in Fig. 1.4 [11].

Fig. 1.4 "Tensile Strain (ε) versus time to rupture (t_b) in hydrazine as a function of temperature for AF-E-332 elastomer" [11]
The trends in Fig. 1.4 show that at lower temperatures and lower strains, the time to rupture decreases. The material’s behavior, such as stiffness is also affected by the material thickness. Fig. 1.5 shows a plot of modulus of elasticity as a function of thickness, showing a general trend towards an increase in thickness leading to an increase in stiffness [12].

Fig. 1.5 Experimental modulus of elasticity (MPa) data and curve fit as a function of diaphragm thickness (mm) at the glass transition temperature ($T_g$) and $T=20^\circ C$ [12]

A propellant tank employing an elastomeric diaphragm functions through the principle of positive expulsion.
Positive expulsion is a technique whereby a pressure differential is created across the diaphragm by pressurizing the gas hemisphere of the tank. The expansion of the gas in the pressurant hemisphere of the tank forces propellant volume to be expelled. This technique also allows for propellant expulsion under high or low accelerations by modifying the pressure in the pressurant hemisphere to compensate.

The diaphragm is also oriented in such a way that will allow for a majority of the propellant to be expelled when the tank is drained. As seen in Fig. 1.2, the diaphragm hemisphere's undeformed shape is the same as that of the propellant hemisphere, allowing the diaphragm to unfold and lay flush with the inner surface of the propellant hemisphere. These strategies allow for propellant expulsion efficiencies of 99.9% or greater when an elastomeric diaphragm is employed [8].

While the performance of a fuel tank with a slosh diaphragm system is superior to vane PMDs in the ways discussed above, its cost is often initially larger due to custom tooling such as the diaphragm mold. However, these larger costs are non-recurring. Only redesigns will carry
the additional development costs, and so over time, slosh diaphragms become cost competitive to vane PMDs [8].

It is, therefore, concluded that slosh diaphragms as an existing technology are viable and worthwhile. However, to advance the technology, predictive models must be created for the diaphragms. The existing research into modeling an elastomeric slosh diaphragm is limited. In fact, while many Fluid Structure Interaction (FSI) type models exist, only one study in which a numerical model for a slosh diaphragm was attempted could be found. The FSI diaphragm model was created by Sances et al. [13]. Although the FSI model was never validated, the material was treated as iso-elastic to simplify the structural calculations, and the material's unstressed-resting states were incorrectly modeled as highly deformed.

The Sances et al. model, however, shows that the capability exists to couple the fluid-structural interactions. Their model concluded that the fluid behavior when constrained by a diaphragm is found to be highly effected by the viscoelastic nature of the diaphragm and the increase in stiffness effects at the liquid-diaphragm interface. This increases the natural slooshing frequency of
the coupled system. This thesis purposes to create a more complete diaphragm model using hyperelastic theory to model the elastomer as well as explore new types of numerical techniques available for which models can be created.

1.2 MOTIVATION

The characterizations of slosh diaphragm behavior serve to forward several functional goals for the development of next-generation slosh diaphragms. Current slosh diaphragms are highly susceptible to wear due to rubbing when folds occur; non-uniform folds in particular are undesirable because they can damage the diaphragm internally. Diaphragms that deform predictably under loading are highly sought after for these reasons. Diaphragm mass optimization is also a joint goal; obtaining predictable diaphragm behavior while minimizing diaphragm weight is essential to competitive design practices when the diaphragm structure is used in space systems.

Currently, these forms of optimized slosh diaphragms are nonexistent. Additionally, rigorous numerical analysis of slosh diaphragm propellant tanks have not been made, so reliability assessments are customarily made using qualitative assessments of the behavior and failure modes.
In order to develop these technologies, new models of slosh diaphragms are required in order to perform numerical design and optimization. These models require the analysis of large pressure driven deformations in hyperelastic materials. The poor numerical stability of computational schemes used to model these types of conditions typically makes them exceptionally difficult and time-consuming to successfully create. Therefore, there is motivation for the creation of predictive diaphragm models as well as for a procedure for the reliable creation of these models.

1.3 OBJECTIVES

The purpose of this thesis is to enable the modeling of slosh diaphragm behavior in order to predict where and when folding and wear of the diaphragm is likely to occur. In order to reach this goal, the following objectives are approached:

1) Create a structural model of the diaphragm using commercially available software (ANSYS Mechanical). The model must be capable of predicting large hyperelastic deformations in membranes;

2) Create a fluid model of the fuel tank enclosed by the diaphragm and tank walls using commercially available
software (ANSYS Fluent). The model should be capable of predicting the draining/filling and sloshing behavior of the propellant volume;
3) Couple the fluid and structural models in order to simulate the entire tank system (ANSYS Workbench);
4) Create a custom model of the diaphragm using modern numerical methods;
5) Compare the custom and ANSYS models to experimentally obtained 3D scans of a diaphragm; and
6) Make conclusions about how improvements to the modeling techniques and procedures might be made.

1.4 APPROACH

In order to model the behavior of a slosh diaphragm, ANSYS's Workbench Suite was used to create a structural, fluid, and coupled model of the membrane-tank system. A literature search was conducted to evaluate the current state of hyperelastic membrane simulations, and to determine the best methods for approaching the creation of a complete membrane model.

An alternative approach was jointly undertaken for simulating hyperelastic materials using soft body particle models. These relatively new methods of creating physics
based particle cloth simulations were researched, and the cloth based schemes were modified in order to be applicable to thin hyperelastic materials. The model was coded in Matlab. The resulting particle-based hyperelastic membrane model was coupled with fluid and body forces in order to complete the particle based membrane-tank model.

The experimental data utilized was produced using Florida Tech's 3D scanning equipment. 3D scans of a 16.5 inch diameter hemispherical 0.07 inch thick AF-E-332 elastomeric membrane. The 3D scans are used to evaluate the models created in this thesis.

1.5 THESIS OVERVIEW

This thesis begins with an overview of propellant tank slosh diaphragm background and an introduction to diaphragm modeling problem in Chapter 1. Next Chapter 2 includes an overview of the theory behind how the slosh diaphragm’s behavior couples with vehicle and propellant dynamics. Chapters 3 and 4 cover the modeling of the diaphragm both analytically and computationally, while Chapter 5 presents 3D scans of real diaphragms. The final sections, Chapters 6 and 7 cover the results from a comparison of models and experimental data, and the conclusions.
CHAPTER 2: SLOSH DIAPHRAGM OVERVIEW AND THEORY

2.1 INTRODUCTION

Slosh diaphragm’s behavior can modify the behavior of the propellant volume by influencing how the propellant moves within a fuel tank when excited by a force. The resulting propellant center of mass and its transfer of momentum to the tank walls couples the vehicle and propellant dynamics. This vehicle-propellant-diaphragm coupling can play a significant role in the form of unplanned deviations to the trajectory of the spacecraft. In order to predict and prevent undesired perturbations to the spacecraft’s trajectory, the coupling of the vehicle, fluid, and diaphragm components must be taken into account as well as an accurate simulation of each system.

2.2 VEHICLE DYNAMICS COUPLING

Coupled slosh-vehicle systems are typically modeled as a rigid spacecraft body with an active pendulum body attached or a spring-damper mass attached [14], [15]. The natural frequency of the pendulum model takes into account a change in acceleration (gravity) just like an actual sloshing fluid, while the linear spring-mass systems do not [16].
These models represent the most influential slosh mode for a given fuel tank as the attached mass. Various alterations to these base-configuration models exist, such as more complex models that included multiple pendulums, one for each prominent slosh mode [17]. Feedback controllers that are built using these models allow for the effects of propellant sloshing to be anticipated and corrected. The most common type of these coupled slosh-vehicle models are the mass-pendulum model because of its better accuracy, the equations of motion for a mass pendulum slosh-vehicle coupled model with N slosh modes of vibration are [17]:

**Eqn. 2.1**

\[(m + m_f) \ddot{a}_x + \sum_{i=1}^{N} m_i l_i (\ddot{\theta} + \ddot{\psi}_i) \sin \psi_i + \ddot{\theta} \ddot{\theta} + \sum_{i=1}^{N} m_i l_i (\theta + \psi_i)^2 \cos \psi_i = F \cos \delta\]

**Eqn. 2.2**

\[(m + m_f) \ddot{a}_z + \sum_{i=1}^{N} m_i l_i (\ddot{\theta} + \ddot{\psi}_i) \cos \psi_i + \ddot{\theta} \ddot{\theta} - \sum_{i=1}^{N} m_i l_i (\theta + \psi_i)^2 \sin \psi_i = F \sin \delta\]

**Eqn. 2.3**

\[\ddot{\theta} - \sum_{i=1}^{N} m_i l_i h_i \left[ (\ddot{\theta} + \ddot{\psi}_i) \cos \psi_i - (\theta + \psi_i)^2 \sin \psi_i \right] + \ddot{\theta} \ddot{\theta} - \sum_{i=1}^{N} \epsilon_i \dot{\psi}_i = M + F(b + d) \sin \delta\]
Eqn. 2.4

\[
(l_i + m_i l_i^2) (\dot{\theta} + \dot{\psi}_i) - m_i l_i \dot{h}_i (\dot{\theta} \cos \psi_i + \dot{\theta}^2 \sin \psi_i) + m_i l_i (a_x \sin \psi_i + a_z \cos \psi_i) + \epsilon_i \dot{\psi}_i = 0
\]

where \((a_x, a_z) = (\dot{v}_x + \dot{\theta} v_x, \dot{v}_z - \dot{\theta} v_x)\), \(\overline{m b} = m b - \sum_{i=1}^{N} m_i l_i\), and \(\overline{l} = l + l_0 + m b^2 + m_0 h_0^2 + \sum_{i=1}^{N} m_i h_i^2\). A diagram for this multiple-slosh-mode/multiple-pendulum model for a spacecraft is shown in Fig. 2.1.

Fig. 2.1 Schematic for multiple slosh mode model [17].

Subscript \(z\) refers to the transverse component, while subscript \(x\) refers to the axial component. \(a\) is the
acceleration of the center of the tank. Subscript $i$ refers to $i^{th}$ slosh vibration mode where $i = 0$ refers to the bulk static mass of the propellant. The $\epsilon_i$ is the sloth damping constant for $i^{th}$ mode. $v$ is the velocity of the center of the fuel tank. $l_i$ and $m_i$ are the pendulum lengths and masses respectively, for each $i^{th}$ mode. $h_i$ is the distance from the center of the fuel tank to the pendulum axis for each $i^{th}$ pendulum. $m_f$ is the total fuel mass, while $m$ is the mass of the spacecraft. $\theta$ is the attitude angle between the spacecraft’s coordinate system and the fixed reference frame. $\psi_i$ is the angle between the pendulum arm and the transverse axis of the spacecraft. $I_i$ is the moment of inertia for the pendulum mass for $i^{th}$ mode, and $I$ is the moment of inertia for the spacecraft. $F$ is the thrust from a gimbaled engine at a deflection angle of $\delta$, and $M$ is the moment force around the center of mass of the spacecraft. $d$ and $b$ are the distance to the gimbaled engine from the center of mass and the distance from the center of mass to the center of the fuel tank, respectively.

By utilizing a slosh diaphragm in a spacecraft, the designers introduce additional damping that causes a shift in slosh frequency and magnitude, modifying the terms $\psi_i, l_i,$
and $m_l$. These shifts to frequency and damping – once measured – can be accounted for in the equations of motion for the spacecraft in question, allowing for a more complete model of the vehicle dynamics to be created. These shifts in slosh mode dynamics are accounted for by creating the equivalent dynamic model of slosh using a mass pendulum system. This can be accomplished by using analytical slosh motion equations or measured data. First the equivalent pendulum length is found by knowing the system’s frequency, and knowing that the frequency is equal to the square root of gravity over length for a pendulum. The excitation amplitude is then scaled to the pendulum length, and the slosh mass term can be computed using the experimentally measured or analytically derived slosh force term [16].

2.3 FLUID AND DIAPHRAGM COUPLING

The PMD diaphragms used in fuel tanks are typically thin elastomeric hemispherical diaphragms. Modeling these diaphragms and their influence on the fluid volume they enclose presents a challenge as the fluid and diaphragm’s behaviors are strongly coupled. In order to solve for the behavior of these types of systems, the operating conditions of a fuel tank in a space craft are considered.
The two physical cases that arise from these considerations, which are of interest in the fluid and diaphragm coupling problem are:

1) If the fluid volume is considered to be at rest.
2) If the fluid volume is considered to be dynamic.

The first case would apply to static conditions or for very gradual changes in the fluid volume such as a slow draining process where the dynamic pressure of the fluid and air volumes induced by the draining motion does not reach an order of magnitude similar to that of the hydrostatic and static pressures. If the static forces exceed the dynamic forces significantly in these systems, it is thought to be reasonable to assume that the result of a static equilibrium analysis is sufficient to solve the diaphragm boundary’s shape, and the fluid volume is then also considered static and resolved in terms of center of mass and effect on spacecraft inertial changes. A good analogy for this type of system is a liquid sessile droplet, where liquid surface tension acts as an analogous elastic gas-liquid interface that influences the droplet’s shape at equilibrium.
The second (dynamic) case would be applicable to launch conditions or trajectory changes in the spacecraft — any conditions which would cause the order of magnitude of the dynamic forces to increase to influence the membrane’s shape and the fluid’s contribution to the tank’s momentum. The dynamic case is of interest to predicting diaphragm influence on a spacecraft's trajectory changes.

In the first case the fluid body's interaction with the membrane can be approximated as pressure gradient. This approach for modeling the fluid volume simplifies many of the problems a fully coupled model would face by not solving the Navier-Stokes equations using computational methods. This approach of modeling the fluid as a pressure gradient was undertaken in order to develop the ANSYS and Matlab model for cases which could be approximated as static. A free body diagram of this pressure gradient approximation is shown in Fig. 2.2.
Fig. 2.2 Schematic for a gradient based static equilibrium model of a discretized diaphragm element

In Fig. 2.2 variable "a" is the body force, $\rho$ is fluid density, $P_l$ is the total liquid pressure at distance $d$ from the center of the tank, $P_g$ is the total gas pressure, $h$ is the equivalent fluid height, $d$ is the distance from the center of the tank to a diaphragm element, $Z$ is the main axis of the tank, and $\theta$ is the angle between the body force vector "a" and the $Z$ axis of the tank. $\theta$ is equal to zero, and $a = 9.81 \text{ m/s}^2$ in the cases where earth is used as the static frame of reference and the tank is upright. The pressure differential on a membrane element as a function
of distance from the center of the tank in the body force vector direction.

Note that the equivalent fluid height is not equal to the maximum fluid height, but is rather a representative term used to describe the height of the fluid in an equivalent system of fluid without the effects of a diaphragm. To understand this reasoning, it is observed that for a discretized element of the boundary, the elastic curvature causes additional tension and moment forces, whose components contribute to the force balance in the normal direction to the diaphragm element. This new equilibrium, with the effects of the membrane taken into consideration determines both the equivalent fluid height $h$, for a given tank fill volume, and the shape of the diaphragm.

Thus, a problem arises when attempting to create a computational scheme controlling volume of the liquid in the tank - because in order to solve for equilibrium, the equivalent fluid height must be known in order to determine the diaphragm shape, which can then be used to find the fluid volume for that shape. In order to solve this issue, an iterative scheme is introduced for all diaphragm models.
The scheme is inspired by the iterative technique used in determining the Young-Laplace force balance for ideal liquid droplet profiles as outlined in Pozrikidis [18]. The technique used is as follows: an initial value for the equivalent fluid height is guessed, the membrane shape is determined which allows for a fluid volume to be found. This volume is compared with the desired volume and the equivalent fluid height is changed according to the error in volume. The process is then repeated until the error in volume is below the desired tolerance.

In the second dynamic case, the same equilibrium balance is solved; however, the forces imparted onto the membrane are now determined by solving the Navier-Stokes equations in order to account for the dynamic pressure contribution, and the additional effects of the fluid's momentum in a likely non-inertial frame of reference. Additionally, the membrane is no longer given an infinite amount of time to reach a static equilibrium position, and so its momentum and time dependent dynamics such as the membrane's vibration due to hyperelasticity must then be solved for in order to predict realistic behavior.
In order to solve the more comprehensive second dynamic case, commercial software can be used to solve the fluid-structure coupling problem. For example, ANSYS includes a coupling module that can be used to create a FSI model. In fact, a FSI model of a diaphragm has already been modeled by Sances et al. [13], although the model was never validated and the material was treated as isoelastic to simplify the structural calculations. The Sances et al. model however, shows that the capability exists to couple the fluid-structural interactions.

In conclusion, the fluid dynamic problem is fairly straightforward and can be solved numerically using conventional techniques and commercial software platforms such as ANSYS Fluent. However, the structural calculations for the membrane are more difficult because of the high amount of nonlinearity introduced by the coupling and hyperelastic material. As such, the stability of the structural solution becomes an issue. In the next section, this structural component is discussed.

2.4 STRUCTURAL INTERACTIONS AND FOLDING

The problem of solving for large deformations in thin elastomeric materials is a particularly difficult problem,
as direct non-axisymmetric analytical methods are apparently nonexistent, and the requisite numerical methods are computationally expensive and often highly unstable. In point, a large deflection simulation of an elastic slosh diaphragm inversion has never been achieved prior to this work. For a diaphragm model, the structural portion is the most difficult aspect to resolve accurately, and is therefore the focus of this thesis.

While a slosh diaphragm is in operation, it may experience a large range of accelerations while at multiple fill levels. These conditions modify the equilibrium balance between the gas and liquid volumes, shifting the membrane towards the liquid end as the tank drains. As the membrane moves, its curvature is altered, and collapses in the structure’s original resting-state-shape occur, which is called folding. Folds form when low bending resistance and high in-plane stiffness interact when the material becomes stressed. When the resulting in-plane deformation is much smaller than the out-of-plane deformation (curvature), then the material can be assumed to be inextensible, meaning that the in-plane deformation is small enough to be neglected. Inextensibility is often a
good assumption for most thin slosh diaphragms where capturing the folding behavior of the material becomes the primary concern for a predictive diaphragm model.

2.4.1 FOLD TOPOLOGY

There are many types of fold structures, each distinctive fold pattern including a network of connected sharp or blunt directional folds and cone-line singularities. Artists were the first to characterize folds according to their structures, making it easier to categorize which fold geometries are commonly present in various situations and materials. This is useful for diaphragm fold classifications, as it allows for more accurate descriptions of common diaphragm topology as a function of usage conditions.

The seven basic types of folds that have been characterized, and are shown in Fig. 2.3, are [19]:

1) **Half-lock fold**: Visually slack material changing direction sharply to form a single fold.

2) **Zigzag fold**: Repeating interlocked folds which alternate direction.

3) **Pipe fold**: Tubular-shaped folds whose cross section form an s-like snaking pattern.
4) **Spiral fold**: Multiple fold ridges form a spiral around a central axis.

5) **Diaper fold**: A triangularly shaped fold which flows from two sharp points and curves gently at the third.

6) **Drop fold**: Irregular interacting arc-like folds, whose overall shape moves in one direction as if dropping.

7) **Inert fold**: Topology caused by a cloth-like material resting on a surface to form non-uniform wave-like and static folds.
Fig. 2.3 Fold types with representative symbols: A) pipe fold; B) diaper fold; C) half-lock fold; D) zigzag fold; E) drop fold; F) spiral fold; G) inert fold [19]

While only seven types of basic folds are classified, they can be combined to form highly complex structures. Examples of these folds and some of their many possible combinations are shown in Fig. 2.4.
Fig. 2.4 Sketched examples of the seven fold types [19]

Due to varying tank fill levels the amount of non-tensioned material available to create folds increases when the liquid fill level nears 50% volume. When a high potential for deformation is reached, half-lock, inert, and zigzag folds are most common as shown in the diaphragm folding example pictures in Fig. 2.5.
Fig. 2.5 Diaphragm in a 40 inch diameter tank, (a) 50% fill fraction, (b) 75% fill fraction, (c) and (d) show examples of diaphragm folding, with fold types labeled. Images courtesy of NASA.

When large irregular folding occurs as shown in Fig. 2.5, there is potential for damage to occur to the diaphragm through wear, rubbing, and overstress. Fig. 2.6 shows the folding configuration that result in the maximum amount of strain to be applied to a diaphragm. This occurs when two complete folds are made to an elastomeric sheet, deforming the surface at the corner of the fold to 100% in each axial direction. The prediction and prevention of
irregular folding, and high strain folding, is a large concern to an engineer designing these systems.

Fig. 2.6 Double fold in a thin elastomeric member [11]

In a slosh diaphragm the maximum strain possible through bending alone, as shown in Fig. 2.6, is not sufficient to cause a rupture. Over time, however, wear due to the rubbing caused by these types of folds in the diaphragm are a concern. To predict diaphragm topology in order to account for a diaphragm's predisposition towards wear, surface strain values are required. Material models that can account for behavior due to the stress-strain relation of the elastomeric diaphragm are therefore important.
2.4.2 MATERIAL MODELS

The tank materials are elastomer based and undergo large deformations that equate to large strains (around or greater than 100%) at small loading conditions. Hooke's Law, which predicts linear elasticity, is accurate only for strains less that 5-10% and so an elastomeric diaphragm's behavior undergoing large deflections is not predicted by linear elastic theory [20]. The diaphragm material's elasticity is consequently categorized as nonlinear and must be modeled as a hyperelastic material, also known as a Green elastic material [21].

Hyperelastic materials are a subset of Cauchy elastic materials, which means that the material's stress is a function of the current deformation and an arbitrary reference position, but unlike a Cauchy elastic material, a hyperelastic material conserves the work done by stress. Hyperelastic materials are fully or nearly incompressible, and their deformations are fully recoverable. Hyperelastic materials are materials whose force-displacement behavior follows a constitutive relation (a relation between two physical quantities) that is always derived from the strain energy density function. The strain energy density function
is defined by the strain energy potential $W$ described as a function of the three strain invariants composed from stretch ratios, and sometimes the volume ratio $J$. The volume ratio $J$ is the ratio of deformed volume $V$ over the undeformed volume $V_0$ as defined in Eqn. 2.7. The stretch ratio, $\lambda$, is defined by an element's ratio of current length $L$, to undeformed length $L_o$, which is also a function of engineering strain $\varepsilon_E$, as shown in Eqn. 2.5 [22]-[26].

Eqn. 2.5

$$\lambda = \frac{L}{L_o} = 1 + \varepsilon_E$$

There are three principle stretch ratios: $\lambda_1$ and $\lambda_2$ that describe in-plane deformation where $\lambda_1$ and $\lambda_2$ are perpendicular to each other and parallel to the plane element, and $\lambda_3$, which describes the deformation in the normal-to-plane (thickness) direction of an element. The three types of testing data that can be used to derive the principle stretch ratios are as follows: uniaxial extension, biaxial extension, shear extension, and compression or volumetric extension. Uniaxial refers to data gathered about deformation in one principle axis, biaxial refers to deformation in the principle axis and
perpendicular to the principle axis in-plane, shear refers to diagonal in-plane deformation, and volumetric refers to volume deformation or in other words deformation in all 3 principle directions.

The strain invariants are the same invariants from the left Cauchy-Green deformation tensor. The three strain invariants are $I_1$, $I_2$, and $I_3$, in Eqn. 2.6 they are described in terms 1-3.

Eqn. 2.6

\begin{align*}
(1) \quad I_1 &= \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \\
(2) \quad I_2 &= \lambda_1^2 \lambda_2^2 + \lambda_2^2 \lambda_3^2 + \lambda_3^2 \lambda_1^2 \\
(3) \quad I_3 &= \lambda_1^2 \lambda_2^2 \lambda_3^2
\end{align*}

Eqn. 2.7

\[ J = \lambda_1 \lambda_2 \lambda_3 = \frac{v}{v_0} \]

Hyperelastic materials are modeled using several methods; these methods are classified as models using empirical methods (phenomenological), using statistical mechanics (mechanistic), or using a hybrid of the two. Some of the most well-known hyperelastic models by classification are as follows [22]:
Empirically-based methods:

1) **Fung**: A pseudo-hyperelastic model used to describe the finite behavior of biological soft tissues while they undergo deformation [23].

2) **Mooney–Rivlin**: A hyperelastic model where the linear combination of the first two invariants of the left Cauchy-Green deformation tensor are equated to the strain energy density function. The Mooney-Rivlin model is the same as the first order polynomial form [24].

3) **Ogden**: A hyperelastic model where the strain energy density function is a function of principal stretches and material constants [25].

4) **Polynomial**: A hyperelastic model where the polynomial combination of the first two invariants of the left Cauchy-Green deformation tensor are equated to the strain energy density function. When the second invariant is omitted, the model is known as the reduced polynomial model [26].

5) **Saint Venant-Kirchhoff**: This is the simplest of the hyperelastic models, and is only valid for combinations of large deflections and small strains.
Piola–Kirchhoff stress tensor is used as a stress measure, and the higher order terms in the second Piola–Kirchhoff stress tensor are neglected [27]. Since elastomers typically have large strains under small loads and are nearly incompressible, the Saint Venant-Kirchhoff model would not be appropriate for diaphragm analyses.

6) Yeoh: A hyperelastic model, similar to the polynomial model, where the polynomial representation of only the first invariant of the left Cauchy-Green deformation tensor is equated to the strain energy density function [28].

7) Marlow: A hyperelastic model for an incompressible material whose strain-energy can be completely determined by the first invariant, which is derived from data of a single mode of deformation such as uniaxial tension stretch. For this model, the integral of the uniaxial stress-strain curve from zero to the directional strain as a function of the first invariant is equated to the strain energy density function [29].
Methods based on statistical mechanics:

1) **Arruda–Boyce model**: A constitutive model for rubber elastic materials. The strain energy is equated to the sum of the strain energies of the individual chains oriented randomly in space [30].

2) **Neo-Hookean**: This hyperelastic model's form is the same as that of the polynomial model, but omits the second and third invariants of the left Cauchy Green tensor, only using the first. Additionally, the strain energy is equated to a function of the number of network chains per unit volume, the Boltzmann’s constant, and the material's absolute temperature. It is a function based on the statistical theory of rubber elasticity. This model is often used when material testing data is insufficient [30].

Hybrid models based on empirical and statistical mechanics:

1) **Gent**: A hyperelastic model where the strain energy density is equated to a logarithmic function of the first strain invariant, and two material constants [31].

2) **Van der Waals**: Also known as the known as the Kilian model where the strain energy density is equated to a
function of the two invariants of the left deviatoric Cauchy Green tensor, a linear mixture parameter, an interaction parameter, and the material's locking stretch parameter [30].

The application of these hyperelastic models all require careful parameter and test data gathering. Most cases require that at least the uniaxial (and occasionally biaxial, shear, and volumetric) test data be curve fitted correctly to derive coefficients for the numerical implementation of the model. The models with fewer coefficients are less computationally expensive, but may see a trade off in accuracy depending on their implementation [30]. A good strategy for choosing a material model is to first narrow the list by compatibility with your simulation, then again narrow the list by quality of the uniaxial\biaxial\shear\volumetric data's curve fit to develop accurate parameter models, and finally weigh these models by how computationally expensive each is.

From these definitions and my own computational experimentation with these models as discussed in the ANSYS modeling section, the Yeoh hyperelastic model was chosen to
be used in this thesis for its stability in pressure-driven conditions.

2.4.3 STRUCTURAL SOLUTION IMPLEMENTATION

Once a material model is chosen a numerical implementation must be used to solve for the forces and displacements resulting from the material's interactions. The most common and oldest numerical method used to approximate large deformations in hyperelastic materials is the structural finite element method (FEM) or structural finite element analysis (FEA). In FEA, a material is modeled as a system of discretized finite elements connected at node points. Each element contains the material property data. These material properties and node-element properties are composed into matrices that are used in the matrix stiffness method or flexibility method for analysis of the element's resulting forces and displacements. The matrix stiffness method, however, being a better application to non-linear systems, is the method referred to and used in this thesis during ANSYS modeling.

The FEA method has been around for the longest and is the industry standard for structural analysis. It is able to be coupled with a fluid solver in an FSI simulation as
seen in Sances et al. [13]. However, the method does have a few drawbacks. Problems can arise with badly scaled solution matrices, and this method is particularly susceptible to numerical destabilization given large time steps in large deflection systems which means FEA methods are often very slow and computationally expensive. It is also less common and more difficult to parallelize the solutions to a high degree using graphics processing units (GPUs) and so, in linear iterative cases the method remains computationally expensive for large grids that are needed to model complex folds [32].

Moving past traditional FEA methods, the development of new methods for simulating thin elastic materials has been undertaken largely by the computer graphics industry. These new methods are called soft body dynamic models or soft body deformation models. Soft body deformation models are made for realistically and quickly simulating non-rigid bodies, such as muscle, cloth, hair, vegetation, rope, and most organic materials. These models have been of increasing interest due to the rapid expansion of the computer graphics industry. These new methods are often initially based on physics-based simulations, and then
large amounts of simplifications and approximations are applied to the solutions in order to reduce the solution’s computational time for graphics applications such as video games and movies. However, physically based versions of these models can be created that are predictive.

Because solutions with large approximations and simplifications tend to be inaccurate scientifically, they also tend to be unconvincing in their intended computer graphics application. And so, the result of the combination of economic force and a graphical-realism-feedback-loop is for the computer graphics industry to trend towards the production of highly specialized, highly accurate, and computationally inexpensive simulations of soft body materials such as inextensible elastic materials (cloth). The simulation of elastomeric membranes can, therefore, benefit from this recent work, as it falls into the category addressed by these very specialized solution methods.

In order to apply the methods developed by the graphics industry to the problem of an elastomeric slosh diaphragm, it is necessary to ignore the solutions that provide only visually plausible results instead of
scientific simulation results. If the model is not scientifically accurate, it is often a matter of simply reverting the method back to a more scientific basis (e.g. removing the simplifications in the solution algorithm); however, this is not always possible.

A few of these new physically-based simulation methods - originally developed by the graphics industry - listed in the order of their development are: discrete explicit integration for solving the equations of motion methods, discrete implicit unconditionally stable methods, continuous methods, linear finite element methods with a co-rotational formulation, constrained Lagrangian mechanics in combination with a fast projection methods, continuum mechanics models, strain limiting methods, continuum-based strain limiting methods, position-based simulation methods, region based shape matching methods, and multi-resolution shape matching methods. Bendera et al. contains an excellent and up to date overview of the history of the physically-based simulation of thin elastic models [33].

In order to provide a better overview of the kind of processes these solution methods entail, the discrete explicit method that is implemented in this thesis will be
reviewed here. The method is a physically-based spring-mass n-particle body system, which is a subset of soft body dynamic models.

When simulating a slosh diaphragm or other thin elastic member, the assumption that the in-plane deformation is very small compared to the out-of-plane deformation leads to the conclusion that in-plane inextensibility is an ideal condition to impose onto a numerical solution. Typically, in-plane deformation in FEA is determined by solving the force equilibrium in-plane, leading to unstable solutions when high stresses and low strains are encountered such as in an inextensible material.

When making the assumption of inextensibility, these in-plane stress calculations become unnecessary. Instead, in order to simulate in-plane stiffness, a numerical constraint to fix in-plane deformation can be applied leaving only the out-of-plane force balance to be contended with.

In a spring-mass n-particle body system, the material is discretized into a series of panels, each connected at a node point. Each node is given mass and position.
information allowing the problem to be treated as an n-particle body simulation. These particle/nodes are connected by a series of perfectly damped springs. The springs act as the required inextensibility constraints by employing a mass weighted relaxation method at every time step until the inextensibility condition is met. The method is called a spring relaxation constraint. For strain in both axial directions, deformation is controlled by "axial structural springs", while in the shear direction "shear springs" are used. The typical computer graphics simulations may utilize additional "bending springs" in order to quickly simulate out of plane strain control. In their application, the springs are critically damped, in order to rearrange the points into a relaxed position without causing oscillations. A diagram of a spring-mass n-particle body system with all three strain modes of springs is shown in Fig. 2.7.
Fig. 2.7 Schematic of a nine node section for a spring-mass n-particle body system using all three spring modes

These constraint springs can be evaluated iteratively, or as a system of linear equations (SLE). The benefit of using a SLE method is that it solves for a perfectly inextensible solution, with zero in-plane strain. The benefit of using the iterative spring method of inducing
in-plane inextensibility constraints is that it is unconditionally stable, unlike the SLE formulation that may be unable to converge on an in-plane strain limited configuration for large deformations. The largest drawback for this increase in stability is that large numbers of iterations are needed for higher amounts of in-plane inextensibility. However, for very small non-zero amounts of allowable in-plane strain, such at 0.1% or 0.01%, the iterative method is faster than the SLE method [34].

Once the inextensibility constraint is imposed on a node grid, it is then a simple matter of solving the dynamic N-body system using our definition of stiffness and Newtonian mechanics. The construction of such a model is explained analytically in Chapter 4.

Other promising soft body dynamic methods for simulating thin elastic materials are continuum-based strain limiting methods as discussed in Thomaszewski et al. [35], and finite element strain limiting methods as discussed in Wang et al. [36]. The continuum-based strain limiting methods approach the problem of introducing inextensibility constraints through limiting the deformation of elements based on the linear co-rotated
strain tensor instead of individual strain components or springs. The finite element strain limiting methods acts on the strain tensors in a coordinate-invariant fashion that enable elastic isotropic materials to be simulated quickly by computing the singular value decomposition of the deformation gradient for each element. The finite element strain limiting methods can also utilize multiple resolution grids to enforce a hierarchy of deformation behavior in large resolution grids [36].

The parallelization of these types of models is also highly desired since real time simulation of dynamic elastic materials with large node resolutions can and are frequently achieved using these methods in conjunction with GPU parallel processing [33].

To conclude the discussion of numerical structural analysis, modeling methods, and solution implementation, it is found that traditional methods for simulating the structural component of the elastomeric diaphragm model are often unstable and computationally expensive. However, many new methods have recently been developed to allow for thin elastic materials to be simulated with relatively small computational expense. A traditional FEM model and a new
spring-mass model are created in this thesis. Moreover, more advanced methods such as the finite element strain limiting method - utilizing GPU architecture - could potentially provide real time simulation of slosh diaphragms in the future.
CHAPTER 3: ANALYTICAL MODEL OF SLOSH DIAPHRAGM

3.1 INTRODUCTION

In a spherical fuel tank with an elastomeric slosh diaphragm acting as a barrier between propellant and gas volumes, such as the one shown in Fig. 1.2, predictive knowledge of the diaphragm deflection under severe loading conditions is significant to life cycle evaluation. Specifically, the case of one such propellant tank being filled to its maximum extent, resulting in a crescent shaped gas volume opposite the propellant from the diaphragm. Under these conditions, ground transportation can occur which may involve the reorientation of body forces on the tank to produce a normal loading on the tank diaphragm in the downward direction as the propellant's weight compresses the gas volume. If the diaphragm is of a low stiffness and deflects notably, folding, rubbing, and increases in wear can increase as a result.

Tanks with constant thickness diaphragms that have larger diameters tend to have more folding, ridges, and ripples in their slosh diaphragm as compared with smaller tanks. Larger diameter tanks exhibit more motion of the diaphragm during fluid sloshing than smaller tanks. This
can be understood as the combination of several effects, based on the scaling of the tank diameters as compared with a constant diaphragm thickness.

As the diameter of the tank increases, there is more diaphragm material per diameter in a larger tank as compared with a smaller tank. Another effect of tank scaling is that the fluid mass inside the tank scales with the radius cubed for a spherical tank, and the diaphragm area scales with the radius squared.

For this chapter, the case of a static diaphragm deflecting under a propellant load is assessed in order to determine how much deflection can occur in high loading conditions when tank scaling is taken into effect, in order to determine the relationship between tank scaling and diaphragm stiffness. This case can be approximated using analytical solutions for thin circular plate bending in order to actively predict if large deflections in the diaphragm may occur from its initial static position. If plate theory predicts large deflections in the diaphragm greater than an order of three times the plate thickness, linear material behavior assumptions can no longer be made,
and numerical solutions must then be considered in order to accurately evaluate diaphragm behavior [37]-[38].

3.1.1 ANALYRICAL PLATE THEORY

Analytical solutions for circular membrane deflections are traditionally obtainable using Kirchhoff-Love plate theory. The Kirchhoff-Love plate deflection solutions evaluated in this report include small plate deflection theory and large plate deflection theory. Both analytical solutions remain accurate for plate deflections on the same order as the thickness of the plate. Small plate deflection theory is known to be accurate for deflections on the order of 1/5 the plate thickness, while large plate deflection theory is considered applicable for deflection up to three times the plate thickness [20], [37]-[38].

For larger deflections, highly nonlinear behavior is observed experimentally and numerical techniques such as finite element analysis are found to exceed the predictive ability of established analytical solutions. Therefore, the plate deflection theory presented in this report remains valid only for cases in which the equilibrium of forces is shifted between sides of a diaphragm originally at its non-
deformed position, resulting in deflections on the order of three times the surface thickness [37]-[38].

In plate deflection theory, $d$, is the flexural rigidity, where, $E$, is Young's modulus, $t$, is plate thickness, and, $v$, is the Poisson's ratio [38].

**Eqn. 3.1**

$$d = \frac{Et^3}{12(1-v^2)}$$

Flexural rigidity, $d$, is used throughout thin plate theory as a material property independent of loading conditions.

### 3.1.2 SMALL DEFLECTION PLATE THEORY

Small plate deflection theory for a circular plate under a uniform unidirectional loading, with clamped edge boundary conditions is first examined where, $w$, is deflection as a function of radial position, $a$, is the plate radius, $r$, is the radial position, and, $P$, is the net pressure applied to the plate in one direction [38], [20].

**Eqn. 3.2**

$$w(r) = \frac{Pa^4}{64d} \left[ 1 - \frac{(r/a)^2}{2} \right]^2$$
At the center of the plate, where \( r = 0 \), maximum deflection, \( f \), is measured. For small deflection plate theory using Eqn. 3.2, a correlation for maximum deflection of a material as a function of uniform force, \( F \), diameter, \( D \), and plate thickness, \( t \), is given by Eqn. 3.3.

**Eqn. 3.3**

\[
f \propto F \frac{D^2}{t^3}
\]

The relationship for maximum deflection shows that for a given loading, \( F \), and constant material thickness, \( t \), the deflection in a plate will increase as an exponentiation to the power two of a plate diameter increase. This relationship is plotted in Fig. 3.1.
Fig. 3.1 Displacement profile trends for circular diaphragms of varying diameters with clamped edges, ethylene-propylene terpolymer material, constant thickness $= 0.07$ in.

Fig. 3.1 shows that as diaphragm radius is increased for a diaphragm of constant thickness, the displacement increases at a nonlinear rate described by Eqn. 3.3. It can therefore be concluded that geometric diaphragm stiffness for a constant force, scales with diaphragm thickness, $t$, cubed and inversely with the diameter, $D$, squared. I.E. geometric diaphragm stiffness is proportional to the inverse of $f$. 
3.1.3 LARGE DEFLECTION BENDING PLATE THEORY

For large deflections as presented by Eaton et al. [37], the analytical plate deflection theory is described analytically by Eqn. 3.4.

Eqn. 3.4

\[ w(r) = f \left[ 1 - \left( \frac{r}{a} \right)^2 \right]^2 \]

The term \( f \) is the maximum deflection of the plate, whose coefficients are detailed in Eqn. 3.5 through Eqn. 3.8, where \( \varepsilon_i \) is the residual strain.

Eqn. 3.5

\[ f = \left( -\frac{\beta}{2} + \gamma \right)^{\frac{1}{3}} + \left( -\frac{\beta}{2} - \gamma \right)^{\frac{1}{3}} \]

Eqn. 3.6

\[ \gamma = \sqrt{\frac{\alpha^3 + \beta^2}{27 + \frac{\beta^2}{4}}} \]

Eqn. 3.7

\[ \alpha = 14 \frac{(4t^2 + 3a^2\varepsilon_i(1 + v))}{((1 + v)(23 - 9v))} \]

Eqn. 3.8

\[ \beta = \frac{-7Pa^4t^2}{8d(1 + v)(23 - 9v)} \]
At the center of the plate, where \( r = 0 \), maximum deflection \( f \) is measured. A simplification of the proportionality between \( f \), diameter \( D \), force \( F \), and plate thickness \( t \) is highly complex and so the relationship is more simply described through original coefficients shown in Eqn. 3.1 and Eqn. 3.5 through Eqn. 3.8.

### 3.1.4 ANALAGOUS CASES FOR SPHERICAL FUEL TANKS WITH A FUEL TANKS WITH AN ELASTOMERIC SLOSH DIAPHRAGM

Large deflection plate theory is used to predict deflection profiles for four analogous cases to propellant tanks with circular diaphragms. Water is used as the fill fluid, ethylene-propylene terpolymer as the diaphragm material, clamped edge boundary conditions are used, and a constant diaphragm thickness of 0.07 in is used. Cases 1 and 2 have tank diameters of 16.5 in, and cases 3 and 4 have tank diameters of 40 inches. In cases 1 and 3, a 50% tank fill fraction by volume is used, while a 25% fill fraction by volume is used for cases 2 and 4. The displacement profiles for the four cases are shown in Fig. 3.2 and Fig. 3.3.

For these cases, the fluid mass inside the tank scales with the radius cubed for a spherical tank, and the
diaphragm area scales with the radius squared. For two tanks with the same fill fraction, a tank that is twice as large in diameter has eight times the propellant mass and twice the force per unit diaphragm area.

In the comparison between the cases with differing tank diameters, more diaphragm material is used. By considering the diaphragm to be a bisecting plane in the sphere with surface area $S = (\pi D^2) / 4$, it can be shown that as the diameter increases, the surface area of the diaphragm increases with $D^2$, and the area of the diaphragm per diameter, $S/D$, increases with $D$. This results in more diaphragms available to form folds, ridges and ripples.
Fig. 3.2 Displacement profile trends for flat circular diaphragms in spherical tanks of 16.5" and 40" diameters with clamped edges, ethylene-propylene terpolymer material, constant thickness = 0.07 in., water fill fraction is measured by volume.

Fig. 3.2 shows a to-scale trend of displacement profile that indicates that, for smaller tank sizes, the decrease in loading fluid volume and diaphragm diameter result in smaller maximum deflections when compared to larger tanks with a proportional but larger mass of liquid and a larger diaphragm diameter. The large deflection plate theory used to produce Fig. 3.2 diverges in accuracy for
deflections greater than three times the plate thickness. In Fig. 3.2, the predicted deflection amount is far greater than the upper analytical predictive limit; therefore, it can be concluded from this study that for diaphragms under the conditions presented inside a spherical propellant tank of the type of interest to this thesis, the analytical solution is an inaccurate method of evaluation when compared to numerical alternatives.
Fig. 3.3 Nondimensional displacement profile trends for flat circular diaphragms in spherical tanks of 16.5" and 40" diameters with clamped edges, ethylene-propylene terpolymer material, constant thickness = 0.07 in., water fill fraction is measured by volume.

Fig. 3.3 is a non-dimensionalized plot of displacement profiles showing the same cases as in Fig. 3.2. Fig. 3.3 shows that for a constant diaphragm thickness, increasing fill fraction without increasing tank size results in an increase in deflection. While increasing tank diameter without increasing fill fraction results in a larger increase in deflection. This relationship between diaphragm
size, loading, and thickness is highly nonlinear, represented by, \( f \), which is not a linear function of force. Therefore, unlike small deflection plate theory where geometric diaphragm stiffness is proportional to the inverse of \( f \), large deflection theory's geometric diaphragm stiffness is complexly described by Eqn. 3.1 and Eqn. 3.5 through Eqn. 3.8.

A plot of \( f \) as a function of radius for a constant diaphragm thickness is given by Fig. 3.4, where plates with \( r \gg t \) show a nonlinear increase in \( f \).
Fig. 3.4 Maximum displacement (f) trends for flat circular diaphragms in spherical tanks with clamped diaphragm edges, ethylene-propylene terpolymer, constant thickness = 0.07in., water fill fraction is measured by volume.

Fig. 3.4 shows that large deflection plate theory predicts an increasing trend of displacement/radius for an increase in radius for thin plates. This trend excludes plate geometries where the radius is on the same order as thickness as seen in Fig. 3.4 where radius approaches the value for plate thickness. It can therefore be concluded that a decrease in geometric diaphragm stiffness is the result of an increase in diaphragm diameter, while the
diaphragms fully deflected cross-sectional aspect ratio \((f/r)\) varies complexly with diaphragm diameter for a constant diaphragm thickness.

The variations in diaphragm deflection profiles resulting from the analytical solutions presented in this report establish a clear trend of increasing deflection as a result of increasing diaphragm radius for a constant thickness. The relationship between a change in deflection and geometric parameters varies from the proportionality described by Eqn. 3.3 for small deflections on the order of 1/5 the plate thickness, to the complex relation, \(f\), described by Eqn. 3.1 and Eqn. 3.5 through Eqn. 3.8 for large deflections on the order of less than three times the plate thickness.

Therefore, stiffness, equal to force over deflection, can be related to the diaphragm deflection profiles as proportional to diaphragm thickness, \(t\), cubed and inversely with the diameter, \(D\), squared, for small deflections. For large deflections on the order of less than three times the plate thickness, the diaphragm stiffness is a complex relation described Eqn. 3.1 and Eqn. 3.5 through Eqn. 3.8. These results indicate that the relative geometric
stiffness of a diaphragm is effectively higher when the radius is increases for a constant thickness, i.e. larger tank's elastomeric slosh diaphragms will deflect more easily when diaphragm material thickness is held constant for all tank sizes.

As indicated by literature, the stiffness of a diaphragm with deflections greater than three times the material thickness will need to be evaluated experimentally or numerically using such techniques as finite element analysis. This study found that the magnitude of deflection predicted for a diaphragm within a spherical propellant tank of the type of interest to this report, were several orders of magnitude larger than the analytical solution limit. Despite this, the analytical solution can make accurate claims about overall trends of stiffness when looking at the initial small deformations that occur in the analytical regime. The initial bending in the diaphragm that causes nucleation points to develop and folds to later form is initiated by this initial small deflection regime. This mechanical theory helps explain why larger folding and deflections occur in larger fuel tanks. It is therefore concluded that the analytical solution presents compelling
descriptions for stiffness trends for small deflections, while finite element analysis and experimental techniques must be pursued in order to simulate large deflections and complex behavior such as folding.

3.1.5 MODELS IN LITERATURE

Shi et al. [39], presents an axisymmetric analytical model for the deformation of rubber membranes subjected to fluid loading. This model must be discretized and evaluated numerically in order to obtain solutions. Clariond, N. ([40] and [41]) presents an analytical mode for the hydroelastic behavior of an inextensible membrane, as well as a numerical implementation. However an English translation of these later works has not been made.
CHAPTER 4: NUMERICAL SIMULATION OF SLOSH DIAPHRAGM

4.1 ANSYS MODELING

In this section, the methods for creating a slosh diaphragm model in ANSYS will be outlined in detail. This guide is meant to direct the reader to understand which program options must be included, and why, in order to fully model slosh diaphragm behavior. Due to the high computational cost of an individual simulation and the multitude of variations to the program options which may arise when creating such a model, it is impractical to focus on the creation of a single simulation variation. This section instead focuses on the methodology of model creation.

4.1.1 TRANSIENT STRUCTURAL

This section covers the creation of an elastomeric slosh diaphragm model in ANSYS Workbench 14.5 for the Transient Structural solver. Specifically, the model is meant for the simulation of an inverting slosh diaphragm. The Transient Structural solver requires a material model and a geometry file. The material model is the most important aspect of simulating a thin elastomeric diaphragm. Elastic materials subject to large deformations
are often computationally unstable and prone to numerically divergent behavior if their elastic model is improperly composed. Large deformation models in particular, often require increasingly small time step sizes to resolve their behavior within a solution's stability limits.

The first input to the Transient Structural solver is the material model module. A block schematic for the entire ANSYS Transient Structural model is shown in Fig. 4.1. The material model for an elastomeric slosh diaphragm must include a material density parameter, which for AF-E-332 is equal to 1070kg/m$^3$. In order to resolve the hyperelastic behavior of the elastomer, the stress strain curves from test data must also be input. If the test data is not available, a 1st order hyperelastic model is recommended in combination with an initial modulus of elasticity that may produce visually plausible simulations.
If test data is available, the hyperelastic models included in ANSYS will require uniaxial test data, biaxial test data, and shear test data. If non-shell elements are used in the creation of the membrane geometry, it is also important to included volumetric test data. Once the test data is imported into ANSYS's material's module, the data can be curve fitted to a specific hyperelastic model. It is recommended that the Yeoh hyperelastic model be used, specifically the 1st, 2nd, or 3rd order versions. The order chosen is dependent on the quality of data curve fit. A better curve fit will lead to improved results, and the model order impacts curve fitting to a high degree.
Most of the hyperelastic models included in ANSYS by default, such as: Neo-Hookean, Polynomial, Mooney-Rivlin, and Ogden, are able to predict large displacements in translation or impact driven simulations fairly well for their relevant regimes of application. Note that these models are discussed in more detail in Section 2.4.2. However, in the case of an inverting slosh diaphragm, the deformation is entirely pressure-driven.

Through experimentation, it is found that the Yeoh hyperelastic model is prone to less instability, is able to be used on shell elements, and the model converges a higher percentage of the time in pressure driven simulations. Despite Yeoh's ability to model hyperelastic materials in pressure-driven conditions, the models are still very unstable. While the goal for this model is to simulate an inverted diaphragm, only partial deformations or fractional inversions are common before the model diverges unless excessively small time steps are used (<1e4s). While decreasing the time step can improve stability, it also increases computational time. This is due to an increase in the number of time steps necessary to reach the same end time as the time step increment lengths get smaller, and
therefore excessively small time steps are impractical in many cases. If a time step size lower than a specific tolerance is required for stability, this can also effect the achievable end time since computational effort is based on number of steps and not time step parameters. So for a given computational effort available of X steps, and a required time step of a certain size, $dt$, the time achievable is limited to: less than or equal to the product of X and $dt$.

In order to use the Yeoh hyperelastic model, material constant(s) must first be derived from the curve fits of the material's experimental data. The number of constants that describe the stress strain data's curve fit depend entirely on the type and order of the hyperelastic model. ANSYS does this derivation automatically once the experimental data is inputted into the hyperelastic model and a curve fit is evaluated. Once the hyperelastic model material constant(s) are evaluated, the material model is complete. For more information on material model creation and curve fitting in ANSYS, see the ANSYS Mechanical APDL Material Reference Manual [42].
The next component of the elastomeric diaphragm model which is addressed is the diaphragm's geometry. The geometry used for this model was a 20 inch diameter shell hemisphere created in ANSYS's built-in geometry creation tool. The shell's virtual thickness is defined as 0.007 inches, to match the thickness of the experimental AF-E-332 diaphragm. The edge of the diaphragm is modified so that once half inch of the edge is straight rather than curved like the rest of the shell surface. The edge is tangent to the diaphragm hemisphere. This modification is used to split the diaphragm into two surfaces, the smaller of which can be fixed in order to create a fixed support edge condition.

A shell surface is recommended for increased computational speed, and additionally thin shell features do not require as high of grid resolutions as tetrahedral or hexahedral grids. In order to preserve an aspect ratio close to one for a tetrahedral or hexahedral grid, the discretization length in-plane must be close to that of the thickness length. Because the diaphragm thickness is so small, acceptable aspect ratios are difficult to maintain as this results in a grid with millions of nodes, making
the simulation impractical. This aspect ratio problem needs only to be solved in-plane when dealing with shell geometries, resolving the issue to some degree.

After the geometry is created, the model module inside of the Transient Structural solver is addressed. Inside the model module, the geometry's surface body menu requires the material assignment to be chosen from a dropdown list. The hyperelastic material model previously discussed should be assigned in this field. In the surface body menu, nonlinear effects should also be turned on, and it should be confirmed that the surface thickness carried over correctly from the geometry module. It is also advised that a thickness offset type be defined as: middle.

If surface-to-object contact or surface-self-contact is expected, contact regions must be defined. To define a contact element, a manual contact region must be inserted into the connections menu, which can then be tailored to the specific contact method required. For surface self-collisions of a diaphragm, both the contact body and target body are the same diaphragm surface. However, two contact regions must be used in order to specify the two cases where the surface can self-intersect: on the top of the
shell face or the bottom of the shell face. Therefore, one contact will specify "top" for both contact and shell face regions, whereas the other will specify bottom for both contact and target regions.

The final step in defining contact regions is to specify the type of contact behavior. For this model, "rough" is used, which approximates an elastomeric material's contact behavior more than "frictionless". However, if the material's friction coefficient is available, "frictional" should be used in order to utilize the experimentally obtained value. An example of this contact treatment on a thin shell feature, driven by element displacement, using "rough" contact behavior is shown in Fig. 4.2.

![Fig. 4.2 Example of a rough surface contact model, A is the undeformed geometry, B is the displaced geometry](image)
Once contact regions are taken care of, the geometry must be meshed. To avoid grid dependencies, the resulting mesh must have a high mesh quality. Mesh quality can be determined by three mesh properties: aspect ratio, skewness, and smoothness. Ideally, each element should have an aspect ratio close to one, a low amount of skewness (shear of an element), and should not change element size abruptly (smoothness). For ANSYS, these statistics can be viewed in the mesh menu's statistics submenu under "Mesh Metric".

Note that higher grid resolutions will increase the simulation's ability to model complex folding topologies with sharp creases, so the maximum element characteristic length for a smooth mesh should be set as the radial length of creases for that material. This allows for a small number of nodes while also preventing aliasing from occurring at small folds. Adaptive meshing would be ideal to simulate small fold topologies; however this method is not included in ANSYS Transient Structural.

In order to create a quality mesh, the following settings are recommended: A mapped face meshing settings module using the quadrilateral method should be inserted,
which forces the grid to generate using quadrilateral elements. Quadrilateral elements conform to the uniform symmetrical folding which occurs when a diaphragm inverts, as opposed to triangular elements that struggle with modeling the smooth edges in folds. Note that the quadrilateral element's edge directions should be aligned with the edges of the fold creases in order for a benefit to be obtained. A refinement settings module is also inserted in order to control the resolution to a higher degree. A refinement level of one is adequate for this case.

The meshing method for sizing used is "On:Curvature" with a coarse relevance center, a course span angle center, and a smoothing setting of "high". The curvature normal angle should be set to a maximum of 2 degrees, where a smaller angle will results is a smoother grid. All additional mesh settings are kept as default. The resulting mesh can be seen in Fig. 4.3.
Fig. 4.3 Grid generated for ANSYS Simulations

Once the grid is generated, the transient model settings can be addressed. Unfortunately, initial static conditions are not supported for shell elements, so the transient will need to run through the initial pressurization which occurs instantaneously on the diaphragm. If shell elements are not used, a Static Structural module can be used to prime the solution with a displacement map of the diaphragm in equilibrium position.
given the initial pressure state. Moving on to the Analysis Settings options, the number of steps should be set to one, and auto time stepping turned on.

A significantly long end time should be defined so that dynamic effects play less of a role in solving for topological changes in the diaphragm. For this simulation the end time is set to 100 seconds. The time stepping should be defined by time and an initial step sized defined as 5 seconds. The minimum time step size should be 0.01 seconds or smaller. The maximum time step should be 5 seconds, and time integration should be left on.

For the solver controls, the solver type should be set to "Iterative", which helps solution stability in this case. Weak springs should be turned off, and large deflection must be turned on.

Moving on to the Nonlinear Controls menu, all convergence controls should be removed including line search and stabilization. By turning these controls off, less computational time is used to enforce these controls, and the solution becomes more stable. The Output Controls menu should also be addressed if hard drive space is a concern, 34 gigabytes of data will be created if all time
steps are stored. Lower the "Store Results At" number to something reasonable for your particular system if space is an issue. Additionally, from the Damping Controls menu, numerical damping should be set to 0.1, which improves stability.

The pressures and constraints on the model must at this stage be implemented. To understand the reasoning for the following implementation of pressures and constraints, read section 2.3, which includes an in depth discussion. A fixed support must be added to the edge surface that was created in the geometry discussion previously. The fixed support application functions as a fixed support edge condition.

To simulate pressure due to a static water volume, a hydrostatic force gradient must be inserted using a "Hydrostatic Pressure" condition on the fluid side of the diaphragm. Because ANSYS does not allow for a hydrostatic pressure gradient's zero pressure plane to be moved, a pressure gradient across the diaphragm must be induced by ramping up an additional pressure input. To accomplish this, a changing pressure condition is inserted on the pressurant side of the hemisphere. The acceleration field
in the hydrostatic can also be scaled by a reducing factor if the material model is particularly sensitive to deformation and the solution diverges often; however, the fact that this is necessary in some cases indicates a poor material model.

The changing pressure condition must cause the pressure difference equilibrium position, i.e. the equivalent fluid height, to move in the draining or filling direction. To save computational time, polynomial pressure "easing" functions can be used to change the pressure. An easing function allows for more computational time to be afforded to the largest buckling events and less time to pressure variations that do not trigger large deformation. Effectively speeding up the pressure change in low deformation conditions and slowing down pressure changes once high deformation conditions become prevalent. A pressure function application proportional to the inverse curve of the stress strain relation is found to work well at speeding up computational time.

The relation for producing a pressure easing function is given by Eqn. 4.1, where $P$ is pressure, and $P_{\text{max}}$ is the maximum pressure which the slope will converge to at the
final end time. The $P_{max}$ term is highly dependent on the elasticity of the material model. For this simulation, $P_{max}$ is equal to the hydrostatic pressure at the distance of the diaphragm radius, allowing for a full inversion to take place. The $T_{end}$ variable is the final time for a time step, $T_{end}$ is 100 seconds for this simulation, the $C$ constant influence the type of easing function, a small $C$ will produce a larger initial slope, while a larger $C$ will produce a smaller initial slope until an exponential increase occurs. A $C$ value of 0.003 is used for this model.

Eqn. 4.1

$$P = -P_{max}(\text{time} / T_{end})^{(T_{end}-\text{time})C}$$

Once the pressure conditions are applied, the last step is to set the solution output to display the desired conditions. Since we are concerned with topology, plots of displacement are outputted. The displacement plot results for this model are included in Appendix A. A pictorial timeline of the diaphragm inverting is included in Fig. 4.4. The model uses the material test data from the "Elastomer Sample (Yeoh)" for the material model curve fitting, this sample data is included by default in ANSYS 14.5's material library.
Fig. 4.4 Results of ANSYS Transient Structural model numbered in order of occurrence, maximum deflection is 0.4263 m, lower scale bar uses 0.1 m increments

The full size simulation results are included in Appendix A, the simulations are shown to be highly dependent on the material model, and since the exact stress strain curves for the diaphragm material were not available
for this thesis, the models are not predictive of the experimental case. Despite this, the methods for creating such a model in ANSYS - assuming a correct material model - are still valid.

The computers used to perform these simulations were Dell, 3GHz Intel Xeon 64-bit, 8 GB RAM workstations. The runtimes for these calculations, span lengths on the order of 1-3 weeks (or longer when smaller time steps are used to guarantee stability), and more often than not not diverge. Because of these difficulties, it is not recommended that ANSYS be pursued as a primary simulation tool for the modeling of slosh diaphragms. The computational time required and the instability of the model renders ANSYS impractical for any engineering applications discussed in the motivation section of this thesis.

4.1.2 FLUID STRUCTURE INTERACTION COUPLING

In this section, the creation of a FSI diaphragm model is reviewed. The model's results, however, are not included because the computational time needed to simulate the model (4+ weeks using the workstations mention in the last section) is unfeasible at the time of this thesis's construction. Additionally, the structural component of the
ANSYS model is still subject to frequent destabilization as discussed in the previous section. More practical methods of simulating a slosh diaphragm are instead made the focus of this thesis in section 4.2 in order to be more productive with development time. As such, this section is brief and only includes an overview of the required model structure and components.

In order to simulate dynamic conditions, the fluid volume influencing the diaphragm can no longer be approximated as a pressure gradient. Instead, the fluid must be modeled using a Navier-Stokes solver to take into account dynamic terms, and the structure solution coupled to the simulation in an FSI model. Fluent is used as the fluid solver for this model and is coupled with the ANSYS Transient Structural Model using the System Coupling module. A layout of the entire model is shown in Fig. 4.5.
The required components of the FSI model include: the hyperelastic material model discussed in the previous section, diaphragm geometry, a Transient Structural module, the tank fluid volume geometry, a Fluid Flow (Fluent) module, a System Coupling module, and a Results module.

The procedure the system coupling module uses for solving dynamic systems at a time step is to solve the Static Structural module for the deformation of the diaphragm. Then this deformation data is transferred to the Fluent module where the fluid flow is solved for the deformed fluid volume. The pressure output from Fluent is then transferred to the Static Structural module and the entire process is repeated for a number of set iterations. It is important to iterate the system coupling until the
change in deformation drops below some desired tolerance for each time step.

The geometry of the membrane should be created using the procedure in the previous section. The Tank geometry should be representative of the propellant half of the tank. The pressurant half of the tank is ignored when the assumption is made that the gas volume does not contribute any significant dynamic effects on the membrane, and instead equalized instantaneously.

In this case, a uniform pressure can be applied to the pressurant side of the membrane geometry in the Transient Structural module, which is equal to the atmospheric reference pressure in Fluent. However, if the atmospheric reference pressure in Fluent is set to zero, no additional pressure condition to the Transient Structural module should be applied since the pressure difference across the diaphragm should be made equal in both cases. The other necessary differences between the previous section's Transient Structural module and this sections module, should include: No hydrostatic pressure (This is taken care of by the Fluent coupling.), and a "Fluid Solid Interface" condition applied to the fluid side of the membrane. The
Fluid Solid Interface condition allows the system coupling module to receive the deformation of the interface.

An example of the tank and membrane geometry is shown in Fig. 4.6, where a cross section of the tank is shown, and the Fluid Solid Interface is labeled as A. The diaphragm in this picture is a simplified geometry represented as a flattened hemispherical surface.
Fig. 4.6 Cross section of FSI geometry including Fluent pressure result for a static tank in gravity, A) membrane and tank geometry (pressure is applied here), B) tank geometry, C) fluid outlet or inlet, D) pressure in tank.

The same grid generation conditions applied to the diaphragm must be applied to the tank. This assures that the meshes will align during the FSI coupling and no information loss will occur due to changes in grid resolution during the coupling. The surfaces of the tank
geometry must also be named inside the ANSYS meshing module so that they may be referenced inside of Fluent later on.

Once the tank geometry is created and meshed, it can be imported into a "Fluid Flow (Fluent)" module. The Fluent module setup should include the following settings. Double precision should be turned on in the starting menu, and the processing option should be set to serial instead of parallel so that system coupling will interact with Fluent correctly. This processing limitation will most likely be fixed in a future release of ANSYS.

In the solution setup's general menu, the solver is set to pressure-based, and time is set to transient. Gravity is set to match the requirements of the case; earth's gravity in the direction of -Y was used in Fig. 4.6. In the models menu, energy should be turned on. The standard k-epsilon (2 eqn) model is used as a viscous model. The viscous and energy equation options are included so that energy may dissipate allowing the solution to reach a static equilibrium given a long enough end time with small time steps.

In the materials menu, water-liquid and air are included as the fluids. The fluids must be changed if the
fuel tank simulated uses another gas such as nitrogen, or another liquid such as hydrazine. In the cell zone conditions menu, the tank volume must be specified as a fluid, and the operating conditions must include an operating pressure that is atmospheric pressure unless other ambient pressurant side conditions are required. The operating density must be specified as a zero unless a volume of fluid approach is used, in which case it is specified as the density for the less dense medium.

In the boundary conditions menu, the fluid outlet/inlet port at the bottom of the tank, which was named in the meshing mode earlier, must be set to the correct outlet or inlet type. The port type is dependent on the simulation desired, a pressure-outlet/inlet type will allow the pressurant side pressure (controlled in the Transient Structural module) to determine the draining or filling of the tank. A mass flow inlet condition will allow the user to specify an exact filling rate, or draining rate if the magnitude is set to a negative value. All other surface's types, included the diaphragm, are set to the "wall" condition.
If the simulation is meant to be used for a static tank, the wall and fluid port conditions are the only necessary boundary definitions. However, if the simulation requires the tank walls to move, the motion must be applied using a User Defined Function (UDF), for information on UDFs see the "ANSYS Fluent UDF Manual" [43].

In order to allow the diaphragm surface to deform, a dynamic mesh option must be used so that the internal volume of the tank can restructure its mesh as the diaphragm deforms. In the dynamic mesh menu, the remeshing must be turned on. Smoothing may also be used if the remeshing results in high mesh element size variations. In the settings menu for dynamic meshing the "Local Cell" remeshing option must be used since the mesh is fully 3D. The minimum and maximum length scales should be set to a range close to, but smaller than, the current mesh scale lengths. The maximum cell skewness should not exceed 0.9, and the remeshing interval is set to one so that deformation is handled properly at each time step.

In the dynamic mesh zones menu, a system coupling type interface must be applied to the surface of the tank that is shared with the diaphragm. In the creation menu, under
meshing options, the cell height should be set to the maximum length scale for the current mesh. See "11.3.1: Setting Dynamic Mesh Modeling Parameters" of the ANSYS Fluent internal help menu for more information on dynamic remeshing. Once the dynamic mesh zone is set up, the solution method can be addressed.

In the solution methods menu, the solution scheme is set to Coupled, and the default solution settings are preserved. In the solution controls menu, the Courant number is set to a value representative of the amount of fluid flow expected, in the case of a filling or draining tank this number would be low (around 1). The Courant number is defined in Eqn. 4.7, and a discussion of Courant number is included in Section 4.2.3. For the coupled solution method, a standard initialization can be used, computed from the tank walls.

Lastly, in the run calculations menu, the time step size should be set to match that of the Transient Structural module, and the number of time steps should allow the solution to reach the end time also specified in the Transient Structural module. A high enough iteration count for each time step should be set so that the solution
reports it has converged. If one time step ever reaches the iteration limit without converging, the limit should be raised or the time step size decreased.

In the system coupling module, the analysis type must be set to "Transient". The time duration and step sizes should match that of the Fluent and Transient Structural modules, or the solution will fail. The choice of maximum and minimum coupling iterations must be made so that change in deflection is driven to an acceptably small value; a value of 5 to 10 is recommended. In the coupling module, two "Data Transfer" connections should be initialized. The first should have Transient Structural as the source, and Fluent as the target. The second Data Transfer connection should be the same as the first, but have the source and target reversed.

After all of the modules are completed, the results can be reviewed from a results module. This model may be used to simulate tank draining and filling. Though, because the Transient Structural module alone can perform draining and filling simulations, the model is better utilized to simulate dynamic tank movements. With the help of a movement UDF, the model is capable of simulating dynamic
conditions such as periodic movement. Periodic and non-periodic movement simulation allows the model to be useful for the analysis of slosh's effect on diaphragm stresses and topology changes.

4.2 MATLAB MODELING

In this section, a slosh diaphragm soft body model is created and its theory is discussed. This model, while able to simulate a slosh diaphragm, has less features compared to the state of the art soft body simulators; however, the model's theory and construction can be considered to be a backbone necessary for many more advanced simulators. One of the model's main values lies in the fact that it employs techniques which can be used to understand the design of other more cutting edge soft body simulators or as a base for advancement of the governing simulation methods, when applied to slosh membranes. The full code for the model and relevant subroutines are included in Appendices B-J.

4.2.1 GRID GENERATION

The first part of the simulator that needs to be addressed is the creation of the discretized membrane, and its computational representation. Because the slosh diaphragm is hemispherical and the soft body simulation
technique used works using quadrilateral elements, a custom
grid creation subroutine was created to generate a
hemispherical shape with proper uniform discretization.
The measure of mesh quality can be determined by three mesh
properties: aspect ratio, skewness, and smoothness.
Ideally, each element should have an aspect ratio close to
one, a low amount of skewness (shear of an element), and
should not change element size abruptly (smoothness). The
grid creation algorithm developed includes all of these
features for the purpose of creating a hemispherical grid.

Because smoothness is desired and the mesh must be
quadrilateral at all locations, an Azimuthal equidistant
projection method as shown in Fig. 4.7 (left) could not be
used. Instead gnomonic projection as shown in Fig. 4.7
(right) of a cube onto a sphere, which is a type of
equiangular projection and is also known as a cubed sphere,
is used as a base for further work [44].
Fig. 4.7 Azimuthal equidistant projection (left), gnomonic projection of an inscribed cube (right)

While one of the six segmented projections from the gnomonic projection method includes the requisite properties for the slosh diaphragm model, the combination of the six do not, as they do not exist uniformly in a square matrix. A single patch from the gnomonic projection method, as shown in Fig. 4.8, exists in a space occupying a quarter pi of the sphere, when the projection is expanded to occupy a half pi of the sphere, a hemisphere is produced which occupies a square matrix in the computational domain.
In Fig. 4.8 it can be seen that the original quarter \( \pi \) segment has excellent grid quality, while the expanded hemisphere contains highly skewed element and a low amount of smoothness near the edges. Despite the expanded grid's low quality, it is maintained as a square matrix of points in the computational domain, and is therefore ideal as a base for further grid quality refinement.

The equations for creating a gnomonic projection of a cube onto a sphere are given by Eqn. 4.2 [44] where \( a \) is a constant and function of sphere radius \( R \). \( x \) and \( y \) are local coordinates and, \( X \) and \( Y \) are global coordinates, \( r \) is the local radius and, \( \alpha \) and \( \beta \) are grid line angles around the \( y \) and \( x \) axis.
Once the grid is initialized by expanding the $\alpha$ and $\beta$ ranges to $[-\frac{\pi}{2}, \frac{\pi}{2}]$ then the expanded portion can be unskewed to create a higher quality mesh. To unskew the mesh, a map of the $(X,Y,Z)$ 2-D matrix grids are adapted into a spiral pattern spatially as shown in Fig. 4.9. Computationally, this pattern is recorded as $(X,Y,Z)$ 1-D arrays.
Fig. 4.9 Array of grid points in spiral pattern for angle equalization

Each arm of the spiral (repeating segments in the range of zero to two pi radians) is then analyzed to find the number of nodes and their angular position in the X-Y plane. The points can then be rotated to an equal spaced position around the Z-axis. The result from this modification to the mesh is shown in Fig. 4.10.
The resulting grid method, shown in Fig. 4.10 (right), is the final grid method used in the slosh diaphragm simulation.

To finish this discussion of grid creation, it is also worth discussing that several recent advancements in soft body simulation methods have utilized a technique called multiple grid resolutions [36]. This technique involves the creation of several grids with multiple resolutions which are created cospatially and then made to share common connecting nodes. By doing this, there can be variation in the allowable strain in each of the grids, resulting in more realistic and less grid dependent bending solutions. For this thesis, a second grid was superimposed onto the first courser grid for improved bending visual realism. The second finer grid is created as a high resolution spline.
surface, while accuracy is maintained by imposing physical constraints on only the first grid, allowing only one-way coupling. An example of the type of complex bending and folding which this technique allows is shown in Fig. 4.11.

![Complex Topologies](image.jpg)

**Fig. 4.11 Two examples of complex topologies for simulated diaphragms with high allowable strain rates**

The pictures in Fig. 4.11 are sample outputs from the Matlab diaphragm model developed in this thesis. High strain rates were permitted in order to show off the model's ability to simulate folding.

### 4.2.2 SOLUTION INITIALIZATION

In order to simulate the slosh diaphragm once the grid is generated, several parameters must first be initialized. The area of each diaphragm element face \((A)\) must be found using trigonometry. Once the face area is found, a given
The diaphragm material density \((\rho)\) and thickness \((t)\) can be used to calculate the diaphragm node mass by using Eqn. 4.3.

\[ m = \rho At \]

The fixed or driven conditions for the grid must also be initialized. This is accomplished by setting the node mass for the fixed and driven nodes to infinity, preventing them from moving when the physics solver is used. An X, Y, or Z position as a function of time can also be applied to a node in order to drive the node. Any driven node must also have an infinite mass in order to maintain independence from the physics solver.

In order to later apply relaxation constraints to the grid, several undeformed element to element lengths must be recorded, these are the spring lengths. Fig. 2.7 shows a visual of these perfectly damped spring connections. The subroutine "GridLengthMap" serves to calculate these lengths, it returns the spring lengths (all in the computational domain) for X axial direction, Y axial direction, positive slope shear direction, negative shear direction, bending spring length in the X axial direction, and bending spring length in the Y axial direction. The
summation of the structural and shear lengths are also calculated, in order to later check how much strain exists in-plane to the grid elements.

4.2.3 **PHYSICS SOLVER**

At the heart of the simulator is n-body physics solver. The physics solver is the explicit Euler integration forward step method, chosen for its stability given a Courant number less than one, and for its variability once implemented. This method allows for the evaluation of an n-body system which tracks the position of each particle. Using Verlet integration, velocity and acceleration are derived from the position data. The equations to update the position at every time step for one particle are given in Eqn. 4.4 terms 1-4, where $t$ is time, $dt$ is change in time, $a$ is an acceleration vector, $F$ is a force vector, $m$ is particle mass, $v$ is velocity vector, and $(X,Y,Z)$ is the position array. It must also be noted, that when constraint or collision position modifications are added to the scheme, the actual position vs. the predicted position in Eqn. 4.4 differs. The movement information from the previous step must therefore be corrected to the actual movement instead of the predicted movement.
The forces on a diaphragm element at height \( Z \) in the case of a static inverting diaphragm are given by the hydrostatic pressure equation, Eqn. 4.5. Where \( \rho \) is fluid density, \( g \) is acceleration (gravity in the case of these simulations), and \( h \) is the equivalent fluid height. \( P \) is the resulting 2-D matrix of pressures at each discretized element.

Eqn. 4.4

(1) \( t = t + dt \)
(2) \( a = a + F/m \)
(3) \( v = v + a \ast dt \)
(4) \( (X,Y,Z) = (X,Y,Z) + v \ast dt \)

The force due to pressure is directional, and must be applied normal to the surface of the membrane. In order to apply the force correctly a normal map is created of the membrane at each time step, this 2-D matrix is called \( N \).

The map of each node's surface area, a 2-D matrix called \( A \), is used to determine force using Eqn. 4.6. The force due to gravity is also added as equal to mass multiplied by
gravity. A structural force due to the bending elasticity of the membrane $F_b$ is also added to create Eqn. 4.6, the final force equation which calculates $F$, the force applied at each node. Additional force terms can also be added to the right hand side of this equation in order to account for any additional forces on the membrane such as forces due to dynamic fluid motion which may augment the physics solution.

**Eqn. 4.6**

$$F = F_b + A P N + M g$$

The time step size is chosen to satisfy the Courant–Friedrichs–Lewy condition where the Courant number must be less than one for the explicit method used in order to insure a stable solution. The length scales for change in position are used to evaluate the Courant number which is given by Eqn. 4.7. The Courant number is evaluated at each iteration and in each of the three dimensions. In Eqn. 4.7, the magnitude of the velocity vector at a node is evaluated and multiplied by the time step size and divided by the characteristic element length at that node. To satisfy the Courant–Friedrichs–Lewy condition, each node by is checked by Eqn. 4.7 to make sure no points have exceeded a Courant
number limit of one, and if it has exceeded one, a lower time step size is required [45].

Eqn. 4.7

\[ C = v \times dt/dx \]

After some element displacement is applied by the physics solver, and the Courant number check confirms the stability of the solution, relaxation constraints, collision detection, and bending strain limiting can be applied.

4.2.4 MATERIAL BEHAVIOR SIMULATION

Modeling deformable continuous material requires additional handling to the rigid body physics simulation method discussed in the previous section. In order to model deformation behavior for a thin elastomeric material, the out-of-plane and in-plane stress-strain reactions are segregated in order to simplify the calculations. For the soft body model created in this thesis, there are three types of deformation control methods which are utilized: in-plane strain constraints, out-of plane strain constraints and bending forces, and collision constraints. The outlines for these methods are as follows.
In-plane strain constraints:

In order to speed up computational time for the simulation of soft bodies, particularly thin elastic material, simplifications can be made to the models. In the case of a thin slosh diaphragm, the in-plane strain is very low compared to the out-of-plane strain, allowing for the assumption of inextensibility to be valid. This means that any displacement in-plane is insignificant, allowing this in-plane behavior to be modeled by a simple relaxation constraint rather than wasting computational time on an in-plane FEA calculation.

In Section 2.4.3, a high level explanation of the methods for the application of this relaxation constraint is given. The simplest, and for the purposes of this thesis, the fastest method for applying in-plane strain limitations are the spring relaxation constraints. These computational springs are critically damped and applied to the computational domain's grid as shown in Fig. 2.7.

The method for applying the iterative spring relaxation constraints to the three 2-D matrixes: X, Y, and Z, for slosh diaphragm grid positions, is outlined as follows for the X direction axial springs. The scheme uses
the Gauss-Seidel method for iterative relaxation [46].

Where: i and j are the x direction and y direction indices in the computational domain for the slosh diaphragm grid, and m is the 2-D mass matrix corresponding to the grid nodes. Point1 and Point2 are the vectors for the two points connected by a spring. Vector_Distance is the vector distance between the two nodes, Total_Distance is the magnitude of the Vector_Distance, and strain is the deformation accumulated between the points. An outline is as follows, showing the process for a single iteration.

\[
\text{Given: } X(i,j), Y(i,j), Z(i,j), m(i,j), \text{ and } \text{LengthX}(i,j)
\]

\[
\text{loop1 for } i = 1 \text{ to end}
\]

\[
\text{loop2 for } j = 1 \text{ to end} - 1
\]

\[
\text{Point1} = [X(i,j), Y(i,j), Z(i,j)]
\]

\[
\text{Point2} = [X(i,j + 1), Y(i,j + 1), Z(i,j + 1)]
\]

\[
\text{Vector\_Distance} = \text{Point2} - \text{Point1}
\]

\[
\text{Total\_Distance} = \text{norm}(\text{Vector\_Distance})
\]

\[
\text{Strain} = \text{Total\_Distance} - \text{LengthX}(i,j)
\]

\[
\text{Point1} = \text{Point1} + \text{Strain} \times \left( \frac{\text{Vector\_Distance}}{\text{Total\_Distance}} \right) m(i,j)^{-1}
\]

\[
\text{Point2} = \text{Point2} - \text{Strain} \times \left( \frac{\text{Vector\_Distance}}{\text{Total\_Distance}} \right) m(i,j + 1)^{-1}
\]

\[
\text{endloop2}
\]

\[
\text{endloop1}
\]
When this method is iterated, the overall magnitude of the each element's strain is lowered and the method converges into the Gauss–Seidel solution for grid relaxation for a desired allowable strain percentage.

Out-of-plane strain constraints and bending forces:

Elastomeric material behavior is more accurately classified as hyperelastic instead of linearly elastic. A typical hyperelastic material stress strain curve in comparison to a linear elastic curve is shown in Fig. 4.12 [47].

![Stress Strain Trends](image)

**Fig. 4.12 Stress Strain trends for linear elastic and hyperelastic materials [47]**
When attempting to simulate this hyperelastic behavior in a soft body model, the specific material's stress vs. strain curve may be applied directly to the problem. In order to model the correct bending behavior directly, bending springs as shown in Fig. 2.7 may be used to apply selective grid relaxation as a function of measured stress from the physics solver. For this case, however, since the material stress-strain curves for AF-E-332 are not available, a simple two-part linear curve fit method is used. This method functions by modeling the hyperelastic curve as two linear segments.

For the first segment, the bending elasticity is modeled as a linear section of the stress strain curve according to plate deflection theory, which is covered in Section 3.1.1. This method requires only the material's Young's Modulus, $E$, and thickness, $t$, which are available for this thesis. The method involves calculating the reactionary elastic force exerted by a bent mesh member and then applying that force to the grid in the physics solver.

In order to calculate the reactionary elastic force on a bent grid member in one axial direction, defined by three grid points, spanning a length, $L$, the bending stiffness is
needed. The equivalent bending spring stiffness, $k$, for one three node bending member on the diaphragm's grid, derived from analytical plate bending theory, is given by Eqn. 4.8. Where for a flat simply supported plate in one direction the area moment of inertia, $I$, is defined in Eqn. 4.9. Using Hooke's law given in Eqn. 4.10, where $w$ is the deflection due to bending, the reactionary force due to bending, $F_b$, of the bending member is derived [48]. This force due to bending is then fed back into Eqn. 4.6.

**Eqn. 4.8**

$$k = \frac{48 E I}{L^3}$$

**Eqn. 4.9**

$$I = \frac{t^3 L}{12}$$

**Eqn. 4.10**

$$F_b = k w$$

The second segment of the simplified elastic bending model is modeled as a vertical linear section, which is created using the Gauss-Seidel grid relaxation method presented earlier in this section. This vertical linear segment serves as a strain limiter, simulating the sharp
upturn at the end of the hyperelastic stress vs. strain curve. The second strain limit is achieved through iterating the Gauss-Seidel relaxation method until the strain limit is reached, or instead for a smaller set number of iterations to simply damp out-of-plane strain by a constant factor.

**Collision Constraints:**

Various methods for handling self-collisions in thin deformable materials have been developed throughout the history of soft body model development and FEA development [49]-[52].

The method used in this thesis for determining whether a collision has occurred and then correcting the collision is outlined as follows. First, every possible mathematical combinatorial difference between two nodes is calculated by subtracting the two nodes position vectors and determining their mathematical norm. If one of these differences is less than a given collision distance tolerance, it is then determined that the points have intersected. The two intersecting points are then moved apart normal to one another using the same numerical technique as the Gauss-Seidel grid relaxation method.
It is important to keep in mind that this and other methods of determining whether or not a collision has occurred and then correcting that collision are often very computationally expensive. Thus, for simulations of a slosh diaphragm where self-intersection is unlikely to occur, it is best to omit the collision subroutine.

4.2.5 MATLAB RESULTS

In order to simulate a slow inversion of the slosh diaphragm whose conditions are matched to the equivalent laboratory conditions discussed in Chapter 5, the parameters listed in Table 4.1 were used as inputs to the Matlab model. The parameters were chosen to simulate the conditions of the experimental diaphragm inversion captured by 3D scans.
Using the parameters listed in Table 4.1, the simulation was run. The simulation took 6 hour and 50 minutes. The topology results from the simulation, for volume fill levels from 0% to 100% by 10% increments, are shown in Appendix K.

It was observed that decreasing the allowable strain can improve realism, as the solution approaches the inextensibility assumption. However, for low resolution grids a phenomenon called "grid locking" can occur in which the
bent elements cannot deform enough to invert, causing the grid to lock into place, artificially stopping the solution. Allowable strain levels less than 2% were observed to be particularly susceptible to grid locking. Allowable strain levels less than 2% also provided more realistic results when used in conjunction with higher grid resolutions to prevent grid locking.
CHAPTER 5: EXPERIMENTALLY OBTAINED 3D SCANS OF INVERTING SLOSH DIAPHRAGM

In order to determine the accuracy of the computationally simulated slosh diaphragm, 3D scans were taken of a 16.5 inch diameter hemispherical 0.07 inch thick AF-E-332 elastomeric membrane. The experimental data utilized was produced using Florida Tech's 3D scanning equipment. The scanning equipment is composed of 6 Xbox 260 Kinect infrared sensors. Each of the sensors utilizes a small electric motor which vibrates the sensor, reducing interference from other sensors. The sensors are shown in Fig. L.3.

The scanned images were processed using the program "Kscan3D" and later imported into Matlab. An example of a 3D scan of the diaphragm is shown in Fig. 5.1, the scan is from a fuel tank diaphragm at 70% fill volume. A sample photo of the tank at a 70% fill volume, which was scanned and used to create Fig. 5.1, is shown in Fig. L.4.
Fig. 5.1 3D scan of diaphragm at 70% fill volume

The 3D scans are accurate to within 1 mm, however they are unable to capture sharp crevices such as the ones that exist near the edge of the diaphragm. These hard to image valley topologies are most prominent at lower fill levels. Because of this reduction in scan quality at low volume fill levels, scans were only made for volume fill levels of 40% to 100% in 10% increments.

The full set of 3D scans topologies from 40% fill volume to 100% fill volume are shown in Chapter 6 alongside the simulation topologies.
The experimental platform that houses the fuel tank is shown in Fig. L.1 and Fig. L.2. The tank includes a gas port, a fluid port, a lexan case, and an aluminum retaining ring. The mounting platform is made of 80-20 aluminum, and doubles as a platform for measuring center of gravity. The tank is mounted on four springs, one spring holding up each corner. LVDT displacement sensors are placed at the corners of the tank to measure the displacement caused due to shifts in the center of gravity. The top of the frame can be tilted in order to tilt the fuel tank, allowing the calculation of the vertical center of gravity component.

A photo of the entire 3D scanning experiment setup, with the 6 Xbox Kinects mounted on cinderblocks pointing down at the tank, and the computer used to process the 3Dscans is shown in Fig. L.5.
CHAPTER 6: RESULTS OF COMPARISON

The results from the soft body deformation Matlab model were processed and compared for tank volume fill levels from 40\% to 100\% by 10\% increments. The data processing and comparison Matlab code is included in Appendix I.

In order to compare the topologies from the experimentally obtained 3D scans and the simulated soft body deformation Matlab model, an error definition was derived. First, the assumption is made that the maximum error 100\% is achieved when the simulated diaphragm surface and experimental diaphragm surface are inverted at opposite ends of the tank, creating an offset volume equal to the volume of the tank. Alternatively, if the two surfaces are completely co-spatial, the offset volume between them is zero. These assumptions allow for the definition of error to be given as the volume offset between both surfaces divided by the total volume of the tank, as shown in Eqn. 6.1.

\textbf{Eqn. 6.1}

\[
\text{Error} = \frac{\text{Offset Volume}}{\text{Total Tank Volume}}
\]
Before comparing the scan and simulated topologies, the limits imposed by the 3D scanning must be noted. Due to the limits of 3D scanning resolutions at sharp viewing angles, the diaphragm edge is particularly hard to resolve. In the real and simulated diaphragm the edge of the diaphragm is fixed tangent to the side of the tank as shown in Fig. 6.1.

Fig. 6.1 Dissection schematic of elastomer diaphragm's connection to inside of propellant tank [12]

The clamped edge condition represented in Fig. 6.1 is unable to be scanned and so the edge represented in the experimentally obtained surfaces are inaccurate - artifacts of the metal tank edge. Additionally, because the sharp
valley folds that can develop at the tank edge cannot be captured using the 3D scanning equipment, a large amount of noise is often present at the edge. Because of these reasons, only the center 80% of the surface is evaluated in the error calculation, though the entire cross sections of the surfaces are shown in the comparison plots for transparency.

In Fig. 6.2, the comparison between simulation and scanned topology is shown for a fill volume of 40%. The error measured between these two surfaces is 12.1%. It can be seen that for this low fill volume, noise is introduced at the edge of the tank where a valley fold should be present. The clamped edge condition shown by the simulation topology is also not captured in the 3D scanning process; as such the outer 20% ring of the scan is excluded from the error comparison.
Fig. 6.2 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 40% fill volume.

It can also be seen in Fig. 6.2 that a dimple forms downward in the simulation while the experimental surface remains in a dome shape. This feature can be seen throughout the different fill volumes and is attributed to the fact that the simulation modeled a perfect hemispherical geometry, while the real diaphragm's surface has varied thickness, and geometry resting shape.
abnormalities that influence the surface stiffness, and therefore the deformation behavior. The most prominent display of this difference is seen in Fig. 6.8, where the experimental diaphragm is shown to, in reality, contain a large dimple deformity which is likely very influential to the lack of inversion of the center of the diaphragm seen in Fig. 6.2, Fig. 6.3, Fig. 6.4, and Fig. 6.7.

**Fig. 6.3** Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 50% fill volume.
Fig. 6.4 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 60% fill volume

In Fig. 6.3 and Fig. 6.4, the comparisons between simulated and scanned topologies is shown for the fill volumes of 50% and 60%, respectively. The error measured between these two surfaces decreases as the fill level increases, from 7.6% to 6.4% error.
In Fig. 6.5 and Fig. 6.6, the comparisons between simulated and scanned topologies is shown for the fill volumes of 70% and 80%, respectively. The error measured between these two surfaces decreases as the fill level increases, from 5.7% to 4.1% error.
Fig. 6.6 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 80% fill volume
Fig. 6.7 Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 90% fill volume

In Fig. 6.7, the comparison between simulation and scan topology is shown for the fill volume of 90%. The variation measured between these two surfaces increases to a 10.7% error. It can be observed that the simulation predicts that the middle of the diaphragm will invert downward and the sides will bubble outward at this equilibrium position, while the experimental data shows
that diaphragm remains dome-like at this fill level. This difference is likely caused by differences in resting diaphragm local stiffness between the real diaphragm and the modeled diaphragm. These differences in local stiffness can be caused by the manufacturing and storing process, the installation process, and the operational life degradation.

**Fig. 6.8** Topology comparison between the Matlab simulation and the experimentally obtained scans of the slosh diaphragm at 100% fill volume
In Fig. 6.8, where the fill volume is at 100%, and the error is 3.1%, the geometry differences between the two diaphragms are revealing in the fact that the experimental topology can be clearly seen to not be uniform in shape. There is a dimple in the real diaphragm which can be seen at the top of the cross-sectional figure in Fig. 6.8, this dimple would prevent opposing bending at this local spot on the real diaphragm, which explains some of the differences in simulation vs. experimental data.

![Graph showing error percentage vs. fill volume percentage]

**Fig. 6.9 Simulated topology and experimental topology measurement error, as a function of fill volume percent, with an average of 7.1% error**

The error results of these simulations are summarized in Fig. 6.9. A trend of decreasing error at higher fill...
levels can be seen. The average error from this soft body deformation simulation technique results in an average error of 7.1% with a standard deviation of 3.3% when applied to a nearly static inverting diaphragm.
CHAPTER 7: CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

From the comparison between the experimentally obtained 3D scans of the 16.5 inch diameter diaphragm and the simulated diaphragm, the following can be concluded. 3D scans of actual diaphragm show that the diaphragm's geometry is in fact irregular in shape, while the simulation was performed with a perfectly hemispherical geometry. Visually examinations of the diaphragm also show that thickness varies along the surface of the diaphragm adding additional non-uniformities to the local stiffness of the diaphragm. These geometry irregularities and the fact that the scanning equipment was unable to capture sharp valley folds, both contributed to the measured error between the simulated diaphragm and the experimental diaphragm.

The analytical modeling revealed that it is indeed necessary to numerically evaluate the diaphragm as analytical methods are invalid for deformations on the large order with which the problem is concerned. However, for the initial small deformation period that the diaphragm goes through can be informed by analytical theory, which is
valid for this initial range. It then follows that initial buckling nucleation points will arise and form bends on lengths which are a function of material stiffness, pressure on the surface, and material thickness. The general trend is described as: increasing surface area for a given pressure and material thickness will cause a non-linear increase in bending.

For the ANSYS model creation methodology, it was shown that creation of a slosh diaphragm model is possible but very impractical. A simulation that can be run in a reasonable amount of time is not predictive, and often unstable.

The soft body deformation model was shown to be the most successful model. The model is able to predict the diaphragm shape with an average error of 7.1% with a standard deviation of 3.3% when applied to a nearly static inverting diaphragm. The model is more accurate at higher fill levels, and is able to model complex folding events, such as those which are present when dynamic forces become prevalent. Additionally, these simulations take 1-3% the computational time of traditional FEA methods and with additional optimization may be allowed to simulate in real-
time. These improvements to traditional simulation methods allow for many new possibilities when considering future work and applications.

7.2 FUTURE WORK

With the successful implementation of a soft body deformation model to simulate a slosh diaphragm, the subject of future applications becomes promising. One of the most important applications to this work is that it can be used as a foundation for creating stiffness maps of real diaphragms. By 3D scanning a particular diaphragm's geometry and measuring its thickness using optical methods, the small non-uniformities in the material can be recorded and inputted into the soft body deformation modeler as a geometry or stiffness map.

The method of applying measured diaphragm properties to the computational model would create a more comprehensive and predictive diaphragm modeler. This model could then be used to predict diaphragm behavior in a variety of conditions such as zero gravity, conditions that would be impractical to test with using a real diaphragm. Additionally, if the stress strain curve for an aged or worn material sample can be measured, those material
properties could then be applied to the computational model along with the geometry of a brand new diaphragm. This would instantly allow for the simulation and prediction of that particular diaphragm's long term future behavior in a now aged state. This application would be particularly useful for failure mode analysis, and iterative design due to the high speed at which these soft body deformation models run. Enabling fast iterative design would improve an engineer's ability accomplish goals such as reducing the amount of diaphragm material required, saving weight.

Because this is the first time a slosh diaphragm soft body deformation model has been created, the computational methods are not fully optimized. The n-body construction of the soft body deformation model lends itself to GPU acceleration, which would markedly speed up the code. Additionally, applying the state of the art in soft body deformation modeling techniques would speed up the code further and improve accuracy, possibly allowing for real-time simulation of these diaphragms to be achieved.

On the subject of increased accuracy, the method for accounting for bending forces in this thesis's soft body deformation model is simplistic and not necessarily the
ideal implementation. Experimenting with different ways to account for bending forces, and replacing my method with a system with more predictive ability of non-linear hyperelastic bending behavior would improve the model.

In conclusion, the deformation model is currently able to predict deformation with a relatively small error. Also, if the soft body deformation method of creating these models is pursued and improved in the future, it would prove to be invaluable to the future of slosh diaphragm engineering and analysis.
REFERENCES


Fig. A.1 ANSYS Transient Structure model deformation, max deformation = 0.0072623 m
Fig. A.2 ANSYS Transient Structure model deformation, max distortion = 0.022054 m
Fig. A.3 ANSYS Transient Structure model deformation, max
deformation = 0.045602 m
Fig. A.4 ANSYS Transient Structure model deformation, max deformation = 0.06871 m
Fig. A.5 ANSYS Transient Structure model deformation, max deformation = 0.13522 m
Fig. A.6 ANSYS Transient Structure model deformation, max deformation = 0.31408 m
Fig. A.7 ANSYS Transient Structure model deformation, max deformation = 0.36669 m
Fig. A.8 ANSYS Transient Structure model deformation, max deformation = 0.40974 m
Fig. A.9 ANSYS Transient Structure model deformation, max deformation = 0.4263 m
APPENDIX B: MATLAB MODEL - MAIN PROGRAM DRIVER

% ElastomerSimulator_Main_Driver:
%   ElastomerSimulator_Main_Driver simulates thin elastic materials
%   using soft body dynamics. Its main application is to spacecraft
%   fuel tank slosh diaphragm simulation during tank draining or
%   filling.
%
%   Created by: Darren V. Levine  4-8-14

clc; clear all; close all; %Resetting the workspace

%% Parameters
dt=1/10000; %[s] Time step size (must satisfy Courant Condition)
etime=5; %[s] End time
GridSize=26; %[# of nodes / grid side length] Cubic grid resolution
ErrorLimit=1; %[%] Percentage of max allowable Strain (Effects Speed)
gravity=[0; 0; -9.81]; %[m/s^2] gravity in the [x y z] directions
Radius=0.20955; %[m] Membrane Radius
Thickness=0.001778; %[m] Membrane Thickness
rhoMembrane=1070; %[kg/m^3] Membrane Density
E=8.6e6; %[Pa] Youngs Modulus
rho=1000; %[kg/m^3] density of liquid

%% Program Settings
record=1; %Record animation (movie)? yes=1 no=0
PlayFPS=0.05/dt; %Frames per second to plot at
MovieFPS=30; %Frames per second to render movie at
FluidHeight=-Radius*1.05; %[m]Initialize fluid height
invertmultiplier=2; % Inversion range: 2 = full, 1=half, 0=none
BendStrainLimiting=1; %Limit bending strain? yes=1 no=0
BendDamping=10; %Relaxation iterations for limiting bending strain
SupportType=1; %Edge support type? Fixed=1, Simply Supported=2;
CollisionDetection=0; %Use collision detection? yes=1 no=0
Cbuffer=1; %Collision distance buffer: 0->1 where 0 is no buffer and 1
%is a buffer the length of the smallest element edge length
ConstraintIterations=1; %Number of coupling iterations to perform on
%constraints, it is recomended that this be changed to a
%integer>1 when collision detection is turned on
SteadyState=1; %Enforce Steady State (Acceleration to 0)? yes=1 no=0

%% Behind the Scenes and Variable Initializations
x=1; y=2; z=3; m=4; %Indexing values for the x y z dimensions and mass
Nodes=GridSize^2; %Total number of particle nodes
FP=(endtime/dt)/PlayFPS:endtime/dt; %Time frames to print
IL=(1/12)*(Thickness^3); %[m^3] Area moment of inertia (I) over element
%length (L) as a function of thickness
vX=1:GridSize,1:GridSize=0; %initialize X velocity
vY=vX; vZ=vX; %initialize Y, Z velocities
aX=vX; aY=vX; aZ=vX; %initialize X, Y, Z acceleration vectors
FX=vX; FY=vX; FZ=vX; %initialize X, Y, Z Force vectors
FbX=vX; FbY=vX; FbZ=vX; %initialize X, Y, Z Bending Force vectors

155
vXprev=vX;vYprev=vX;vZprev=vX; %initialize for verlet motion correction

c=vX; %preallocating c for speed
timesteps=0;
time=0;
frame=0;
FluidHeight0=FluidHeight; %initial fluid Height
timetrack=0;
estimate=0;

%% Check to make sure movie is the appropriate length
Canceling=0;
if record==1
    PromptText=strcat('This will render ',int2str(length(FP)),...
    ' frames for the movie. The movie will be ',...
    int2str(length(FP)/MovieFPS),' seconds long. Continue?');
    % Construct a questdlg with two options
    choice = questdlg(PromptText,...
    'Options',...
    'Yes','Cancel Run','Cancel Run');
    switch choice % Handle response
        case 'Yes'
        case 'Cancel Run'
            Canceling=1;
    end
end
if Canceling==1
    break
end

%% Generates Grid and Finds its Position, Size, and Mass Information:
[X,Y,Z] = GridGenerator(GridSize,Radius); %Generates the grid
Z=-Z;

Xprev=X;Yprev=Y;Zprev=Z; %initialize for verlet motion correction
[NodeArea] = FindNodeArea(X,Y,Z,GridSize); %Finds the node area
NodeLengthScale=NodeArea.^0.5; %Characteristic edge length matrix
Volume=Thickness*NodeArea; %[m^3] Volume matrix
M=rhoMembrane*Volume; %[kg] Mass matrix
%creates mass matrix with zeros instead of infinities for verlet
%motion correction
Mcorrected=M;

%% Choose Which Particles will Be Driven or Fixed
if SupportType==1
    RigidParticles(1,:)=[1:GridSize,1:GridSize,ones(1,GridSize)...
    ,ones(1,GridSize)*GridSize,1:GridSize,1:GridSize,...
    ones(1,GridSize)+1,ones(1,GridSize)*GridSize-1];
    RigidParticles(2,:)=([ones(1,GridSize),ones(1,GridSize)*GridSize...
    ,1:GridSize,1:GridSize,ones(1,GridSize)+1,ones(1,GridSize)...
    *GridSize-1,1:GridSize,1:GridSize];
elseif SupportType==2
    RigidParticles(1,:)=1;
ones(1,GridSize)*GridSize;
RigidParticles(2,:)=[ones(1,GridSize),ones(1,GridSize)*GridSize,...
1:GridSize,1:GridSize];
end

%Initialize Gravity:
gX(1:GridSize,1:GridSize)=gravity(x);
gY(1:GridSize,1:GridSize)=gravity(y);
gZ(1:GridSize,1:GridSize)=gravity(z);

%Set mass of fixed particles to infinity, and remove gravity effect:
for i=1:length(RigidParticles)
    M(RigidParticles(1,i),RigidParticles(2,i))=inf;
    Mcorrected(RigidParticles(1,i),RigidParticles(2,i))=0;
    gX(RigidParticles(1,i),RigidParticles(2,i))=0;
    gY(RigidParticles(1,i),RigidParticles(2,i))=0;
    gZ(RigidParticles(1,i),RigidParticles(2,i))=0;
end
Minv=1./M; %create inverse mass array for computational speed boost

%% Create a Grid Element Length Map
[LengthX,LengthY,LengthPS,LengthNS,LengthXB,LengthYB,Totaldist]...
= GridLengthMap(X,Y,Z,GridSize);

%% Evaluate Solution
while time<endtime
    tic %Start timer
    timesteps=timesteps+1; %Record number of time step iterations
    time=dt*timesteps; %[s] Track time

    % Find Force on Nodes due to Hydrostatic Pressure
    FluidHeight=FluidHeight-invertmultiplier*...
        FluidHeight0/(endtime/dt);%[m] Equivalent Fluid Height
    Pressure=rho*gravity(z)*(Z-FluidHeight); %[kg/(m s^2)] Pressure at
    %node normal to surface

    % Creating Normal Vector Map:
    [NX,NY,NZ] = surfnorm(X,Y,Z);

    % Finding Node Area Map
    [NodeArea] = FindNodeArea(X,Y,Z,GridSize);

    % Find Reaction Forces due to Structural Bending (this part of the
    ...program is a promising area for future work, particularly the
    ...
    %implimentation of new bending solutions to increase accuracy)
        LengthYB,E,IL,NX,NY,NZ);

    %Enforce Steady State?
    if SteadyState==1
        aX=aX*0;aY=aY*0;aZ=aZ*0;
    else
        %correct for actual constraint inclusive movements
        157
vX=(X-Xprev)/dt; vY=(Y-Yprev)/dt; vZ=(Z-Zprev)/dt;
aX=(vX-vXprev)/dt; aY=(vY-vYprev)/dt; aZ=(vZ-vZprev)/dt;

% record previous movements
Xprev=X; Yprev=Y; Zprev=Z;
vXprev=vX; vYprev=vY; vZprev=vZ;
end

%% Combining all forces on the material into FX, FY and FZ
if CriticalDamping==1;
c=2*(Mcorrected.*kfd).^0.5; % For a critically damped spring: F = F - c*v
end
FX=FbX+NodeArea.*Pressure.*NX+Mcorrected.*gX-c.*vX; % [Newtons]
FY=FbY+NodeArea.*Pressure.*NY+Mcorrected.*gY-c.*vY; % [Newtons]
FZ=FbZ+NodeArea.*Pressure.*NZ+Mcorrected.*gZ-c.*vZ; % [Newtons]

%% Solving Physics using explicit Euler integration forward step
aX=aX+FX./M; aY=aY+FY./M; aZ=aZ+FZ./M; % a = F/m
vX=vX+aX*dt; vX=vY+aY*dt; vZ=vZ+aZ*dt; % v = v + a*dt
X=X+vX*dt; Y=Y+vY*dt; Z=Z+vZ*dt; % xyz = xyz + v*dt

%% Courant-Friedrichs-Lewy Condition check
for i = 1:GridSize
    for j = 1:GridSize
        % Courant number at each node:
        c(i,j) = norm([vX(i,j) vY(i,j) vZ(i,j)])*dt/(NodeLengthScale(i,j));
    end
end
CourantNumber=max(max(c)); % Maximum Courant number
if CourantNumber>1
    fprintf('
***ERROR: Courant Number Exceeded 1***
')
    fprintf('Reduce Time step size

')
    break
end

%% Preventing the Surface From Going Past the Fuel Tank Wall
[X,Y,Z] = ContainerCollisionCheck(X,Y,Z,GridSize,Radius);

%% Satisfy Material Spring Relaxation Constraints and Collisions
for ConstraintCoupling=1:ConstraintIterations
    [X,Y,Z] = ConstraintsInPlaneStrain(X,Y,Z,Minv,GridSize,...
    ErrorLimit,LengthX,LengthY,LengthPS,LengthNS,Totaldist);
    if BendStrainLimiting==1
        [X,Y,Z] = BendLimiting(X,Y,Z,Minv,GridSize,LengthXB...,
        LengthYB,BendDamping);
        [X,Y,Z] = ConstraintsInPlaneStrain(X,Y,Z,Minv,GridSize,...
        ErrorLimit,LengthX,LengthY,LengthPS,LengthNS,Totaldist);
    end
    if CollisionDetection==1
        [X,Y,Z] = CollisionCheck(X,Y,Z,GridSize,LengthX,...
        LengthY,Cbuffer);
        [X,Y,Z] = ConstraintsInPlaneStrain(X,Y,Z,Minv,GridSize,...
        ErrorLimit,LengthX,LengthY,LengthPS,LengthNS,Totaldist);
    end
end

%% Preventing the Surface From Going Past the Fuel Tank Wall
[X,Y,Z] = ContainerCollisionCheck(X,Y,Z,GridSize,Radius);
end

%% Status Printing
endtimetrack=toc; %Stop and record timer
timetrack=(timetrack+endtimetrack)/2; %first moving average
estimate=(estimate+(timetrack*(endtime/dt-timesteps)))/2; %Second
%moving average with conversion to seconds for a estimate on time
%remaining for the simulation.
fprintf('h=%g Estimated Time Remaining=%g seconds
'...
,FluidHeight,estimate)

%% Plotting and Movie Recording
if any(timesteps==FP)
    frame=frame+1;
    % Smooth display grid for better visuals and add lighting
    xI=interp2(X,2,'spline');
    yI=interp2(Y,2,'spline');
    zI=interp2(Z,2,'spline');
    SurfacePlot=surf(xI, yI,zI, 'FaceColor', 'interp',...
                      'EdgeColor', 'none');
    colormap(cool)
    camlight right
    view(frame/3+30,45)
    axis equal
    if record==1
        strFrame = sprintf('%05.0f',frame);
        FigTitle = strcat('Frame',strFrame,'.fig');
        FigTitle2 = strcat('Frame',strFrame,'.jpeg');
        saveas(SurfacePlot,FigTitle)
        axis off
        grid off
        pause(0.02)
        set(gcf,'position',get(0,'Screen-size'))
        saveas(SurfacePlot,FigTitle2);
    end
    pause(0.01)
end
APPENDIX C: MATLAB MODEL - GRID GENERATOR

function [X,Y,Z] = GridGenerator(GridSize,Radius)
%------------------------------------------------------------------
% GridGenerator.
% [X,Y,Z] = GridGenerator(GridSize,Radius) returns the position
% components of the 3-D surface (X,Y,Z), for a hemisphere using a
% modified gnomonic projection method when given a cubic grid
% resolution (GridSize), and a sphere radius (Radius).
%------------------------------------------------------------------
% Created by: Darren V. Levine  4-8-14
%------------------------------------------------------------------
x=1;y=2;z=3;
iii=0;
sphereLimit=pi/2.01;
for x0=-sphereLimit:2*sphereLimit/(GridSize-1):sphereLimit;
    for y0=-sphereLimit:2*sphereLimit/(GridSize-1):sphereLimit;
        iii=iii+1;
        aa=Radius*sqrt(3)/3;
x0=aa*tan(x0);
y0=aa*tan(y0);
ra=sqrt(aa^2+xa^2+ya^2);
yyn(iii)=Radius*xa/ra;
zzn(iii)=-Radius*(-aa)/ra;
xxn(iii)=Radius*ya/ra;
    end
end
P=[xxn;ynn;zzn]; % x=P(x,:),y=P(y,:),z=P(z,:)
for i=1:GridSize
    X(i,:)=P(x,1+(i-1)*GridSize:GridSize*i);
    Y(i,:)=P(y,1+(i-1)*GridSize:GridSize*i);
    Z(i,:)=P(z,1+(i-1)*GridSize:GridSize*i);
end

%% Spiral Mapping:
i=round(GridSize/2);
j=i+1;
k=0;
b=0;

k=k+1;
xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j);SM(k,:)=[i;j];
j=j-1;
k=k+1;
xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j);SM(k,:)=[i;j];
while k<GridSize^2-GridSize*2
    b=b+1;
    for n=1:b
        i=i+1;
        j=j;
        k=k+1;xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j);SM(k,:)=[i;j];
end

end
end
b=b+1;
for n=1:b
  i=i;
  j=j+1;
  k=k+1; xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j); SM(k,:)=[i;j];
end
b=b-1;
for n=1:b+1
  i=i-1;
  j=j;
  k=k+1; xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j); SM(k,:)=[i;j];
end
b=b+1;
for n=1:b+1
  i=i;
  j=j-1;
  k=k+1; xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j); SM(k,:)=[i;j];
end
b=b+1;
for n=1:b
  i=i+1;
  j=j;
  k=k+1; xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j); SM(k,:)=[i;j];
end
b=b;
for n=1:b
  i=i;
  j=j+1;
  k=k+1; xS(k)=X(i,j); yS(k)=Y(i,j); zS(k)=Z(i,j); SM(k,:)=[i;j];
end

%% Circular Segment Creation
a=-1;
b=0;
c=1;
for arccount=1:GridSize/2
  a=a+2;
  b=b+2;
  d=a*2+b*2+c-1;
  Arcs(arccount,:)=[c;d];
c=d+1;
end

%% Track angle in 90 degree increments
for i=1:GridSize
  for j=1:GridSize
    if Y(i,j)>0 && X(i,j)>0
      beta(i,j)=atan(Y(i,j)/X(i,j));
    elseif Y(i,j)<0 && X(i,j)>0
      beta(i,j)=360-atan(-Y(i,j)/X(i,j));
    elseif Y(i,j)<0 && X(i,j)<0
      beta(i,j)=180+atan(Y(i,j)/X(i,j));
    end
  end
elseif \( Y(i,j) > 0 \) \&\& \( X(i,j) < 0 \)
    beta(i,j) = 180 - \text{atan}(Y(i,j)/-X(i,j));
else
    \text{fprintf}('error')
    break
end
end

%% Determine Correct Rotational Position
for k=1:GridSize/2-1
    c=Arcs(k,1);
    d=Arcs(k,2);
    kki=SM(c:d,1);
    kkj=SM(c:d,2);
    clear anglearc1
    arcnumber=length(SM(c:d,1));
    for ii=1:arcnumber
        anglearc1(ii)=beta(kki(ii),kkj(ii));
    end
    requiredAngle=[360:-360/arcnumber:360/arcnumber];
    for iii=1:arcnumber-1
        if anglearc1(iii)<anglearc1(iii+1)
            g=iii;
        end
    end
    RequiredAngleOrdered = [requiredAngle(end-g+1:end)...
    requiredAngle(1:end-g)];
    ThetaChange=(RequiredAngleOrdered-anglearc1).*...
    ((Radius-zS(c:d))./Radius);
    % Rotating points to corrected position
    xxm=X;
    yym=Y;
    for ii=1:arcnumber
        xzn(c+ii-1)=xxm(kki(ii),kkj(ii)).*\cos(ThetaChange(ii))...
        -yym(kki(ii),kkj(ii)).*\sin(ThetaChange(ii));
        yyn(c+ii-1)=xxm(kki(ii),kkj(ii)).*\sin(ThetaChange(ii))...
        +yym(kki(ii),kkj(ii)).*\cos(ThetaChange(ii));
    end
end

% operations to the last spiral arm with corrections for the end edge:
k=GridSize/2;
c=Arcs(k,1);
d=length(SM(:,1));
kki=SM(c:d,1);
kkj=SM(c:d,2);
clear anglearc1 requiredAngle RequiredAngleOrdered ThetaChange
arcnumber=length(SM(c:d,1));
for ii=1:arcnumber
anglearc1(ii)=beta(kki(ii),kkj(ii));
end

requiredAngle=[360-45:-270:270/arcnumber+45];
RequiredAngleOrdered=requiredAngle;
ThetaChange=(RequiredAngleOrdered-anglearc1);

% Rotating points to corrected position
xxm=X;
yym=Y;
for ii=1:arcnumber
  xxn(c+ii-1)=xxm(kki(ii),kkj(ii)).*cosd(ThetaChange(ii))...
    -yym(kki(ii),kkj(ii)).*sind(ThetaChange(ii));
  yyn(c+ii-1)=xxm(kki(ii),kkj(ii)).*sind(ThetaChange(ii))...
    +yym(kki(ii),kkj(ii)).*cosd(ThetaChange(ii));
end

% Convert temporary values into global X Y coordinate system
for k=1:length(SM(:,1))
  ki=SM(k,1);
  kj=SM(k,2);
  X(ki,kj)=xxn(k);
  Y(ki,kj)=yyn(k);
end
end
APPENDIX D: MATLAB MODEL – NODE AREA

function [NodeArea] = FindNodeArea(X,Y,Z,GridSize)

% FindNodeArea.
% [NodeArea] = FindNodeArea(X,Y,Z,GridSize) returns the area at each
% node point, this area is an average of the areas of the four faces
% which come into contact with each node point.
% Created by: Darren V. Levine  4-8-14

%% Finding the area at each face patch:
PatchArea(1:GridSize+1,1:GridSize+1)=0; %preallocation
for i = 1:GridSize-1
    for j = 1:GridSize-1
        vec1=[X(i,j) Y(i,j) Z(i,j)];
        vec2=[X(i,j+1) Y(i,j+1) Z(i,j+1)];
        vec3=[X(i+1,j) Y(i+1,j) Z(i+1,j)];
        vec4=[X(i+1,j+1) Y(i+1,j+1) Z(i+1,j+1)];
        aa=vec2-vec1;bb=vec3-vec1;cc=vec4-vec1;
        PatchArea(i+1,j+1)=0.5*(norm(cross(aa,
cc))+norm(cross(bb,cc)));
    end
end

%% Averaging the face areas to create a node area:
NodeArea(1:GridSize,1:GridSize)=0; %preallocation
for i = 0:GridSize-1
    for j = 0:GridSize-1
        NodeArea(i+1,j+1)=(PatchArea(i+1,j+1)+PatchArea(i+2,j+1)+
PatchArea(i+1,j+2)+PatchArea(i+2,j+2))/4;
    end
end
function [LengthX, LengthY, LengthPS, LengthNS, LengthXB, LengthYB, ... 
    Totaldist] = GridLengthMap(X, Y, Z, GridSize)
%----------------------------------------------------------------------
%GridLengthMap.
%   [LengthX, LengthY, LengthPS, LengthNS, LengthXB, LengthYB, Totaldist] = 
%   GridLengthMap(X, Y, Z, GridSize) returns the edge lengths for X 
%   direction in the computational domain (CD), Y direction in CD, 
%   Positive Shear (PS) direction in CD, Negative Shear (NS) direction 
%   in CD, Bending Length in the X direction in CD (LengthXB), and 
%   Bending Length in the Y direction in CD (LengthYB). The summation 
% of the structural and shear lengths are also calculated 
% (Totaldist), in order to later function as a displacement check. 
% 
%   Created by: Darren V. Levine    4-8-14
%----------------------------------------------------------------------

%% X structural
LengthX(1:GridSize,1:GridSize)=0;
for i=1:GridSize
    for j=1:GridSize-1
        P1=[X(i,j);Y(i,j);Z(i,j)];
        P2=[X(i,j+1);Y(i,j+1);Z(i,j+1)];
        delta=P2-P1;
        LengthX(i,j) = sqrt(dot(delta,delta));
    end
end

%% Y structural
LengthY(1:GridSize,1:GridSize)=0;
for i=1:GridSize-1
    for j=1:GridSize
        P1=[X(i,j);Y(i,j);Z(i,j)];
        P2=[X(i+1,j);Y(i+1,j);Z(i+1,j)];
        delta=P2-P1;
        LengthY(i,j) = sqrt(dot(delta,delta));
    end
end

%% Positive Shear
LengthPS(1:GridSize,1:GridSize)=0;
for i=1:GridSize-1
    for j=1:GridSize-1
        P1=[X(i,j);Y(i,j);Z(i,j)];
        P2=[X(i+1,j+1);Y(i+1,j+1);Z(i+1,j+1)];
        delta=P2-P1;
        LengthPS(i,j) = sqrt(dot(delta,delta));
    end
end

%% Negative Shear
LengthNS(1:GridSize,1:GridSize)=0;
for i=2:GridSize
    for j=1:GridSize-1
        P1=[X(i,j);Y(i,j);Z(i,j)];
        P2=[X(i-1,j+1);Y(i-1,j+1);Z(i-1,j+1)];
        delta=P2-P1;
        LengthNS(i,j) = sqrt(dot(delta,delta));
    end
end

Totaldist=sum(sum(abs(LengthX)))+sum(sum(abs(LengthY)))+
    sum(sum(abs(LengthPS)))+sum(sum(abs(LengthNS)));

%% X Bending
LengthXB(1:GridSize,1:GridSize)=0;
for i=1:GridSize
    for j=1:GridSize-2
        P1=[X(i,j);Y(i,j);Z(i,j)];
        P2=[X(i,j+2);Y(i,j+2);Z(i,j+2)];
        delta=P2-P1;
        LengthXB(i,j)=sqrt(dot(delta,delta));
    end
end

%% Y Bending
LengthYB(1:GridSize,1:GridSize)=0;
for i=1:GridSize-2
    for j=1:GridSize
        P1=[X(i,j);Y(i,j);Z(i,j)];
        P2=[X(i+2,j);Y(i+2,j);Z(i+2,j)];
        delta=P2-P1;
        LengthYB(i,j)=sqrt(dot(delta,delta));
    end
end
APPENDIX F: MATLAB MODEL - STRUCTURAL FORCES

function [FbX,FbY,FbZ] = StructuralForces(X,Y,Z,GridSize,LengthXB,...
   LengthYB,E,IL,NX,NY,NZ)
%------------------------------------------------------------------
%StructuralForces.
%   [FbX,FbY,FbZ] = StructuralForces(X,Y,Z,GridSize,LengthXB,LengthYB,
%   E,IL,NX,NY,NZ) returns the structureal forces on each node which
% result from and oppose the bending of the grid elements.
%------------------------------------------------------------------
% Created by: Darren V. Levine   4-8-14
%------------------------------------------------------------------

FbX(1:GridSize,1:GridSize)=0; %initialize X Force vector
FbY=FbX; %initialize Y Force vector
FbZ=FbX; %initialize Z Force vector
FnormX(1:GridSize,1:GridSize)=0;
FnormY(1:GridSize,1:GridSize)=0;
FnormZ(1:GridSize,1:GridSize)=0;
FnormX2(1:GridSize,1:GridSize)=0;
FnormY2(1:GridSize,1:GridSize)=0;
FnormZ2(1:GridSize,1:GridSize)=0;

%% X Bending
for i=1:GridSize
   for j=1:GridSize-2
      P1=[X(i,j);Y(i,j);Z(i,j)];
      P2=[X(i,j+2);Y(i,j+2);Z(i,j+2)];
      delta=P2-P1;
      deltaLength = sqrt(dot(delta,delta));
      d=abs(real(sqrt((LengthXB(i,j)/2)^2-(deltaLength/2)^2)));
      Keq=48*E*IL/(LengthXB(i,j)^2); %Keq=48*E*(I/L)/(L^2);
      Pc=(P2+P1)/2;
      deltaLength = sqrt(dot(delta,delta));
      d=abs(real(sqrt((LengthXB(i,j)/2)^2-(deltaLength/2)^2)));
      Keq=48*E*IL/(LengthXB(i,j)^2); %Keq=48*E*(I/L)/(L^2);
      Pm=[NX(i,j+1) NY(i,j+1) NZ(i,j+1)]';
      Pc=[NX(i,j+1) NY(i,j+1) NZ(i,j+1)]';
      if abs(norm(Pm-(Pc+N)))>abs(norm(Pm-(Pc-N)))
         signX=1;
      else
         signX=-1;
      end
      temp1=(signX*d*Keq);
      FnormX(i,j+1)=temp1*NX(i,j+1);
      temp2=(signX*d*Keq);
      FnormY(i,j+1)=temp2*NY(i,j+1);
      temp3=(signX*d*Keq);
      FnormZ(i,j+1)=temp3*NZ(i,j+1);
   end
end
FbX=FnormX;
FbY=FnormY;
FbZ=FnormZ;
%% Y Bending
for i=1:GridSize-2
for j=1:GridSize
    P1=[X(i,j);Y(i,j);Z(i,j)];
    P2=[X(i+2,j);Y(i+2,j);Z(i+2,j)];
    delta=P2-P1;
    deltaLength = sqrt(dot(delta,delta));
    d=abs(real(sqrt((LengthYB(i,j)/2)^2-(deltaLength/2)^2)));
    Keq=48*E*IL/(LengthYB(i,j)^2); %Keq=48*E*(I/L)/(L^2);
    Pc=(P2+P1)/2;
    N=[NX(i+1,j) NY(i+1,j) NZ(i+1,j)]';
    Pm=[X(i+1,j) Y(i+1,j) Z(i+1,j)]';
    if abs(norm(Pm-(Pc+N)))>abs(norm(Pm-(Pc-N)))
        signY=1;
    else
        signY=-1;
    end
    temp4=(signY*d*Keq);
    FnormX2(i+1,j)=temp4*NX(i+1,j);
    temp5=(signY*d*Keq);
    FnormY2(i+1,j)=temp5*NY(i+1,j);
    temp6=(signY*d*Keq);
    FnormZ2(i+1,j)=temp6*NZ(i+1,j);
end
end
FbX=FbX+FnormX2;
FbY=FbY+FnormY2;
FbZ=FbZ+FnormZ2;
end
APPENDIX G: MATLAB MODEL - IN PLANE STRAIN

LIMITING CONSTRAINTS

function \([X,Y,Z] = \text{ConstraintsInPlaneStrain}(X,Y,Z,\text{Minv},\text{GridSize},...\)
\text{ErrorLimit,} \text{RestX,RestY,RestPS,RestNS,Totaldist})

\%----------------------------------------------------------------------------
\%ConstraintsInPlaneStrain.
\% [X,Y,Z] = ConstraintsInPlaneStrain(X,Y,Z,\text{Minv},\text{GridSize},\text{ErrorLimit},
\% \text{RestX,RestY,RestPS,RestNS,Totaldist}) returns the solution grid
\% after applying a strain limiting constraint in-plane to the grid
\% elements. The strain is limited by the allowable strain percentage
\% (\text{ErrorLimit}). The constraints operate using a relaxation technique
\% where perfectly damped springs are attached to each node mass. When
\% all spring lengths agree within an acceptable deformation limit,
\% the constraint is considered fully applied.
\%
\% Created by: Darren V. Levine 4-8-14
\%----------------------------------------------------------------------------

errorA=100; %initialize Error
its=0;
while errorA>\text{ErrorLimit}/100
    its=its+1;

    \% X stretching
    \text{xxxA}(1:\text{GridSize},1:\text{GridSize})=0;
    \text{yyyA} = \text{xxxA};
    \text{zzzA} = \text{xxxA};
    \text{xxxS} = \text{xxxA};
    \text{yyyS} = \text{xxxA};
    \text{zzzS} = \text{xxxA};

    \text{xxx}=X;
    \text{yyy}=Y;
    \text{zzz}=Z;
    \text{for } i=1:\text{GridSize}
        \text{for } j=1:(\text{GridSize}-1)
            \text{if } \text{Minv}(i,j)+\text{Minv}(i,j+1)==0
                \text{xxxA}(i,j)=0; \text{yyyA}(i,j)=0; \text{zzzA}(i,j)=0;
                \text{xxxS}(i,j+1)=0; \text{yyyS}(i,j+1)=0; \text{zzzS}(i,j+1)=0;
                \text{errorB}(i,j)=0;
            \text{else}
                \text{P1}=[\text{xxx}(i,j);\text{yyy}(i,j);\text{zzz}(i,j)];
                \text{P2}=[\text{xxx}(i,j+1);\text{yyy}(i,j+1);\text{zzz}(i,j+1)];
                \text{delta} = \text{P2} - \text{P1};
                \text{deltaLength} = \sqrt{\text{dot}(\text{delta},\text{delta})};
                \text{diff} = (\text{deltaLength}-\text{RestX}(i,j))/...\text{Totaldist});
                \text{errorB}(i,j)=\text{deltaLength}-\text{RestX}(i,j);
                \text{AddP}=\text{delta}.*\text{diff}.*\text{Minv}(i,j);
                \%P1=P1+AddP;
                \text{SubP}=\text{delta}.*\text{diff}.*\text{Minv}(i,j+1);
                \%P2=P2-\text{SubP};
            \text{end} 
            \text{end} 
        \text{end} 
    \end{verbatim}
xxxA(i,j)=AddP(1);
yyyA(i,j)=AddP(2);
zzzA(i,j)=AddP(3);
xxxS(i,j+1)=SubP(1);
yyyS(i,j+1)=SubP(2);
zzzS(i,j+1)=SubP(3);
end
end
end
X=X+xxxA-xxxS;Y=Y+yyyA-yyyS;Z=Z+zzzA-zzzS;

errorC=sum(sum(abs(errorB)));% Y stretching
xxxA(1:GridSize,1:GridSize)=0;
yyyA=xxxA;
zzzA=xxxA;
xxxS=xxxA;
yyyS=xxxA;
zzzS=xxxA;

xxx=X;
yyy=Y;
zzz=Z;
for i=1:GridSize-1
    for j=1:GridSize
        if Minv(i,j)+Minv(i+1,j)==0
            xxxA(i,j)=0;yyyA(i,j)=0;zzzA(i,j)=0;
            xxxS(i+1,j)=0;yyyS(i+1,j)=0;zzzS(i+1,j)=0;
            errorE(i,j)=0;
        else
            P1=[xxx(i,j);yyy(i,j);zzz(i,j)];
            P2=[xxx(i+1,j);yyy(i+1,j);zzz(i+1,j)];
            delta=P2-P1;
            deltaLength = sqrt(dot(delta,delta));
            diff = (deltaLength-RestY(i,j))./...
                    (deltaLength*(Minv(i,j)+Minv(i+1,j)));
            errorE(i,j)=deltaLength-RestY(i,j);
            AddP=delta.*diff.*Minv(i,j); %P1=P1+AddP;
            SubP=delta.*diff.*Minv(i+1,j); %P2=P2-SubP;
            xxxA(i,j)=AddP(1);
            yyyA(i,j)=AddP(2);
            zzzA(i,j)=AddP(3);
            xxxS(i+1,j)=SubP(1);
            yyyS(i+1,j)=SubP(2);
            zzzS(i+1,j)=SubP(3);
        end
    end
end
X=X+xxxA-xxxS;Y=Y+yyyA-yyyS;Z=Z+zzzA-zzzS;
errorD=errorC+sum(sum(abs(errorE)));% PosShear
xxxA(1:GridSize,1:GridSize)=0;
yyyA=xxxA;
zzzA=xxxA;
xxxS=xxxA;
yyyS=xxxA;
zzzS=xxxA;

xxx=X;
yyy=Y;
zzz=Z;

for i=1:GridSize-1
    for j=1:GridSize-1
        if Minv(i,j)+Minv(i+1,j+1)==0
            xxxA(i,j)=0; yyyA(i,j)=0; zzzA(i,j)=0;
            xxxS(i+1,j+1)=0; yyyS(i+1,j+1)=0; zzzS(i+1,j+1)=0;
            errorF(i,j)=0;
        else
            P1=[xxx(i,j);yyy(i,j);zzz(i,j)];
            P2=[xxx(i+1,j+1);yyy(i+1,j+1);zzz(i+1,j+1)];
            delta=P2-P1;
            deltaLength = sqrt(dot(delta,delta));
            diff = (deltaLength-RestPS(i,j))./...
                (deltaLength*Minv(i,j)+Minv(i+1,j+1));
            errorF(i,j)=deltaLength-RestPS(i,j);
            AddP=delta.*diff.*Minv(i,j); %P1=P1+AddP;
            SubP=delta.*diff.*Minv(i+1,j+1); %P2=P2-SubP;
            xxxA(i,j)=AddP(1);
            yyyA(i,j)=AddP(2);
            zzzA(i,j)=AddP(3);
            xxxS(i+1,j+1)=SubP(1);
            yyyS(i+1,j+1)=SubP(2);
            zzzS(i+1,j+1)=SubP(3);
        end
    end
end
X=X+xxxA-xxxS;Y=Y+yyyA-yyyS;Z=Z+zzzA-zzzS;
errorG=errorD+sum(sum(abs(errorF)));%

%% NegShear
xxxA(1:GridSize,1:GridSize)=0;
yyyA=xxxA;
zzzA=xxxA;
xxxS=xxxA;
yyyS=xxxA;
zzzS=xxxA;

xxx=X;
yyy=Y;
zzz=Z;

for i=2:GridSize
    for j=1:GridSize-1
        if Minv(i,j)+Minv(i-1,j+1)==0
            xxxA(i,j)=0; yyyA(i,j)=0; zzzA(i,j)=0;
            xxxS(i-1,j+1)=0; yyyS(i-1,j+1)=0; zzzS(i-1,j+1)=0;
            errorH(i,j)=0;
        end
    end
end

171
else
    P1=[xxx(i,j);yyy(i,j);zzz(i,j)];
    P2=[xxx(i-1,j+1);yyy(i-1,j+1);zzz(i-1,j+1)];
    delta=P2-P1;
    deltaLength = sqrt(dot(delta,delta));
    diff = (deltaLength-RestNS(i,j))./...
    (deltaLength*(Minv(i,j)+Minv(i-1,j+1)));
    errorH(i,j)=deltaLength-RestNS(i,j);
    AddP=delta.*diff.*Minv(i,j); %P1=P1+AddP;
    SubP=delta.*diff.*Minv(i-1,j+1); %P2=P2-SubP;
    xxxA(i,j)=AddP(1);
    yyyA(i,j)=AddP(2);
    zzzA(i,j)=AddP(3);
    xxxS(i-1,j+1)=SubP(1);
    yyyS(i-1,j+1)=SubP(2);
    zzzS(i-1,j+1)=SubP(3);
end
end
end
X=X+xxxA-xxxS; Y=Y+yyyA-yyyS; Z=Z+zzzA-zzzS;
errorI=errorG+sum(sum(abs(errorH)));
errorA=errorI/Totaldist;
end
end
APPENDIX H: MATLAB MODEL - OUT OF PLANE STRAIN

LIMITING CONSTRAINTS

function \([X,Y,Z] = \)
BendLimiting(X,Y,Z,Minv,GridSize,RestXB,RestYB,BendDamping)
%----------------------------------------------------------------------
% BendLimiting.
% \([X,Y,Z] = \) BendLimiting(X,Y,Z,Minv,GridSize,RestXB,RestYB) returns
% the solution grid after applying a strain limiting constraint
% out-of-plane to the grid elements. The strain is limited by a
% number of defined relaxation iterations. The constraints operate
% using a relaxation technique where perfectly damped springs are
% attached between every other node mass. This produces a strain
% limit on bending (deflection in the out-of-plane direction).
% % Created by: Darren V. Levine  4-8-14
%----------------------------------------------------------------------

for dampingIterations=1:BendDamping
  %% X Bending
  xxxA(1:GridSize,1:GridSize)=0;
  yyyA=xxxA;
  zzzA=xxxA;
  xxxS=xxxA;
  yyyS=xxxA;
  zzzS=xxxA;

  xxx=X;
  yyy=Y;
  zzz=Z;
  for i=1:GridSize
    for j=1:GridSize-2
      if Minv(i,j)+Minv(i,j+2)==0
        xxxA(i,j)=0; yyyA(i,j)=0; zzzA(i,j)=0;
        xxxS(i,j+2)=0; yyyS(i,j+2)=0; zzzS(i,j+2)=0;
      else
        P1=[xxx(i,j);yyy(i,j);zzz(i,j)];
        P2=[xxx(i,j+2);yyy(i,j+2);zzz(i,j+2)];
        delta=P2-P1;
        deltaLength = sqrt(dot(delta,delta));
        diff = (deltaLength-RestXB(i,j))./...
                   (deltaLength*Minv(i,j)+Minv(i,j+2));
        errorH(i,j)=deltaLength-RestXB(i,j);
        AddP=delta.*diff.*Minv(i,j); %P1=P1+AddP;
        SubP=delta.*diff.*Minv(i,j+2); %P2=P2-SubP;
        xxxA(i,j)=AddP(1);
        yyyA(i,j)=AddP(2);
        zzzA(i,j)=AddP(3);
        xxxS(i,j+2)=SubP(1);
        yyyS(i,j+2)=SubP(2);
        zzzS(i,j+2)=SubP(3);
      end
    end
  end
end

173
end
end
end
X=X+xxxA-xxxS;
Y=Y+yyyA-yyyS;
Z=Z+zzzA-zzzS;

%% Y Bending

xxxA(1:GridSize,1:GridSize)=0;
yyyA=xxxA;
zzzA=xxxA;
xxxS=xxxA;
yyyS=xxxA;
zzzS=xxxA;

xxx=X;
yyy=Y;
zzz=Z;
for i=1:GridSize-2
  for j=1:GridSize
    if Minv(i,j)+Minv(i+2,j)==0
      xxxA(i,j)=0; yyyA(i,j)=0; zzzA(i,j)=0;
      xxxS(i+2,j)=0; yyyS(i+2,j)=0; zzzS(i+2,j)=0;
    else
      P1=[xxx(i,j);yyy(i,j);zzz(i,j)];
      P2=[xxx(i+2,j);yyy(i+2,j);zzz(i+2,j)];
      delta=P2-P1;
      deltaLength = sqrt(dot(delta,delta));
      diff = (deltaLength-RestYB(i,j))./...
        (deltaLength*(Minv(i,j)+Minv(i+2,j)));
      errorH(i,j)=deltaLength-RestYB(i,j);
      AddP=delta.*diff.*Minv(i,j); %P1=P1+AddP;
      SubP=delta.*diff.*Minv(i+2,j); %P2=P2-SubP;
      xxxA(i,j)=AddP(1);
      yyyA(i,j)=AddP(2);
      zzzA(i,j)=AddP(3);
      xxxS(i+2,j)=SubP(1);
      yyyS(i+2,j)=SubP(2);
      zzzS(i+2,j)=SubP(3);
    end
  end
end
end
X=X+xxxA-xxxS;
Y=Y+yyyA-yyyS;
Z=Z+zzzA-zzzS;
end
end
APPENDIX I: MATLAB MODEL - COLLISION HANDLING

function \[X,Y,Z\] = CollisionCheck(X,Y,Z,GridSize,RestX,RestY,Cbuffer)
\%---------------------------------------------------------------------
\%CollisionCheck.
\% [X,Y,Z] = CollisionCheck(X,Y,Z,GridSize,RestX,RestY,Cbuffer)
\% returns the solution grid after checking each grid point to see if
\% the point is within a distance range to any other points and then
\% places it away from that "intersecting" point. The distance at
\% which a collision can occur is given by Cbuffer (Collision distance
\% buffer) which can have a range of 0 to 1, where 0 is no buffer and
\% 1 is a buffer the length of the smallest element edge length.
\% Created by: Darren V. Levine  4-8-14
\%---------------------------------------------------------------------

for i=1:GridSize
    for j=1:GridSize
        if RestX(i,j)==0;
            RestX(i,j)=inf;
        end
        if RestY(i,j)==0;
            RestY(i,j)=inf;
        end
    end
end
mindist=min([min(min(RestX)) min(min(RestY)))]*Cbuffer;
c=0;
for i=1:GridSize
    for j=1:GridSize
        P1=[X(i,j); Y(i,j); Z(i,j)];
        for ii=1:GridSize
            for jj=1:GridSize
                P2=[X(ii,jj); Y(ii,jj); Z(ii,jj)];
                if (i==ii & j==jj) || (i+1==ii & j==jj)...
                    || (i==ii & j+1==jj) || (i+1==ii & j+1==jj) ||...
                    (i-1==ii & j==jj) || (i==ii & j-1==jj) || (i-1==ii & j-1==jj)
                    else
                        if norm(P2-P1)<mindist
                            c=c+1;
                            ClsePts(c,:)=[i;j;ii;jj];
                        end
                    end
    end
end
xxxA(1:GridSize,1:GridSize)=0;
yyyA(1:GridSize,1:GridSize)=0;
zzzA(1:GridSize,1:GridSize)=0;
xxxS(1:GridSize,1:GridSize)=0;
yyyS(1:GridSize,1:GridSize)=0;
zzzS(1:GridSize,1:GridSize)=0;
if c>0
    for i=1:c
        x=ClsePts(i,1);
        y=ClsePts(i,2);
        xx=ClsePts(i,3);
        yy=ClsePts(i,4);
        P3=[X(x,y) Y(x,y) Z(x,y)];
        P4=[X(xx,yy) Y(xx,yy) Z(xx,yy)];
        delta=P4-P3;
        deltaLength = sqrt(dot(delta,delta));
        diff = (deltaLength-mindist)./deltaLength;
        AddP=delta.*diff.*0.5; %P1=P1+AddP;
        SubP=delta.*diff.*0.5; %P2=P2-SubP;
        xxxA(x,y)=AddP(1);
        yyyA(x,y)=AddP(2);
        zzzA(x,y)=AddP(3);
        xxxS(xx,yy)=SubP(1);
        yyyS(xx,yy)=SubP(2);
        zzzS(xx,yy)=SubP(3);
    end
end

X=X+xxxA-xxxS;
Y=Y+yyyA-yyyS;
Z=Z+zzzA-zzzS;
end

function [X,Y,Z] = ContainerCollisionCheck(X,Y,Z,GridSize,Radius)
%---------------------------------------------------------------------
%ContainerCollisionCheck:
%   [X,Y,Z] = ContainerCollisionCheck(X,Y,Z,GridSize,Radius) inforces
%   a collision boundary at the tank radius preventing the grid from
%   expanding into a volume greater than the tank
%
% Created by: Darren V. Levine  4-8-14
%---------------------------------------------------------------------
for i=2:GridSize-1
    for j=2:GridSize-1
        vectormag=norm([X(i,j) Y(i,j) Z(i,j)]);
        if abs(vectormag)>Radius*1.001
            vectordist=[X(i,j) Y(i,j) Z(i,j)];
            DirectionV=vectordist/vectormag;
            X(i,j)=Radius*DirectionV(1);
            Y(i,j)=Radius*DirectionV(2);
            Z(i,j)=Radius*DirectionV(3);
        end
    end
end
end

176
function [Xfig,Yfig,Zfig] = FigureDATA(FigFilePath,FrameNumber)
%%---------------------------------------------------------------------
%FigureDATA:
% [Xfig,Yfig,Zfig] = FigureDATA(FigFilePath,FrameNumber) returns the
% 3D vector data from a matlab file.
% FigFilePath='path/to/your/figure.fig'
% Requires image name formatings=
% For example:'Frame00135.fig' where FrameNumber = 135
% Created by: Darren V. Levine  4-8-14
%%---------------------------------------------------------------------
K=sprintf('/Frame%05.0f.fig',FrameNumber);
K=strcat(FigFilePath,K);
myFigStruct = load(K,'-MAT');
Xfig=myFigStruct(1).hgS_070000(1).children(1).children(1).properties(1)
   .XData;
Yfig=myFigStruct(1).hgS_070000(1).children(1).children(1).properties(1)
   .YData;
Zfig=myFigStruct(1).hgS_070000(1).children(1).children(1).properties(1)
   .ZData;
end

% CompareData.m:
% CompareData processes and compares 3D scan data in .obj format to
% 3D surf data from .fig files in a slosh diaphragm experiment
% Created by: Darren V. Levine  4-8-14
%_roi -roi -vmin 0.0005 -vmax 0.75 -cab 0.35
clc;clear all; close all;
pause(0.1);

%% Parameters:
Percent=90; %Percent fill volume in experimental data
interptime=0; %Interpolate position? Yes=1 No=0;
r=0.20955; %[m] diaphragm Radius
findvolumeFrame=1; %find volume? yes=1 no=0;
FinalFrameNumber=500;
FigFilePath='C:\Users\Darren\Thesis Simulations\sim 11 grid26 strain1
Fixedsupported bending10';
OBJfolder='C:\Users\Darren\Thesis Simulations\3DScanDataOBJS\';
%output path for resulting images:
Outpath='C:\Users\Darren\Thesis Simulations\ScanSim 11 V2\';
if Percent==100
    Objectname='coarse_100p.obj';
elseif Percent==90
    Objectname='coarse_090p.obj';
elseif Percent==80
    Objectname='Flat_080p_comb_trim_copy_final.obj';
else
    Objectname='Flat_080p_comb_trim_copy_final.obj';
end

177
elseif Percent==70
    Objectname='Flat_070p_coarse.obj';
elseif Percent==60
    Objectname='Flat_060p_coarse.obj';
elseif Percent==50
    Objectname='Flat_050p_coarse.obj';
elseif Percent==40
    Objectname='Flat_040p_comb_trim_copy_final.obj';
end

ExperimentalDataPath=sprintf('%s%s',OBJfolder,Objectname);
HalfSphere=pi*(r^3)*2/3;

if findvolumeFrame==1
    %find the full fill volume
    FrameNumber=FinalFrameNumber;%Frame Number of Simulated geometry
    % Find the Fluid Volume
    [Xfig,Yfig,Zfig] = FigureDATA(FigFilePath,FrameNumber);
    GridSize=length(Xfig(1,:));
    [NX,NY,NZ] = surfnorm(Xfig,Yfig,Zfig);
    for i = 1:GridSize-1
        for j = 1:GridSize-1
            vec1=[Xfig(i,j) Yfig(i,j) 0];
            vec2=[Xfig(i,j+1) Yfig(i,j+1) 0];
            vec3=[Xfig(i+1,j) Yfig(i+1,j) 0];
            vec4=[Xfig(i+1,j+1) Yfig(i+1,j+1) 0];
            aa=vec2-vec1;bb=vec3-vec1;cc=vec4-vec1;
            PatchAreaZ=0.5*(norm(cross(aa, cc))+norm(cross(bb,cc)));
            NZmidFace=sum([NZ(i,j) NZ(i,j+1) NZ(i+1,j+1) NZ(i+1,j)])/4;
            CorrectedArea=PatchAreaZ*NZmidFace/abs(NZmidFace);
            zMidpt=sum([Zfig(i,j) Zfig(i,j+1) Zfig(i+1,j+1) ...
                        Zfig(i+1,j)])/4;
            VolumePatch(i,j)=zMidpt.*CorrectedArea;
        end
    end
    GridVolumeZ=sum(sum(VolumePatch));
    FullFillVolume=GridVolumeZ+HalfSphere; %Volume when 100% full
    DesiredVolume=FullFillVolume*Percent/100;
    IntFrameNumber=round(FinalFrameNumber-((FullFillVolume... -DesiredVolume)/((FullFillVolume-0)/(FinalFrameNumber-1))));

    FrameNumber=IntFrameNumber;
    a=0;
    exitcond=0;
    iii=0;
    while exitcond==0
        iii=iii+1;
        % Find the Fluid Volume
        [Xfig,Yfig,Zfig] = FigureDATA(FigFilePath,FrameNumber);
        GridSize=length(Xfig(1,:));
        [NX,NY,NZ] = surfnorm(Xfig,Yfig,Zfig);
        for i = 1:GridSize-1
            for j = 1:GridSize-1
...
vec1=[Xfig(i,j) Yfig(i,j) 0];
vec2=[Xfig(i,j+1) Yfig(i,j+1) 0];
vec3=[Xfig(i+1,j) Yfig(i+1,j) 0];
vec4=[Xfig(i+1,j+1) Yfig(i+1,j+1) 0];
aa=vec2-vec1;bb=vec3-vec1;cc=vec4-vec1;
PatchAreaZ=0.5*(norm(cross(aa, cc))+...norm(cross(bb,cc)));
NZmidFace=sum([NZ(i,j) NZ(i,j+1) NZ(i+1,j+1)...NZ(i+1,j)])/4;
CorrectedArea=PatchAreaZ*NZmidFace/abs(NZmidFace);
zMidpt=sum([Zfig(i,j) Zfig(i,j+1) Zfig(i+1,j+1)...Zfig(i+1,j)])/4;
VolumePatch(i,j)=zMidpt.*CorrectedArea;
end
GridVolumeZ=sum(sum(VolumePatch));
VolumeR=GridVolumeZ+HalfSphere; %total volume
VolCheck=DesiredVolume-VolumeR;
VolNum(iii,:)=[abs(VolCheck) FrameNumber];
ap=a;
if VolCheck>0
a=1;
else
a=-1;
end
if ap==ap
exitcond=1;
end
end
jjj=find(min(VolNum(:,1))==VolNum(:,1));
FrameNumber=VolNum(jjj,2);
%% Get Data From .fig File:
[Xfig1,Yfig1,Zfig1] = FigureDATA(FigFilePath,FrameNumber);
if interptime==1
FrameNumber=FrameNumber+1;
[Xfig2,Yfig2,Zfig2] = FigureDATA(FigFilePath,FrameNumber);
Xfig=(Xfig1+Xfig2)/2;
Yfig=(Yfig1+Yfig2)/2;
Zfig=(Zfig1+Zfig2)/2;
else
Xfig=Xfig1;
Yfig=Yfig1;
Zfig=Zfig1;
end
%% Find the Fluid Volume
GridSize=length(Xfig(1,:));
[NX,NY,NZ] = surfnorm(Xfig,Yfig,Zfig);
for i = 1:GridSize-1
for j = 1:GridSize-1
vec1=[Xfig(i,j) Yfig(i,j) 0];

vec2=[Xfig(i,j+1) Yfig(i,j+1) 0];
vec3=[Xfig(i+1,j) Yfig(i+1,j) 0];
vec4=[Xfig(i+1,j+1) Yfig(i+1,j+1) 0];
aa=vec2-vec1;bb=vec3-vec1;cc=vec4-vec1;
PatchAreaZ=0.5*(norm(cross(aa, cc))+norm(cross(bb,cc)));

NZmidFace=sum([NZ(i,j) NZ(i,j+1) NZ(i+1,j+1) NZ(i+1,j)])/4;
CorrectedArea=PatchAreaZ*NZmidFace/abs(NZmidFace);

zMidpt=sum([Zfig(i,j) Zfig(i,j+1) Zfig(i+1,j+1) Zfig(i+1,j)])/4;
VolumePatch(i,j)=zMidpt.*CorrectedArea;
end
end

%% Compare and Check Fluid Volume
TargetVolume=FullFillVolume*Percent/100;
GridVolumeZ=sum(sum(VolumePatch));
TotalActualVolume=GridVolumeZ+HalfSphere;
fprintf('TargetVolume=%g
TotalActualVolume=%g
FrameNumber=%g
',...
TargetVolume,TotalActualVolume,FrameNumber)

% break %uncomment this if you wish to check the volume

%% Get Data From .obj file
[V,F3,F4]=loadawobj(ExperimentalDataPath);

%% Format .obj Data in Preparation for Comparison
clear GridSize Xobj Yobj Zobj
GridSize=round(length(V(1,:))^0.5)-1;
for i=1:GridSize
    Xobj(i,:)=V(1,1+(i-1)*GridSize:GridSize*i);
    Yobj(i,:)=V(2,1+(i-1)*GridSize:GridSize*i);
    Zobj(i,:)=V(3,1+(i-1)*GridSize:GridSize*i);
end

eavename=sprintf('Data%g.mat',Percent);
save(savename); %create a save point because the code runs slowly

%% Load a save point{ to load: start code from here and uncomment data
% clc;clear all;close all
% load Data100.mat
% load Data90.mat
% load Data80.mat
% load Data70.mat
% load Data60.mat
% load Data50.mat
% load Data40.mat
%
TransformGrid=0; %edit grid? Yes=1; No=0; (use if grids are misaligned)
%% Scale .obj Data
Xcomp=Xobj/500*r/0.5;
Ycomp=Yobj/500*r/0.5;
Zcomp=Zobj/500*r/0.5;

%% Level .obj Data
if Percent==70
    thetaz=50*pi/180; % Rotate .obj data around z axis if necessary
    thetax=3*pi/180; % Rotate .obj data around x axis if necessary
    thetay=-7*pi/180; % Rotate .obj data around y axis if necessary
    aaa=0.5; % Translate .obj Data
    eee=0.5;
    bbb=0.1; % title left vertical adjustment
    ccc=0.11; % title right vertical adjustment
elseif Percent==60
    thetaz=66*pi/180; % Rotate .obj data around z axis if necessary
    thetax=5*pi/180; % Rotate .obj data around x axis if necessary
    thetay=5*pi/180; % Rotate .obj data around y axis if necessary
    aaa=0;% Translate .obj Data
    eee=0;
    bbb=0.08; % title left vertical adjustment
    ccc=0.09; % title right vertical adjustment
elseif Percent==50
    thetaz=14*pi/180; % Rotate .obj data around z axis if necessary
    thetax=-7*pi/180; % Rotate .obj data around x axis if necessary
    thetay=-5*pi/180; % Rotate .obj data around y axis if necessary
    aaa=0.02;% Translate .obj Data
    eee=0;
    bbb=0.05; % title left vertical adjustment
    ccc=0.07; % title right vertical adjustment
elseif Percent==40
    thetaz=35*pi/180; % Rotate .obj data around z axis if necessary
    thetax=-3*pi/180; % Rotate .obj data around x axis if necessary
    thetay=0*pi/180; % Rotate .obj data around y axis if necessary
    aaa=0.045;% Translate .obj Data
    eee=0;
    bbb=0.04; % title left vertical adjustment
    ccc=0.07; % title right vertical adjustment
elseif Percent==80
    thetaz=20*pi/180; % Rotate .obj data around z axis if necessary
    thetax=3.5*pi/180; % Rotate .obj data around x axis if necessary
    thetay=-8*pi/180; % Rotate .obj data around y axis if necessary
    aaa=-0.02;% Translate .obj Data
    eee=0;
    bbb=0.1; % title left vertical adjustment
    ccc=0.11; % title right vertical adjustment
elseif Percent==90
    thetaz=44*pi/180; % Rotate .obj data around z axis if necessary
    thetax=-10*pi/180; % Rotate .obj data around x axis if necessary
    thetay=2*pi/180; % Rotate .obj data around y axis if necessary
    aaa=-0.03;% Translate .obj Data
    eee=0;
    bbb=0.1; % title left vertical adjustment
else
ccc=0.11; %title right vertical adjustment

elseif Percent==100
    thetaz=44*pi/180; %Rotate .obj data around z axis if necessary
    thetax=-4*pi/180; %Rotate .obj data around x axis if necessary
    thetay=-2*pi/180; %Rotate .obj data around y axis if necessary
    aaa=-0.02; % Translate .obj Data
    eee=0;
    bbb=0.1; %title left vertical adjustment
    ccc=0.098; %title right vertical adjustment
end

Mz=[cos(thetaz) -sin(thetaz) 0; sin(thetaz) cos(thetaz) 0; 0 0 1];
for i=1:GridSize
    for j=1:GridSize
        G=[Xcomp(i,j); Ycomp(i,j); Zcomp(i,j)];
        GG=Mz*G;
        Xcomp(i,j)=GG(1);
        Ycomp(i,j)=GG(2);
        Zcomp(i,j)=GG(3);
    end
end

Mx=[1 0 0; 0 cos(thetax) -sin(thetax); 0 sin(thetax) cos(thetax)];
for i=1:GridSize
    for j=1:GridSize
        G=[Xcomp(i,j); Ycomp(i,j); Zcomp(i,j)];
        GG=Mx*G;
        Xcomp(i,j)=GG(1);
        Ycomp(i,j)=GG(2);
        Zcomp(i,j)=GG(3);
    end
end

My=[cos(thetay) 0 sin(thetay); 0 1 0; -sin(thetay) 0 cos(thetay)];
for i=1:GridSize
    for j=1:GridSize
        G=[Xcomp(i,j); Ycomp(i,j); Zcomp(i,j)];
        GG=My*G;
        Xcomp(i,j)=GG(1);
        Ycomp(i,j)=GG(2);
        Zcomp(i,j)=GG(3);
    end
end

%% Translate .obj Data
Xcomp=Xcomp;
Ycomp=Ycomp;
Zcomp=-Zcomp-aaa;

if TransformGrid==1
    surf(Xfig,Yfig,Zfig,'FaceColor', 'interp', 'EdgeColor', 'none')
    hold on
    surf(Xcomp,Ycomp,Zcomp,'FaceColor', 'interp', 'EdgeColor', 'none')
end
```matlab
% plot3(Xcomp,Ycomp,Zcomp,'.') %if surface is noisy, use this code
colormap(cool) %other: colormap(gray)
camlight right
axis equal
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
break
end

%% Interpolate surface from .obj Data
Figlength=length(Xfig(1,:));
iNew=1;
fFinal=Figlength;
for i=iNew:fFinal
    for j=iNew:fFinal
        compareX=abs(Xcomp-Xfig(i,j));
        [C,Ix] = min(compareX);
        [E,Jx] = min(C);
        minix=Ix(Jx);
        minjx=Jx;
        NSX(i-iNew+1,j-iNew+1)=Xcomp(minix,minjx);

        compareY=abs(Ycomp-Yfig(i,j));
        [C,Iy] = min(compareY);
        [E,Jy] = min(C);
        miniy=Iy(Jy);
        minjy=Jy;
        NSY(i-iNew+1,j-iNew+1)=Ycomp(miniy,minjy);
    end
end

NSZ = griddata(Xcomp,Ycomp,Zcomp,NSX,NSY)+eee;

% correct exterior of .obj scan so that edge does not exceed 1 radius
for i=1:Figlength
    for j=1:Figlength
        vectormag=norm([NSX(i,j) NSY(i,j) NSZ(i,j)]);
        if abs(vectormag)>r
            vectordist=[NSX(i,j) NSY(i,j) NSZ(i,j)];
            DirectionV=vectordist/vectormag;
            NSX(i,j)=r*DirectionV(1); %New Surface data X dir
            NSY(i,j)=r*DirectionV(2); %New Surface data Y dir
            NSZ(i,j)=r*DirectionV(3); %New Surface data Z dir
        end
    end
end

%% Finding Percent Variation of middle ComparePercent value
ComparePercent=80; %percent of membrane to evaluate
%getting the area of the obj
GridSize=length(NSX(1,:));
```

[NodeArea] = FindNodeArea(NSX,NSY,NSZ,GridSize);
NSZ2 = griddata(Xfig,Yfig,Zfig,NSX,NSY); %normalizing comparison
difx=NSX-NSX;% (NSX-Xfig) comparison;
dify=NSY-NSY;% (NSY-Yfig) comparison;
difz=NSZ-NSZ2;% (NSZ-Zfig) comparison;

% surf(NSX,NSY,difz)
for i = 1:GridSize
    for j = 1:GridSize
        vectormag=norm([NSX(i,j) NSY(i,j) NSZ(i,j)]);
        if abs(vectormag)<r*ComparePercent/100
            DiffNode(i,j)=norm([difx(i,j) dify(i,j) difz(i,j)]);
            DiffVol(i,j)=DiffNode(i,j)*NodeArea(i,j);
            if isnan(DiffVol(i,j))
                DiffVol(i,j)=1
            end
            DiffVol(i,j)=0;
        end
    end
end
TotalVolDiff=sum(sum(abs(DiffVol)));
SphereVolume=HalfSphere*2;
VolumePercentError=100*TotalVolDiff/SphereVolume;

% Plotting Data
LF=length(Zfig(1,:));
LFH=round(LF/2);
LF2=length(NSZ(1,:));
LFH2=round(LF2/2);

close all
hold off
Zfactor=2;
subplot(2,2,1)
surf(Xfig,Yfig,Zfig, 'FaceColor', 'interp', 'EdgeColor', 'none')
colormap(cool)%other: colormap(gray)
camlight right
axis equal
axis off
zoom(Zfactor)
view(45,30)
Tt1=sprintf('Simulation Topology %g%% Fill Volume',Percent);
Th1=title(Tt1);
Ph1 = get(Th1,'Position');
set(Th1,'Position',[Ph1(1) Ph1(2) Ph1(3)-bbb]);

subplot(2,2,2)
surf(NSX,NSY,NSZ,'FaceColor', 'interp', 'EdgeColor', 'none')
colormap(cool)%other: colormap(gray)
camlight right
axis equal
axis off
zoom(Zfactor*1.1)
if Percent==100
    zoom(Zfactor*0.35)
end
view(45,30)
Tt2=sprintf('Experiment Topology %g\% Fill Volume',Percent);
Th2=title(Tt2);
Ph2 = get(Th2,'Position');
set(Th2,'Position',[Ph2(1) Ph2(2) Ph2(3)-ccc]);

subplot(2,2,3:4)
h3=plot(Yfig(:,LFH)*39.3701,Zfig(:,LFH)*39.3701,'-.k','LineWidth',2);
hold all
h4=plot(NSY(:,LFH2)*39.3701,NSZ(:,LFH2)*39.3701,'k','LineWidth',2);
axis equal
legend([h3 h4],['Simulation','Experiment'],'Location','South');
ylabel('Height (Inches)')
xlabel('Radius (Inches)')

TitleString=...
    sprintf('Cross Sections at %g\% Fill Volume, Error=%g\%','...
    Percent,VolumePercentError);
title(TitleString)
xlim([-10 10])

if Percent==100
    ylim([-1.7 6.5])
end

set(gcf,'PaperUnits','inches','PaperPosition',[0 0 6 5])
PrintTitle=sprintf('%sCompare%gPercentVolume',Outpath,Percent);
print('-dpng',PrintTitle)

figure
surf(Xfig,Yfig,Zfig, 'FaceColor', 'interp', 'EdgeColor', 'none')
colormap(cool)%other: colormap(gray)
camlight right
axis equal
view(45,30)
Tt1=sprintf('Simulation Topology at %g\% Fill Volume',Percent);
Th1=title(Tt1);
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')

set(gcf,'PaperUnits','inches','PaperPosition',[0 0 6 5])
PrintTitle=sprintf('%sSimulated%gPercentVolume',Outpath,Percent);
prient('-dpng',PrintTitle)

figure
surf(NSX,NSY,NSZ,'FaceColor', 'interp', 'EdgeColor', 'none')
colormap(cool)%other: colormap(gray)
camlight right
axis equal
view(45,30)
Tt2=sprintf('3D Scan of Diaphragm at %g\% Fill Volume',Percent);
Th2=title(Tt2);
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')

set(gcf, 'PaperUnits', 'inches', 'PaperPosition', [0 0 6 5])
PrintTitle = sprintf('%sExperimental%gPercentVolume', Outpath, Percent);
print('-dpng', PrintTitle)
APPENDIX K: MATLAB MODEL - SIMULATED DIAPHRAGM

INVERSION TOPOLOGY RESULTS

Fig. K.1 Soft body deformation simulation of diaphragm inverting at 0% Fill Volume

Fig. K.2 Soft body deformation simulation of diaphragm inverting at 10% Fill Volume
Fig. K.3 Soft body deformation simulation of diaphragm inverting at 20% Fill Volume

Fig. K.4 Soft body deformation simulation of diaphragm inverting at 30% Fill Volume
Fig. K.5 Soft body deformation simulation of diaphragm inverting at 40% Fill Volume

Fig. K.6 Soft body deformation simulation of diaphragm inverting at 50% Fill Volume
Fig. K.7 Soft body deformation simulation of diaphragm inverting at 60% Fill Volume

Fig. K.8 Soft body deformation simulation of diaphragm inverting at 70% Fill Volume
Fig. K.9 Soft body deformation simulation of diaphragm inverting at 80% Fill Volume

Fig. K.10 Soft body deformation simulation of diaphragm inverting at 90% Fill Volume
Fig. K.11 Soft body deformation simulation of diaphragm inverting at 100% Fill Volume
Fig. L.1 Side view of slosh diaphragm 16.5 inch diameter fuel tank, A) tank, B) gas port, C) liquid port, D) 80-20 aluminum frame for center of gravity testing, E) LVDT displacement sensors with spring mounts
Fig. L.2 Front view of slosh diaphragm 16.5 inch diameter fuel tank, A) tank, B) gas port, C) liquid port, D) 80-20 aluminum frame for center of gravity testing, E) LVDT displacement sensors with spring mounts
Fig. L.3 Xbox 360 Kinect infrared 3D sensor with an electric motor mounted to the sensor

Fig. L.4 Sample photo of slosh diaphragm 16.5 inch diameter tank at 70% fill volume during the 3D scanning experiment fuel tank, A) gas port, B) cinder block support for Xbox Kinect 3D sensors, C) tank, D) diaphragm, E) 80-20 aluminum support frame
Fig. L.5 Photo of the entire 3D scanning experiment setup, A) 6 Xbox Kinects, B) 3D scan processing workstation, C) cinderblocks sensor mounts, D) propellant tank sitting on 80-20 aluminum support frame