The Impact of System Parameters on a Mock Aorta in an Ex Vivo Heart Perfusion Pump Loop

by

Sining Li

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
University of Alberta

Abstract

Ex Vivo Heart Perfusion (EVHP) preserves a donor heart's natural beating function outside of the body. The purpose of this work was to study the influence of mock aorta properties on pump performance in a mechanical EVHP system. Previous work has shown potential benefits on pump workload by including the compliant nature of the aorta at the pulsatile pump exit. The EVHP system can be assumed analogous to a mechanical reciprocating pump loop.

An experimental test loop was developed to model the EVHP cardiac systemic circulation flow loop for which all system parameters could be defined or measured. The developed mechanical test loop was modular and could be expanded to include more components as needed in future studies. The physiological pulsatile flow is generated using a programmable diaphragm pump. At the pump outlet, a mock aorta section simulates aortic elastic response. To predict system parameters, the flow loop was modeled using a Bernoulli analysis. Silicone mock aortas of variable length and compliance were investigated to understand the flow effects of the compliant section. The silicones tested are two-part silicones from Smooth-On Inc, Ecoflex and Dragon Skin. To test the effect of compliance, a design strategy was developed to cast the compliant aorta. This design strategy can be applied to other compliant components as EVHP testing requires.

A parametric study of system parameters on pump performance with the effect of aortic compliance was conducted. System backpressure, aorta chamber pressure, pump flow rate, pump stroke volume, aorta length, and mock aorta material were varied. Fluid pressure was measured at the pump inlet, pump outlet, aorta inlet, and aorta outlet. Aorta distension was captured using a high-speed camera. Pump energy was calculated using measured fluid pressure. The energy decrease at the aorta outlet from the aorta inlet was found. Maximum aorta percent distension is found through processing high-speed camera images using custom edge detection code. Pump energy, energy decrease, and aorta distension are compared against system parameters to determine overall effect on pump performance.

In general, for the Dragon Skin aorta, increasing backpressure led to an increase in pump workload. Increasing chamber-pressure, which led to a corresponding increase in aortic distension, led to an increase in pump workload. Increasing aorta length led to reduced pump workload. Pulse frequency and stroke volume are considered as one parameter together due to their connected nature during pump operation. Increasing pulse frequency/decreasing stroke volume at first led to an increase in pump workload up to approximately 80-100 bpm/down to approximately 66.5-72.5 mL/beat. From 100 bpm upwards, or 66.5 mL/beat downwards, pump workload decreased. The same trends were reflected in the scaled change in tube distention, $\left(\frac{\Delta D}{D}\right)$, the maximum aorta percent distension results. Backpressure effects on $\frac{\Delta D}{D}$ are inconclusive due to insufficient and conflicting data. There appeared to be an inverse relationship between E and ΔE, where E is the pump energy, and ΔE is the energy decrease from aorta inlet to aorta outlet. Parameters which increase E will decrease ΔE. This agrees with the relationships found for $\frac{\Delta D}{D}$ when plotted against E and ΔE. Parameters which increase tube distension will increase pump workload and decrease ΔE. The effects of tube distension on system energy are more pronounced at lower backpressure and lower pulse frequency/higher stroke volume conditions. Due to the limiting size of the pressure chamber containing the aorta, many Ecoflex aorta data cases were removed. This led to insufficient data points for a full analysis of Ecoflex aorta samples.

Overall, a low system backpressure with low aorta distension due to low chamber-pressure and long aorta length will have a positive impact on pump performance. This applies for all pulse frequency/stroke volume conditions. Aortic parameters which lead to a low distension, such as high aorta length and low chamber pressure, result in both a reduced pump workload and a greater decrease in pump energy downstream of the aorta compliance section. These positive effects are greater when pump operating conditions are at the lower end of the pulse frequency, and correspondingly the higher end of stroke volume. Lowering the backpressure conditions acting on the pump will increase the positive effects of compliance parameters.

Preface

Some of the research conducted for this thesis forms part of a larger research project, led by Dr. Darren H. Freed at the University of Alberta, with Dr. David S. Nobes being the lead collaborator in the Department of Mechanical Engineering. This research follows previous investigations into compliant tubes done by Katie Cameron. The experimental setup described in Chapter 2 was a collaborative design done by a team including myself, Karan Billing, Jake Hadfield, Calynn Stumpf, Isobel Tetreau, and Tod Vandenberg. The literature review in Chapter 1, data processing in Chapter 3, results discussion in Chapter 4, and conclusion in Chapter 5 are my original work.

Acknowledgements

First and foremost, I would like to give a huge thank you to my supervisor Dr. David S. Nobes for his monumental support throughout this journey. He answered the challenges and difficulties I faced at every step with patience and guidance. His approach to viewing his job as a position to help us students succeed is a profound lesson in leadership. Being a part of his research group, I have not only developed engineering skills, but more importantly, the skills necessary to be a team member. I am incredibly grateful to have had the chance to be a part of Dr. Nobes' research group under his supervision.

I would like to thank Dr. Darren Freed, for providing the opportunity to work on this project. Jumping into the world of biomedical engineering was both challenging yet rewarding. I was outside of my comfort zone and benefited from the experience both educationally and personally.

A big thank you goes to all the team members of the heart project who helped to build this experimental setup. Isobel Tetreau and Tod Vandenburg spent an incredibly hardworking co-op term building components and creating solid models for this project. Karan Billing was the project manager this project needed to organize itself and move forward. Jake Hadfield was always there to help, from coding to theory and everything in-between. The setup could not have run so smoothly without the help of Calynn Stumpf, whose manufacturing knowledge was integral to its success.

Last but not least, I would like to thank my mother, my father, and my talented little sister. Their unyielding support gave me the strength to keep going through the toughest of times. Thank you for supporting my dreams, wherever they take me.

Table of Contents

Abstract ii

Preface v

Acknowledgements vi

Table of Contents vii

List of Tables xi

List of Figures xii

List of Symbols xix

Chapter 1. Literature Review 1

1.1. Introduction and Motivation 1

1.2. Ex Vivo Heart Perfusion (EVHP) 3

1.3. Experimental Aorta Phantoms 5

1.4. The Cardiac Cycle 7

1.5. Pulsatile Flow 10

1.6. Experimental Pulsatile Systems 12

1.7. Newtonian vs Non-Newtonian Fluid 15

1.8. Research Objectives 17

1.9. Content and Outline of Thesis 18

Chapter 2. Experimental Setup 19

2.1. Introduction 19

2.2. The Pulsatile Pump 22

2.2.1. Pulsatile Pump/System Development 23

2.2.2. The Selected Diaphragm Pump 26

2.3. Pump Drive Mechanism 28

2.4. Compliant Components 30

2.4.1. General Molding Procedure 31

2.4.2. Outlet Valve 33

2.4.3. The Compliant Tube 40

2.5. Pressure-Chamber 44

2.6. Reservoir and Backpressure Pump 54

2.7. Data Acquisition 55

2.7.1. Timing 57

2.8. Limitations 59

2.8.1. System Vibrations 59

2.8.2. Pressure Transducer Measurement Range 59

2.8.3. Chamber-Pressure Fluctuations 59

2.8.4. Aorta Outlet pressure transducer location 60

2.8.5. Pulsatile Pump Pressure Waveform 61

2.9. Summary 63

Chapter 3. Data Processing Methodology 64
3.1. Introduction 64
3.2. Processing of Imaging Data 64
3.2.1. Aorta Percent Diameter Distension, $\boldsymbol{\Delta DD}$ 67
3.3. Processing of Pressure Data 70
3.3.1. Pressure Unit Conversions 77
3.4. Processing of Pump Stroke Distance 80
3.5. Volumetric Flow Rate, $\boldsymbol{Q(t)}$ 82
3.6. Pump Energy Calculation 89
3.7. Limitations 91
3.8. Summary 92
Chapter 4. Impact of Varying System Parameters on Pump Performance 93
4.1. Introduction 93
4.2. Impact of Experimental Parameters on Pump Energy 94
4.2.1. Pump Energy at Aorta Inlet and Aorta Outlet, E 94
4.2.2. Percent Energy Decrease from Aorta Inlet to Aorta Outlet, ΔE 102
4.2.3. Discussion 113
4.3. Impact of Aorta Properties on Aorta Elastic Response 115
4.3.1. Discussion 127
4.4. Impact of Aorta Elastic Response on Pump Energy 128

4.4.1. Pump Energy at Aorta Inlet and Aorta Outlet, *E* 128

4.4.2. Percent Energy Decrease from Aorta Inlet to Aorta Outlet, *ΔE* 130

4.4.3. Discussion 133

4.5. Limitations 134

4.5.1. Impact of Discarding Specific Experimental Cases 134

4.5.2. Tube Rest State 135

4.5.3. Pulse Frequency and Stroke Volume 135

4.5.4. 60 mm Tube Experiments 135

4.6. Summary 136

4.6.1. Impact of Experimental Parameters on Pump Energy per Cycle, E and *ΔE* 137

4.6.2. Impact of Aorta Properties on Elastic Response 137

4.6.3. Impact of Aorta Elastic Response on Pump Energy 138

4.6.4. Overall Effect on Pump Performance 139

Chapter 5. Conclusion and Future Work 140

5.1. Conclusion 140

5.2. Limitations and Future Work 142

References 143

Appendices 154

List of Tables

Table 2.1: Pulsatile pump design specifications 23

Table 2.2: Pump stroke volume calibration data 26

Table 2.3: Specifications of the lead screw traverse used to transform the rotary motion of the control motor into linear motion 29

Table 2.4: Silicone specifications 31

Table 2.5: The specifications of the compliant tube 41

Table 3.1: Chamber-pressure unit conversions 77

Table 3.2: Backpressure unit conversions 78

Table 3.3: $Qcycle$, maximum stroke volume per cycle at each bpm 82

Table 4.1: Terms and symbols used in results discussion 94

Table 4.2: Number of experimental cases for Ecoflex and Dragon Skin aortas, separated by backpressure 95

Table 4.3: Impact of experimental parameters on energy per cycle and percent energy decrease at aorta outlet from aorta inlet 136

Table 5.1: Limitations and future work 142

List of Figures

Figure 1.1: Labeled photo of test EVHP pump loop in use in Dr. Darren Freed's lab at the University of Alberta 3

Figure 1.2: Labelled plots of three Murgo's Type waveforms, (a) Type A, (b) Type B, and Type C (c). Adapted from [25] for Types A and C, and [26] for Type B 8

Figure 1.3: Plot of viscosity vs shear rate for whole blood, adapted from [64] in red, with the blue plot representing the fitted Ostwald-de Waele power-law model. 16

Figure 2.1: Schematic of experimental set-up. PT stands for Pressure Transducer 21

Figure 2.2: (a) EVHP compared to (b) image of the experimental set-up 22

Figure 2.3: Diaphragm pump design progression from (a) fully 3D printed model, to (b) modified bilge pump, to (c) small air operated diaphragm pump to (d) final design using a larger air operated diaphragm pump 24

Figure 2.4: Pump modifications for (a) front face with diaphragm and drive shaft connection and (b) back face with outlet valve housing and inlet T-connector 26

Figure 2.5: Pump manufacturer's performance curve with necessary experimental range boxed in red; adapted from [66]. The y-axis represent pressure for both air and water. The x-axis represents pump discharge flow rate. The blue curves represent operating air pressure plots, and the orange curves represent the flow rate of air required. 27

Figure 2.6: Labelled image of pump drive mechanism 28

Figure 2.7: Plot of voltage data corresponding to motor rotation showing (a) sine vs (b) heart waveform generated by the motor 30

Figure 2.8: labelled image of the silicone injection process. Silicone flow direction is labelled using blue arrows and molding components labelled using red arrows. 33

Figure 2.9: Images of (a) molded silicone trileaflet valve and (b) anotated solid model of silicone valve 34

Figure 2.10: Dimensioned engineering drawing views of Dragon Skin valve 36

Figure 2.11: Annotated image of 3D printed trileaflet valve silicone positive (left) and negative (right) mold. 37

Figure 2.12: Engineering drawing of trileaflet valve mold assembly exploded view 38

Figure 2.13: Dimensioned engineering drawing of trileaflet valve mold, where (a) is the positive mold (base) and (b) is the negative mold (top) 39

Figure 2.14: Images of (a) compliant tube mold during the curing process and (b) the raw demolded Dragon Skin compliant tube 40

Figure 2.15: Aorta mold assembly isometric view (left) and exploded (right) view with bill of materials included 42

Figure 2.16: Dimensioned orthographic view of aorta mold assembly. Full drawing package of aorta mold parts included in Appendix A. 43

Figure 2.17: Pressure-chamber (a) labeled image and (b) 44

Figure 2.18: Pressure-chamber isometric view (left) and exploded (right) view with bill of materials 45

Figure 2.19: Dimensioned engineering orthographic drawing of (a) pressure chamber assembly and (b) acrylic chamber 46

Figure 2.20: Annotated isometric view of square flange cap 47

Figure 2.21: Dimensioned orthogonal engineering drawing views of square flange cap 48

Figure 2.22: Annotated isometric view of valve casing solid model 50

Figure 2.23: Dimensioned engineering orthographic drawing of pressure chamber assembly 51

Figure 2.24: Annotated views of pump connection plate solid model 52

Figure 2.25: Dimensioned orthogonal engineering drawing views of pump connection plate 53

Figure 2.26: Schematic of data acquisition set-up 56

Figure 2.27: Plot of voltage data from DAQ for the complete data set 57

Figure 2.28: Plot of voltage data from DAQ zoomed in to show camera trigger function 58

Figure 2.29: Pressure-chamber pressure variations throughout pump cycles for internal chamber pressure of 0 (red) and 6.9 (blue) kPa 60

Figure 2.30: Labelled plots of three Murgo's Type waveforms, (a) Type A, (b) Type B, and Type C (c). (d) is a labelled plot of the experimental pressure waveform, with the full experimental range show in (e). Adapted from [25] for Types A and C, and [26] for Type B 62

Figure 3.1: Image processing steps starting from (a) initial image, (b) binarized image, (c) edge detection to find regions to crop, and (d) cropped image 65

Figure 3.2: Plot of peak intensity locations along one tube vertical pixel axis for one image 66

Figure 3.3: Plots of tube distension over time shown as 3D mesh plots of (a) full experimental time and (b) zoomed in to show 2 cycles 68

Figure 3.4: Plots of tube distension over time at tube mid length of (a) full experimental time and (b) zoomed in to show 2 cycles 69

Figure 3.5: Pressure transducer calibration device with pressure transducer attached 70

Figure 3.6: Pressure transducer calibration data for transducer located at (a) pump inlet, (b) pump outlet, (c) aorta inlet, (d) aorta outlet 72

Figure 3.7: Plot of raw vis filtered pressure at the aorta inlet over (a) the full experimental time range, and (b) two cycles 74

Figure 3.8: Plot of filtered pressure at all flow loop locations over (a) the full experimental time range and (b) zoomed in for two pump cycles 76

Figure 3.9: Plot of fluid pressure in both kPa on the left y-axis and mmHg on the right y-axis vs centrifugal pump speed in rpm. A polynomial fit for the conversion is found and plotted in blue. Error bars are show on the plot at two standard deviations from the mean. 79

Figure 3.10: Plot of LVDT voltage and displacement calibration 80

Figure 3.11: Plot of raw vs filtered pump stroke distance against time for three pump cycles 81

Figure 3.12: Centerline velocity as a function of time for three pump cycles at the aorta outlet . 83

Figure 3.13: Approximating pump stroke travel distance using two polynomial expressions 84

Figure 3.14: Volumetric flow rate and centerline velocity as a function of time for three pump cycles at the aorta outlet, using a polynomial expression to approximate the LVDT travel distance 85

Figure 3.15: Pressure plot comparison at the pump inlet and aorta outlet 87

Figure 3.16: Adjusted volumetric flow rate and centerline velocity as a function of time for three pump cycles at the aorta outlet with the pump suction stroke values set to zero 88

Figure 3.17: Plot of all energy types for one experimental case 90

Figure 4.1: Plots of pump energy per cycle at aorta outlet vs backpressure. Chamber-pressure 0-6.9 kPa 96

Figure 4.2: Plots of pump energy per cycle at aorta outlet vs backpressure. Chamber-pressure 10.3-17.2 kPa 97

Figure 4.3: Plot of energy per pump cycle vs pulse frequency for both 0 kPa backpressure, 3.4 kPa chamber-pressure, and 15.66 kPa backpressure, 24.1 kPa chamber-pressure; at both aorta inlet and aorta outlet. Dragon Skin tube material. 98

Figure 4.4: Plot of energy per pump cycle vs aorta length for both 0 kPa backpressure, 3.4 kPa chamber-pressure, and 15.66 kPa backpressure, 24.1 kPa chamber-pressure; at both aorta inlet and aorta outlet. Dragon Skin tube material. 99

Figure 4.5: Plot of energy per pump cycle vs chamber-pressure for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Location: aorta outlet. Material: Dragon Skin. 101

Figure 4.6: Plots of *ΔE,* percent energy decrease at the aorta outlet from the aorta inlet vs backpressure. Chamber-pressure 0-6.9 kPa. 103

Figure 4.7: Plots of *ΔE,* percent energy decrease at the aorta outlet from the aorta inlet vs backpressure. Chamber-pressure 10.3-17.2 kPa. 104

Figure 4.8: Plot of *ΔE* percent energy decrease at the aorta outlet from the aorta inlet vs chamber-pressure, separated by aorta length, for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Material: Dragon Skin. 106

Figure 4.9: Plot of percent energy decrease between aorta outlet and inlet vs (a) pulse frequency and (b) stroke volume per cycle, separated by aorta length, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. 108

Figure 4.10: Plots of percent energy decrease between aorta outlet and inlet vs pulse frequency. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa] *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140. 110

Figure 4.11: Plots of percent energy decrease between aorta outlet and inlet vs pulse frequency. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa] *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140. 111

Figure 4.12: Plots of percent energy decrease between aorta outlet and inlet vs pulse frequency. b: back-pressure [kPa]; c: chamber-pressure [kPa] m: material; e: Ecoflex; Aorta lengths [mm] of red: 60; blue: 100; green: 140. 112

Figure 4.13: Plots of percent energy decrease between aorta outlet and inlet vs aorta length, separated by pulse frequency/stroke volume, for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. 113

Figure 4.14: Cropped image used in image processing. Maximum distension from region x = 300-x = 500, highlighted in red, is averaged. 115

Figure 4.15: Maximum percent diameter distension, ΔDD, vs backpressure. Chamber-0-6.9 kPa. 117

Figure 4.16: Maximum percent diameter distension, ΔDD, vs backpressure. Chamber-pressure 10.3-17.2 kPa. 118

Figure 4.17: Plot of percent diameter distension vs chamber-pressure for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Material: Dragon Skin. 120

Figure 4.18: Plot of max and min tube diameter vs chamber-pressure for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Material: Dragon Skin. 121

Figure 4.19: Plot of maximum percent diameter distension vs aorta length separated by stroke volume, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. 122

Figure 4.20: Plot of maximum percent diameter distension vs aorta length, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; d: Dragon Skin; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4................................ 124

Figure 4.21: Plot of maximum percent diameter distension vs aorta length, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; e: Ecoflex; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4................................ 125

Figure 4.22: Plot of maximum percent diameter distension vs (a) stroke volume and (b) pulse frequency, separated by aorta length, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. 126

Figure 4.23: Plot of pump energy per cycle vs maximum percent diameter distension at (a) aorta inlet and (b) aorta outlet, separated by stroke volume and its corresponding pulse frequency. . 128

Figure 4.24: Plot percent energy change decrease between aorta outlet and inlet vs maximum percent diameter distension, separated by stroke volume and its corresponding pulse frequency, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. 130

Figure 4.25: Plot percent energy change decrease between aorta outlet and inlet vs maximum percent diameter distension, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; d: Dragon Skin; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4.. 132

Figure 4.26: Plot percent energy change decrease between aorta outlet and inlet vs maximum percent diameter distension, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; e: Ecoflex; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4.. 133

List of Symbols

A_n	Cross sectional area at location *n*
b	Backpressure [rpm] applied to the system using the centrifugal backpressure pump
bpm	Beats per minute [bpm], describes pump pulse frequency
c	Pressure chamber internal chamber-pressure [kPa]
D	Tube diameter [m] or [px]
$\Delta D/D$	Diameter distension in percent
$D(t)$	Aorta diameter at camera time *t*
D_{min}	Minimum aorta diameter, averaged over 8 cycles.
$E_{aorta\ inlet}$	Energy per cycle [J/cycle] at the aorta inlet
$E_{aorta\ outlet}$	Energy per cycle [J/cycle] at the aorta outlet
$E_{kinetic}$	Kinetic energy [J/cycle]
$E_{local\ accel}$	Local acceleration energy [J/cycle]
$E_{pressure}$	Pressure pulse energy [J/cycle]
E_{pump}	Total pump energy per pump pulse cycle [J/cycle]
E_{static}	Static energy [J/cycle]
f	Pulse frequency, or heart rate, expressed in beats per minute [bpm], see also: bpm
g	Acceleration of gravity [m/s^2]
k	Flow consistency [Pa.s^n]
L_{eff}	Length of accelerated fluid column.
L_{effn}	Height difference between location *n* and preload reservoir
m	Aorta material, Dragon Skin and Ecoflex silicones
n	Flow index or specified parameters at location n
P_{losses}	Frictional loss along tube
P_n	Fluid pressure at point *n* [Pa]

$\Delta P(t)$	Change in pressure gradient with respect to time between two points in the system
$Q(t)$	Volumetric flow rate [m^3/s]
Q_{cycle}	Maximum volumetric flow per cycle [m^3/cycle]
SV	Stroke volume per beat of heart pump
t_c	Cycle time [s]
u_n	Flow speed at point *n*.
$\left(\frac{dv}{dt}\right)_n$	Acceleration at location *n* [m/s2]
$\bar{v}_n$	Average centerline velocity at location *n* [m/s]
x	Pump displacement [m]
x_{cycle}	Maximum pump displacement per cycle [m]
z_n	Elevation at point *n*
ΔZ_n	Height difference between preload reservoir and location *n* [m]
α	Womersley number
ρ	Fluid density [kg/m^3]
ω	Pulse frequency [rad/s]
μ	Fluid dynamic viscosity [Pa.s].
γ	Shear rate [s^{-1}]

Chapter 1. Literature Review

1.1. Introduction and Motivation

In the United States and Europe, 10-12% of heart transplant waitlist patients die before their turn [1]. This is due to donor heart demand far exceeding supply [1]. There are three main reasons for this shortage. The first is the high discard rate of donor hearts [2], where only 36-39% of donor hearts are transplanted with success [3], [4]. The second reason for the shortage is the brief 6 hour time window for transplantation [5]. The third is the utilization of only Donation after Brain Death (DBD) donor hearts and exclusion of Donation after Circulatory Death (DCD) donor hearts [6].

A factor contributing to the high discard rate of donor hearts is the localized cell damage that occurs during the brain death process for DBD hearts. The current heart transplantation procedure does not incorporate rehabilitation procedures for hearts where the majority of cardiac issue is healthy, and localized cell injuries can be repaired [7].

The brief 6 hour time window for transplantation is imposed by the limitations of the current method of heart storage: hypothermic static storage [8], [9]. The temperature of the donor heart is lowered to reduce metabolic demands and extend its life span. However, this method does not provide live feedback of heart conditions, since the donor heart is not beating in this state. Many indicators of heart health, such as cardiac rhythm or blood pressure, require a beating heart to be measured via electrocardiography or hemodynamic monitoring.

DCD donors are classified using circulatory death criteria [10]. Rapid tissue damage occurs during warm ischemic time, which is longer for DCD donors than DBD donors [10], [11]. Warm ischaemia is the time during which the heart stops beating and suffers asphyxiation due to lack of blood supply [11].

Ex Vivo Heart Perfusion (EVHP) is a method proposed to overcome the aforementioned reasons for donor heart shortage [7], [12], [13]. EVHP preserves a donor heart’s natural beating function outside of the body via connection to a mechanical flow system. Since the heart is maintained in a live state, the transplantation window time can be increased and cell damage can be repaired in both DBD and DCD donors [12]. Heart performance can also be assessed, monitored, and controlled, allowing greater understanding of transplant acceptability.

Although the current state of the medical EVHP system is an improvement on donor heart function and increases potential for transplant viability, there are questions as to whether the donor heart is experiencing conditions similar to the physiological body. Currently, the tubing connections of the medical EVHP system uses standard off-the-shelf, rigid tubing. Blood flows from the heart into this stiff tube. Within the human body, at the exit of the heart is the aorta, the largest artery within the human body. The physiological aorta is compliant, which differs from the stiff tubing used at the heart outlet in the EVHP system.

1.2. Ex Vivo Heart Perfusion (EVHP)

Ex vivo heart perfusion (EVHP), connects a donor heart to a mechanical flow loop outside the human body and maintains its natural beating function [12]. Figure 1.1 shows a photo of the current test EVHP pump loop in use in Dr. Darren Freed's lab at the University of Alberta. This setup allows the heart to function closely to its natural beating rhythm. A pacemaker keeps the heart—labeled at the top of the image—beating, maintaining pulsatile blood flow to the rest of the flow loop. Below the heart, the components represent the basic functions of the human body, such as an oxygenator, backflow pump, reservoir, and waste disposal.

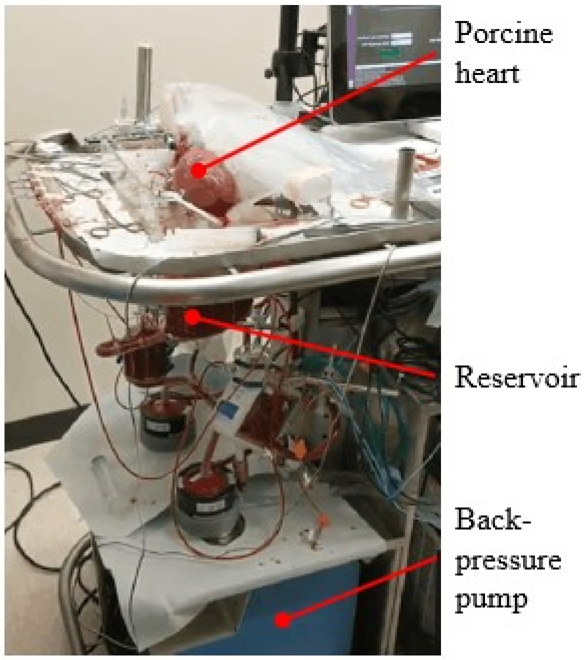


Figure 1.1: Labeled photo of test EVHP pump loop in use in Dr. Darren Freed's lab at the University of Alberta

There are two flow loops linked to the heart. The first is for pulmonary circulation, which delivers deoxygenated blood between the heart and lungs. The second is for systemic circulation, which delivers the newly oxygenated blood from the heart to the rest of the body and brings the deoxygenated blood back to the heart. The systemic circulation flow loop is the focus of this investigation as it encompasses the rest of the human body apart from the lungs. This flow loop starts from the left ventricle (LV), the main pumping chamber of the heart. Directly connected to the outlet of the LV is the aorta, the largest artery in the human body. The aorta compliance response due to the heart pressure pulse, the Windkessel effect [14], is one of the parameters studied in this investigation. The Windkessel effect describes the elastic extension and distension of the aorta in response to the pressure pulse generated by the heart. This elasticity allows the aorta to store a portion of the heart's stroke volume during its ejection phase and release this portion during the relaxation phase. This dynamic both creates a blood flow throughout the body that is nearly continuous, as well as reduces cardiac workload [14].

Previous research has shown that adding a compliant mock aorta in the EVHP system to simulate the Windkessel effect has a positive effect on pulsatile pump performance [15]. However, this effect was only tested using one mock aorta geometry and material. The effects of different mock aorta parameters, such as length and material, on a system with varying pump parameters such as pulse frequency and stroke volume have not yet been fully investigated prior to this investigation.

1.3. Experimental Aorta Phantoms

In order to study the effects of adding a compliant section downstream of a pulsatile heart pump, experimental aorta phantoms were required. Many studies investigating aortic flow patterns have been conducted to date utilizing a manufactured aorta phantom [16]–[22]. Silicone is a popular material to use due to its ability to reproduce arterial elastic response [23]. Many investigations manufacture aorta phantoms for the purpose of Particle Image Velocimetry (PIV) using a clear silicone, Sylgard 184 (Sylgard 184, Dow Corning) [17], [18], [20]. While these experiments show the flow effects within the aorta, the aortic compliance is not quantified or tested. Sylgard 184, although flexible, does not provide a degree of controllable compliance. Büsen et al. [16] developed an anatomically correct silicone aortic arch using Elastosil RT620 (Elastosil RT620, Wacker). The aortic compliance was controlled using an air chamber. The investigation done by Büsen et al. [16] focused on replicating aortic geometry and compliance for use in PIV investigations. To date, the effect of aortic compliance on pump performance have yet to be fully explored. Since the medical EVHP system replicates bodily functions, the heart outlet tubing attachment does not require branching and only needs one inlet and outlet. This greatly simplifies the complicated aortic geometry required in this investigation.

Previous research done by Cameron [15] utilized one molded silicone aorta tested under different system conditions. The aorta had a length of 137 mm, inner diameter of 12.7 mm, and a wall thickness of 4.1 mm. The silicone used was Ecoflex (Ecoflex ® 00-50. Smooth-On Inc.), a commercial silicone with a higher compliance response. A pressure-chamber controlled the extension and distension of the silicone tube. A limitation to the previous investigation is the testing of one aorta sample. The aorta sample had one length and material. In order to understand the broader scope of varying compliant parameters on pulsatile pump function, more aorta samples are required.

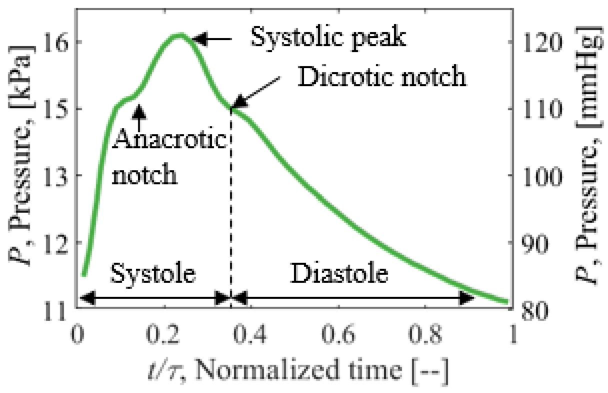
To further investigate the effect of a compliant section on a pulsatile loop, some limitations of the previous investigation must be addressed. The compliant section must be able to facilitate testing of different lengths and materials of aorta phantoms. For ease of keeping aorta samples consistent across the different lengths and materials, as well as to simplify attachment to the surrounding pressure-chamber and flow loop apparatus, a standard phantom shape is necessary. The physiological aorta is in a shape of a J-curve with complex branching to the rest of the body. However, with the simplification of the body in the medical EVHP system, only a straight tube is required at the outlet of the heart to the rest of the flow loop. Hence, for the purpose of this investigation, multiple aorta phantoms of different length and material in a standard tube shape are studied.

1.4. The Cardiac Cycle

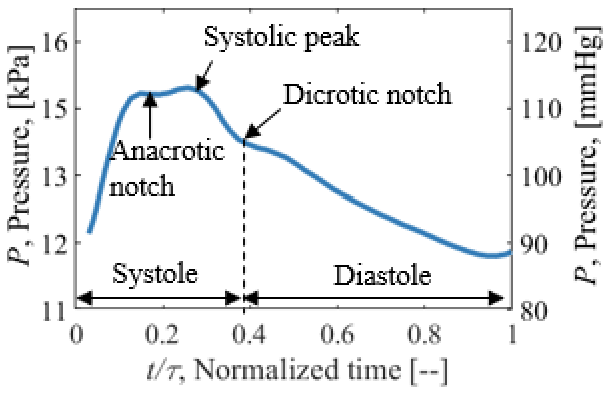
The cardiac cycle is non-sinusoidal [24]. Figure 1.2 shows diagrams of three types of central aortic pressure waveforms, Murgo's Type A, B, and C, in one heart cycle, where Types A and C are adapted from [25], and Type B is adapted from [26]. All three are central aortic pressure waveforms plotted against normalized time in one heart pulse cycle. Each plot is separated into systole (heart ejection phase) and diastole (heart relaxation phase), as labeled at the bottom of each plot. These pressure waveform classifications provide insight on overall cardiac health. The healthiest waveform is Type C [27] shown in Figure 1.2 (c), and will be used to describe the cardiac cycle.

The cardiac cycle is characterized in two stages, systole for heart contraction, and diastole for heart relaxation [24]. Figure 1.2 marks the timing of each stage at the bottom of each plot. Normalized time is the time t during a cardiac cycle divided by the cycle period τ. From Figure 1.2 (c), the beginning of the cycle is marked by the diastolic foot, representing the lowest point of pressure in the cycle before the start of systole [28]. The anacrotic limb represents the rapid increase in aortic pressure after the opening of the aortic valve and blood first flow into the aorta. Some ejected blood is stored in the compliant distension of the aorta. At the end of the anacrotic limb is the systolic peak, occurring at the completion of heart contraction. The aortic pressure drops and the aortic valve closes. At this point, the aortic walls recoil against the closed valve and causes a small rise in pressure. The point at the end of the initial pressure drop and beginning of the small rise is the dicrotic notch, marking the end of systole and beginning of diastole. After the small peak in pressure follows the dicrotic limb, during which blood fills the relaxing heart.

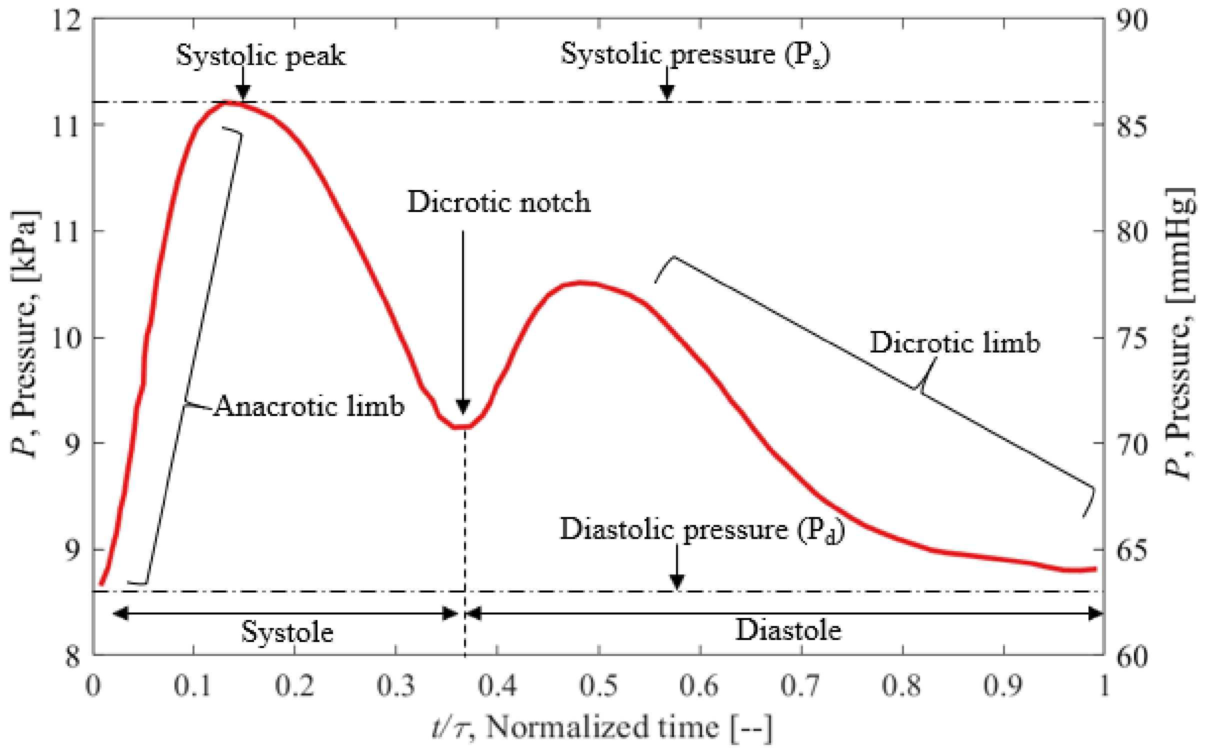
Murgo's Type A [25] and B [26] waveforms shown in Figure 1.2 subfigures (a) and (b) differ from Murgo's Type [25] C in the existence of an anacrotic notch before the systolic peak. The anacrotic notch is due to an early reflected wave during systole [27]. Type A waveform patients typically lie in the spectrum of older age patients, while Type B waveform patients lie in the younger age spectrum [27].



(a)



(b)



(c)

Figure 1.2: Labelled plots of three Murgo's Type waveforms, (a) Type A, (b) Type B, and Type C (c). Adapted from [25] for Types A and C, and [26] for Type B

Recreating the central aortic pressure waveform in a mechanical flow loop is an important design goal for experimental results to provide the best indicator of experimental parameter effects on the pulsatile pump loop. A consistent pulsatile pump pressure waveform with characteristics of a standard cardiac cycle is needed. Designing a pulsatile pump capable of outputting a pressure waveform similar to Murgo’s Type waveforms is ideal.

1.5. Pulsatile Flow

Work is done by the pulsatile pump on the fluid during the ejection phase of the pump cycle. The relationship between pump and fluid energy is an important assessment of pump performance. Equation 1.1 describes the non-dimensional Womersley number, α, used to quantify the relationship between the unsteady and viscous components in a pulsatile flow [29]:

$$\alpha = \frac{D}{2}\sqrt{\frac{\rho\omega}{\mu}} \qquad 1.1$$

where:

α	=	Womersley number
D	=	Tube diameter [m]
ρ	=	Fluid density [kg/m^3]
ω	=	Pulse frequency [rad/s]
μ	=	Fluid dynamic viscosity [Pa·s].

For low α flow as found in peripheral small arteries under steady, incompressible flow conditions, the conservation of energy can be represented using the Bernoulli equation [30]:

$$P_1 + \rho g z_1 + \frac{\rho u_1^2}{2} = P_2 + \rho g z_2 + \frac{\rho u_2^2}{2} \qquad 1.2$$

where:

P_n	=	Fluid pressure at point *n* [Pa]
g	=	Acceleration of gravity [m/s^2]
z_n	=	Elevation at point *n*
u_n	=	Flow speed at point *n*.

For high α flows as found in large arteries such as the human aorta, the conservation of energy require an extra term to represent the acceleration changes due to the unsteady component of the pressure gradient, described by a modified Bernoulli relationship as [15], [31]:

$$P_1 + \rho g z_1 + \frac{\rho {u_1}^2}{2} + \int_0^1 \frac{du_1}{dt} ds = P_2 + \rho g z_2 + \frac{\rho {u_2}^2}{2} + \int_0^2 \frac{du_2}{dt} ds \qquad 1.3$$

where:

$\int_0^n \frac{du_n}{dt} ds$	=	Fluid acceleration at point n integrated along a streamline ds.

Taking into account losses in the system, assuming uniform tube thickness and fluid acceleration varies with area in the same way as flow speed, the equation can be analytically evaluated and rearranged [15], [31] to become:

$$\Delta P(t) - P_{losses} = \rho L_{eff} \frac{du_2}{dt} + \frac{\rho}{2}({u_2}^2 - {u_1}^2) \qquad 1.4$$

where:

$\Delta P(t)$	=	Change in pressure gradient with respect to time between two points in the system
P_{losses}	=	Frictional loss along tube
L_{eff}	=	Length of accelerated fluid column.

Equation 1.4 is used in the calculation for pump energy by breaking it down into its pressure, kinetic, static, and local acceleration components. The work done by the pulsatile pump on the fluid can then be found for varying experimental parameters. This then provides a basis to compare the effects these different parameters have on overall pump performance.

1.6. Experimental Pulsatile Systems

There are commercially available pumps used for generating pulsatile waveforms for arterial flow study [32]–[36]. Pahlevan et al. [32], [37] utilized the ViVitro SuperPump (ViVitro Labs Inc., Victoria, BC) to study the passive pumping mechanisms within a compliant aorta. Other commercial pumps used in the study of arterial flow include the 1400 Series Harvard Apparatus (Harvard Apparatus, Holliston, MA) [38], [39], the CardioFlow 5000 MR (Shelley Medical Imaging Technologies, London, ON) [40], and the Compuflow 1000 piston pump (Shelley Medical, Toronto, ON) [36], [41]. Since these are commercially developed pumps purchased for laboratory use, their modifiability is severely limited for the purpose of this investigation.

There have been many pulsatile flow loops developed for laboratory studies [42]–[51]. Some developed in-house pump setups to generate pulsatile flow [42]–[47], while others developed mock circulatory loops to test artificial hearts and Ventricular Assist Devices (VADs) [48]–[51]. In general, the flow within the flow loop travels as follows. The fluid starts in the pulsatile pump. In-house designed pumps typically utilize a piston-cylinder setup controlled by a servomotor receiving computer input. The fluid exits the pump through a check-valve and into the outlet test section. The fluid then passes through a section design to replicate body conditions, such as a reservoir and backpressure pump, before arriving back at the pulsatile pump.

Previous investigations done by Cameron [15] utilized a commercial Ventricular Assist Device (VAD) to produce the pulsatile waveform of the heart within a mock EVHP flow loop. The VAD utilized in the investigation (Thoratec® Corporation) pumps fluid via expansion and contraction of a polyurethane pumping sac controlled via pneumatic application of positive and vacuum pressure [52]. Inlet and outlet flow control is regulated by two Bjork-Shirley tilting disc valves. A mock aortic test section is located downstream of the VAD outlet. The fluid passes by a backpressure pump and reservoir used to simulate human body conditions, before arriving back at the VAD inlet.

One limitation of the previous setup was the lack of control over the driveline pressure of the VAD [15]. Since VAD pulse frequency is dependent upon the driveline and arterial pressure difference [53], varying the pulse frequency was limited to controlling the backpressure using the afterload pump. In order to fully test systemic effects on varying pump conditions, a method to control pulse frequency and stroke volume independently of system backpressure is necessary.

Another limitation is the use of Bjork-Shirley tilting disc valves at the VAD outlet. Valve closure creates a sharp spike of noise in the pressure profile [15]. The geometry of the tilting disc valves are different from the human aortic valve, which is a tri-leaflet valve [54], [55]. The fluid mechanics of the tilting disc valve is different from a physiological valve [56], [57]. Due to the configuration of the VAD, the valves could not be replaced in the previous investigation. It is in the interest of this experimental setup to be modular for ease of component modification and replacement. A modular tri-leaflet valve at the outlet of the pulsatile pump will allow testing of system parameters on hearts with different valve shapes and conditions.

There are numerous medical prosthetic valve designs with different geometries [57]. These are separated into two main categories: mechanical and bioprosthetic. Current technology on mechanical heart valves are based on St Jude Medical's (St Jude Medical Inc., MN, USA) bileaflet mechanical valve design [58]. Two leaflets made of pyrolytic carbon, chosen for their biocompatible and lasting endurance [59], pivot about a hinge for valve opening and closure. The issue with mechanical heart valves is the geometry and material of the two leaflets, differing drastically from the soft aortic tissue of a physiological tri-leaflet heart valve. This causes the flow field at the valve outlet to differ from physiological flow. Bioprosthetic valves are made from animal tissue [57] and mimic the geometry of human tri-leaflet valves. These valves are more appealing for use in this investigation for their physiological fluid mechanics. However, since the purpose of this investigation is to test different system parameters, extreme pressure ranges not seen within a physiological system may be reached and can damage the valves. A major limitation of bioprosthetic heart valves is leaflet damage due to tearing [57]. All prosthetic valves, especially bioprosthetic ones, are expensive. It is not ideal to constantly replace damaged valves with such high costs associated. Hence, a manufacturing process was developed to mold tri-leaflet valves in this investigation. Since the current setup is modular, valve replacement is simple should damage occur or modifications made.

1.7. Newtonian vs Non-Newtonian Fluid

Blood is a non-Newtonian fluid [60], [61]. Fluid viscosity changes non-linearly with shear rate. It is common in medical investigations involving blood to use a blood analog fluid to simulate its viscoelastic properties. However, the non-Newtonian nature of blood is typically considered only within small arterial flow where shear rates are low due to low fluid velocity. The Ostwald-de Waele power-law model [15], [62] can be used to describe the relationship between fluid viscosity and shear rate as:

$$\mu = k(\dot{\gamma})^{n-1} \tag{1.5}$$

$$\log(\mu) = (n-1)\log(\dot{\gamma}) + \log(k) \tag{1.6}$$

where:

μ	=	Fluid viscosity [Pa.s]
k	=	Flow consistency [Pa.s^n]
$\dot{\gamma}$	=	Shear rate [s^{-1}]
n	=	Flow index.

The flow index, n, is determined using Equation 1.6 by finding the slope of the log-log relationship of viscosity and shear rate. When $n = 1$, the fluid is Newtonian. Shear thickening and shear thinning fluids are characterized by flow indices of $n > 1$ and $n < 1$ respectively. Blood has $n < 1$, and is therefore a shear thinning fluid; viscosity decreases with increasing shear rate [63]. Figure 1.3 show a plot of viscosity vs shear rate for whole blood adapted from [64] in red, the blue plot respresenting the fitted Ostwald-de Waele power-law model. The plot shows that at shear rates greater than approximately 100 s^{-1}, the viscosity of whole blood reaches a low plateau of approximately 4-5 mPa·s and acts as a Newtonian fluid with negligible non-Newtonian effects [62], [63], [65]. High shear rates greater than $\dot{\gamma} = 100$ s^{-1} are present in large arteries such as the human aorta. Since the purpose of this investigation is to test the effect of varying system parameters, the use of a consistent Newtonian fluid is deemed adequate to show system changes in a comparative manner. Therefore, for the purpose of this investigation, water was used as the experimental fluid due to its ease of access and consistency.

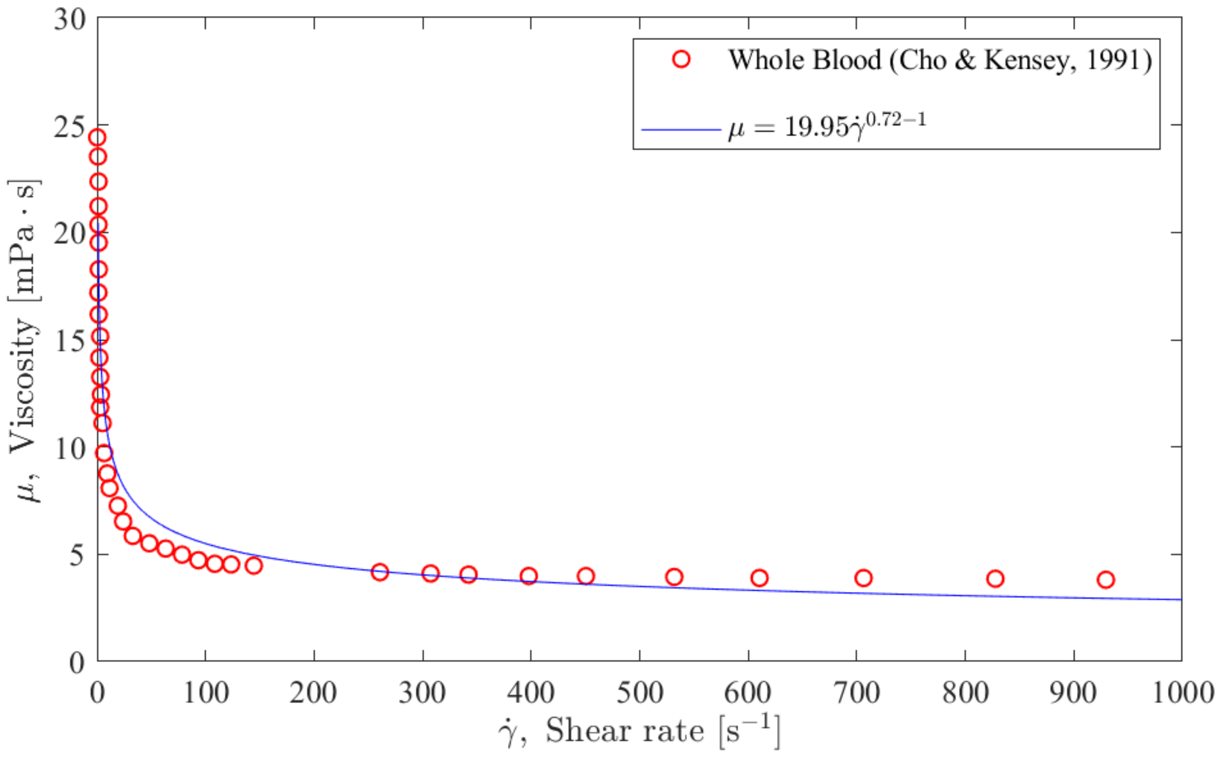


Figure 1.3: Plot of viscosity vs shear rate for whole blood, adapted from [64] in red, with the blue plot representing the fitted Ostwald-de Waele power-law model.

1.8. Research Objectives

In the design of the EVHP system, there is concern the heart does not experience an environment that mimics the physiological effects of the human body. Previous research has shown that including a compliant section at the outlet of a pulsatile pump has a positive impact on pump performance. The main limitation of previous research was that only one compliant mock aorta sample was tested. This investigation seeks to test multiple compliant mock aorta samples under varying systemic parameters to address the following questions:

- What is the effect of changing the length and material of the compliant mock aorta on aortic distension?
- What is the effect of changing pump parameters such as pulse frequency and stroke volume on aortic distension?
- What is the change in pump energy from the upstream to the downstream of the compliant aorta with varying length, material, and distension?
- What is the overall impact of varying the flow-loop parameter of backpressure, as well as varying the aortic parameters of length, material, and distension, on the energy workload of a pulsatile pump at different pump parameters of pulse frequency and stroke volume?

1.9. Content and Outline of Thesis

Four more chapters organize the content of this thesis. Chapter 2 describes the design iterations that led to the final experimental setup. This includes development steps for the pulsatile pump and its pump drive mechanism, molding procedure for the outlet valve and mock aorta compliant tube, pressure-chamber design to house the compliant mock aorta tube, description of the reservoir and backpressure pump, and data acquisition methods. Chapter 3 explains the data processing methodology. Chapter 4 presents and discusses the results of the impact of experimental parameters and aorta tube distension on pump performance. Chapter 5 summarizes conclusions drawn from results, limitations on current experimental setup, and future work.

Chapter 2. Experimental Setup

2.1. Introduction

The objective was to simulate the systemic portion of the EVHP system with the experimental setup and gather pressure and distension data at different locations in order to test the effect of compliant aortic tube properties and flow influences on pump performance. For the aorta, the length and material were varied. For experimental conditions, the backpressure and chamber-pressure within the aorta housing were varied. For the pump parameters, the pulse frequency was varied. Due to the nature of the pump, the stroke volume per pump pulse cycle was directly linked to the pulse frequency. Thus, changing the pulse frequency changed the stroke volume per cycle as well. The fluid pressure at the pump inlet and outlet, and aorta inlet and outlet, were measured. These measurements were used to calculate pump energy at the aorta inlet and downstream from the aorta outlet, as well as the energy change between the two locations. High-speed imaging was used to capture aorta distension effects, and the maximum distension of the aorta under varying system parameters are found. These results will be used to determine the effects of varying system parameters on pump performance.

The experimental set-up is shown in a schematic in Figure 2.1. All major components of the flow loop are labeled. The blue arrows indicate direction of fluid flow. The black arrows indicate electronic component connections to the DAQ. The major components of the experimental setup begin with the pump and its drive mechanism. A MATLAB code was written by Jake Hadfield to generate the heart pressure waveforms used in this study. The objective is to simulate the left side of the EVHP system shown in Figure 2.2(a) with the experimental setup in Figure 2.2(b) and gather pressure and distension data at different locations in order to test the effect of compliant tube properties and flow influences on pump performance. At the outlet of the pump is a silicone trileaflet valve to prevent backflow during the suction phase of the pump cycle, similar to the physiological one in the human heart. Downstream from the valve is the compliant tube and its accompanying pressure-chamber to control the degree of distension. A backpressure pump followed by a reservoir simulates human body conditions. Pressure data is collected at select locations along the flow loop, and compliant tube distension is collected through high speed camera imaging. System parameters were varied via backpressure, compliance chamber pressure, pump pulse frequency, and pump stroke volume. A drawing package for the experimental set-up and molding components can be found in Appendix A.

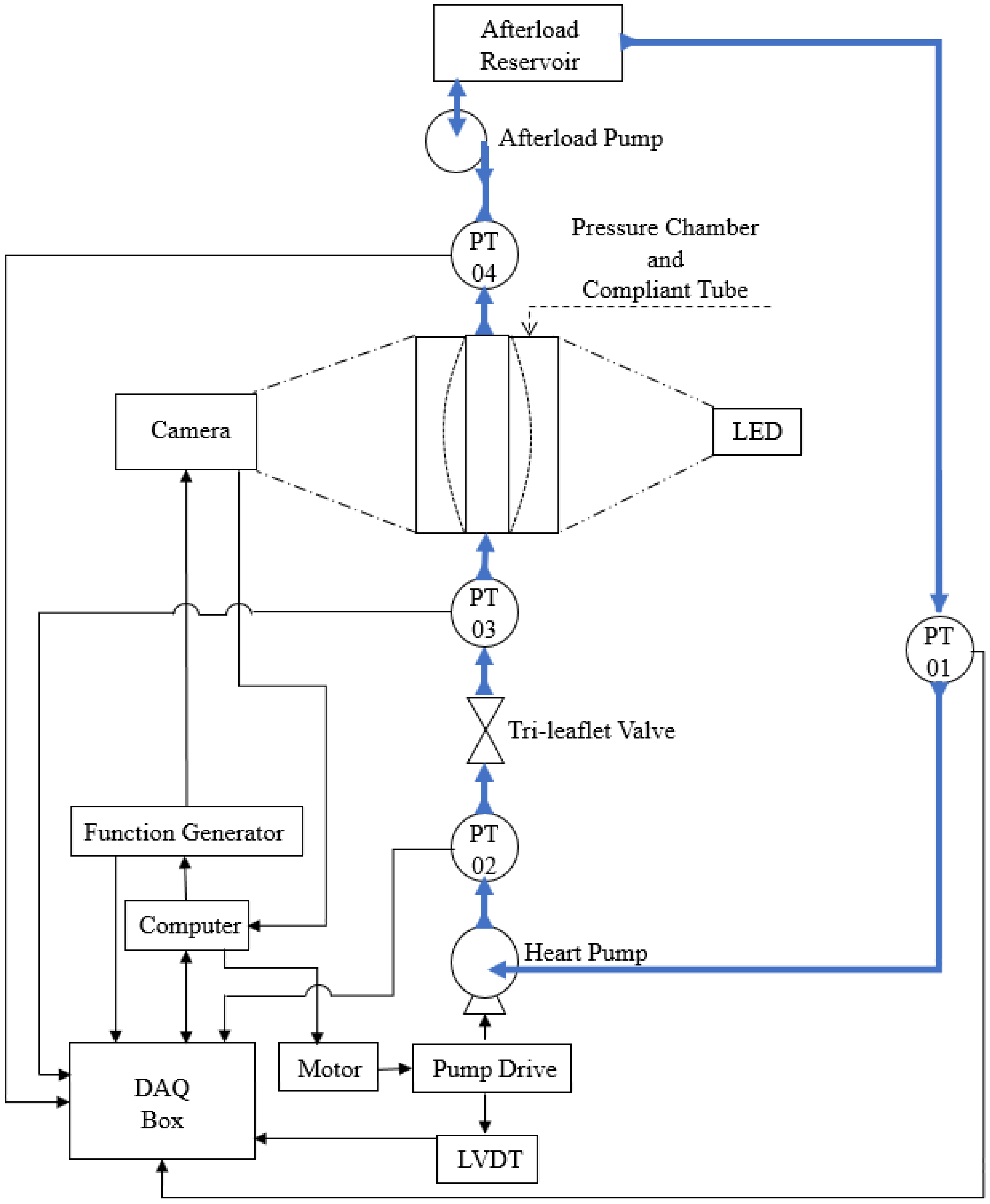
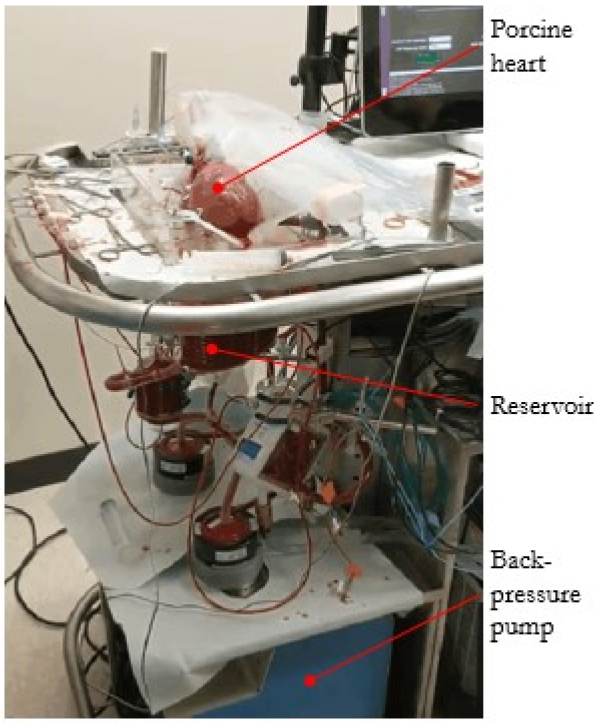
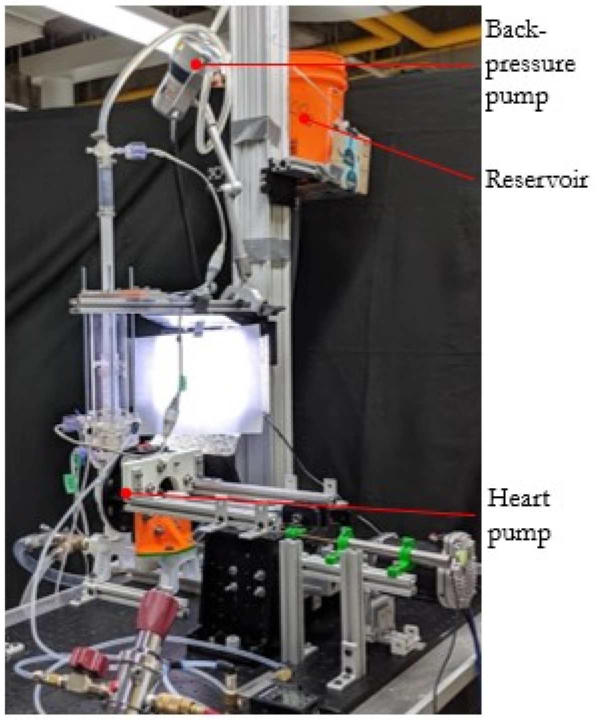


Figure 2.1: Schematic of experimental set-up. PT stands for Pressure Transducer



(a)



(b)

Figure 2.2: (a) EVHP compared to (b) image of the experimental set-up

A detailed description of the experimental setup is provided in this chapter. Each of the following sections describes a major flow loop component in downstream order starting from the pulsatile pump. The final section describes the data acquisition process.

2.2. The Pulsatile Pump

Previous experiments by Cameron [15] used a Ventricular Assist Device (VAD) as the pump simulating the pulsatile action of the heart. The drawback to the VAD was its inability to be controlled and modified. A modifiable pump with control over stroke volume and pulse rate was necessary. The pump must be modular to allow ease of replacement and repair for each component. Ideally, the pump should have similar physical specifications to the physiological heart, allowing comparison between the laboratory flow loop and the medical EVHP system. The specifications for the pump design are given in Table 2.1. Since the heart is a positive displacement pump, it was determined that the most effective solution is a diaphragm pump.

Table 2.1: Pulsatile pump design specifications

Pressure	16/11 kPa systole and diastole, or 120/80 mmHg
Stroke Volume	70 mL
Inlet Diameter	0.75 in or 1.8-2.3 cm for mitral valve
Outlet Diameter	0.75 in or 3.0-3.5 cm for aortic valve

2.2.1. Pulsatile Pump/System Development

The development of the pump system was achieved over four iterations. Figure 2.3 shows labelled images of the four iterations, with the fourth being the chosen design. The first iteration was a fully 3D printed pump with a rolling diaphragm (Class 3C, Bellofram). This pump has three major components secured together. Two pieces provide the housing for the diaphragm attached in-between, while a third cap piece contained both the inlet and outlet with their respective valves. Due to the complexity of the cap design and the limiting size of the rolling diaphragm, it was difficult to 3D print the inlet and outlet section. One of the three major components must be re-printed whenever a design change or repair is required. This required extensive printing time as well as print material. There was also the issue of water leakage through each connecting section. It was decided the pump should be modular at each important section, such as the pump drive mechanism, the inlet and outlet, and the valve components.

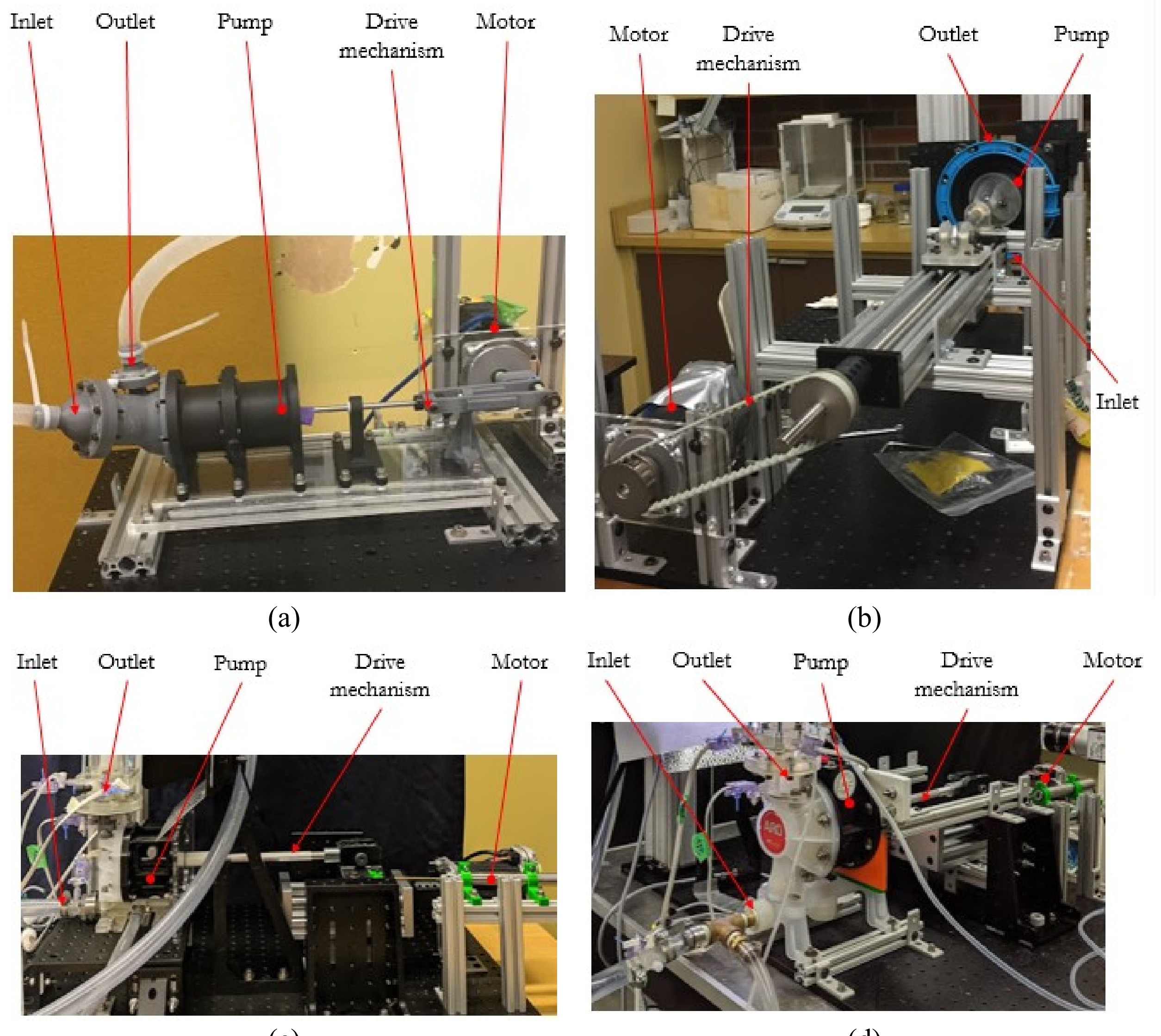
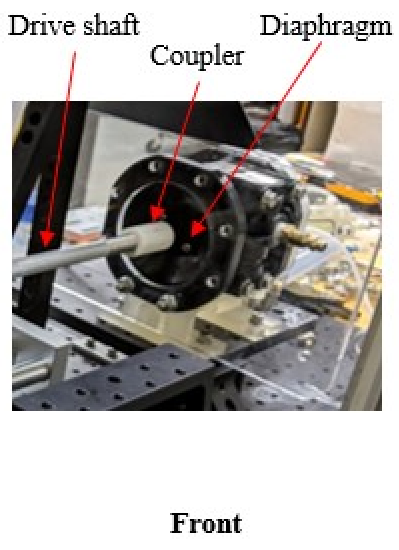


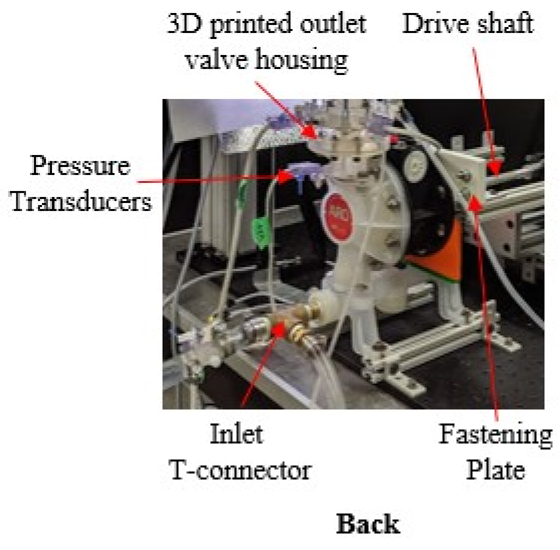
Figure 2.3: Diaphragm pump design progression from (a) fully 3D printed model, to (b) modified bilge pump, to (c) small air operated diaphragm pump to (d) final design using a larger air operated diaphragm pump

The second iteration utilized a bilge pump with modifications to the inlet, outlet, and pump drive mechanism. The advantage was ease of modification the base pump offered. The valve ports were detachable, allowing attachment of 3D printed outlet ports. The handle of the bilge pump was also removable, allowing attachment of the custom pump drive mechanism with ease. The two main issues were the pump's enormous size and the need for it to be oriented vertically. Due to the location of the valve ports and the drive shaft, the pump must be mounted vertically instead of its natural horizontal position. The diaphragm could not support the weight of water without deforming. Since the surface area of the diaphragm was large, it took considerable force for each pump stroke to displace the volume of water. Due to the significant force required to drive the system, vibration and flexing of the entire experimental setup was a major problem in this design.

The third and fourth design iterations both modified Air Operated Double Diaphragm (AODD) pumps. Modifications to the pump can be seen in Figure 2.4. The pump utilized a dual diaphragm system. Each component was secured with standard fasteners and can be easily removed for modification. Although air operated, the diaphragm drive section can be taken apart to attach the in-house drive mechanism for controllable stroke volume and pulse frequency. An outlet is located on either side of the pump. At each outlet is a removable ball valve. Only one side of the diaphragm pump was used, while the other side allows for the attachment of the pump drive mechanism. The third iteration All Flo pump (050-SPP-GGPN-S70, All-Flo Pump Co.) had a stroke volume of 100 mL per stroke when both diaphragm sides were utilized. This was reduced to 50 mL per stroke once the drive mechanism was attached. Since this was less than the required 70 mL, the fourth iteration used a larger AODD pump from ARO (PD05P-ARS-PUU, ARO®/Ingersoll-Rand Inc.). The full stroke volume is 200 mL per stroke, and each side is 100 mL per stroke. The base of the pump was removable, allowing for 3D printed support modifications for firm attachment to the experiment table, thus removing vibration problems during operation.



(a)



(b)

Figure 2.4: Pump modifications for (a) front face with diaphragm and drive shaft connection and (b) back face with outlet valve housing and inlet T-connector

2.2.2. The Selected Diaphragm Pump

The pump selected is the larger AODD pump (PD05P-ARS-PUU, ARO®/Ingersoll-Rand Inc.). The pump can output a maximum of 100 mL per stroke. However, due to the force required to maintain the desired pulse frequency, the actual stroke volume differed based on pump speed. Therefore, the pump was calibrated by averaging total water displaced over 50 cycles. Table 2.2 below show the calibration values and the final stroke volume.

Table 2.2: Pump stroke volume calibration data

bpm	60	80	100	150
cycles	50	50	50	50
Initial water mass (kg)	0.785	0.785	0.785	0.785
Final water mass (kg)	4.464	4.410	4.110	3.055
l/cycle	0.07358	0.0725	0.0665	0.0454

Figure 2.5 shows the manufactures pump performance curve with the necessary experimental operation range boxed in red. The average healthy human blood pressure for systole and diastole is 120/80 mmHg respectively, or approximately 16/11 kPa. This is the pressure specification for the pump design. The AODD pump, when controlled by air, has a pressure range far exceeding the required specifications. However, when replaced with the custom drive mechanism, the stroke pressure can be reduced to approximately 16/0 kPa. Although the diastolic pressure is lower than specifications, this pressure range is sufficient for studying changing parameters on overall pump loop function.

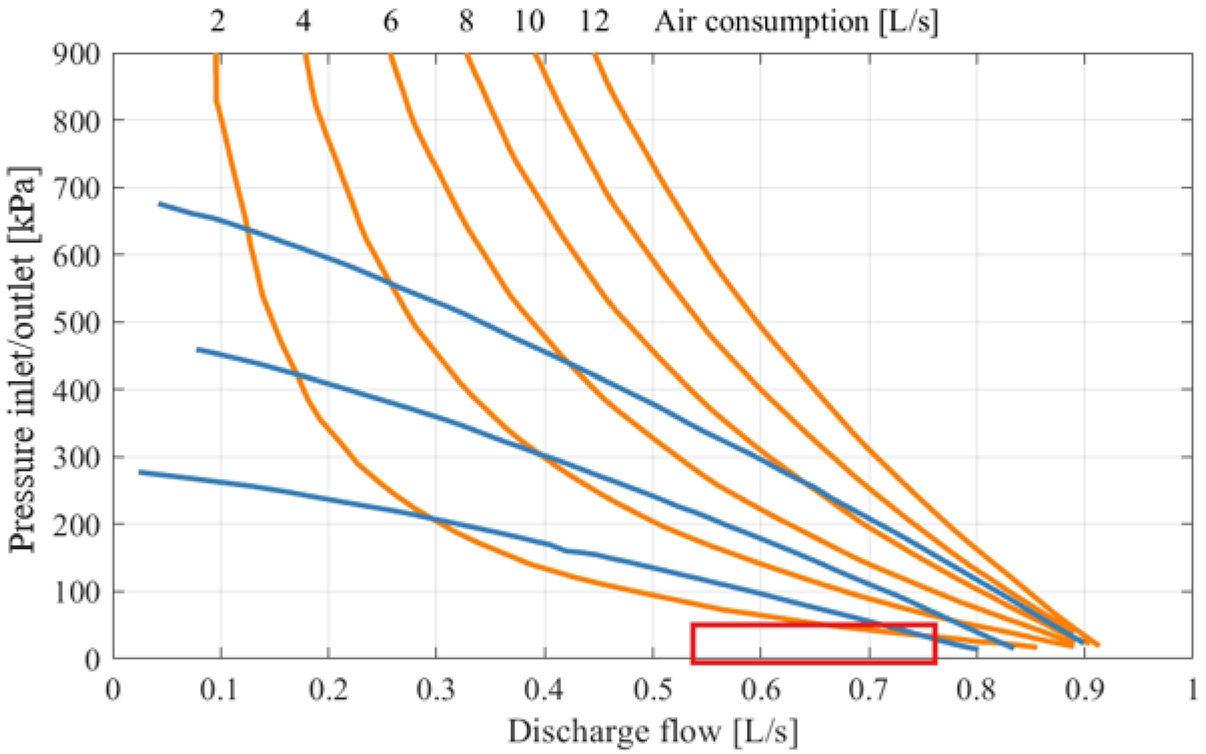


Figure 2.5: Pump manufacturer's performance curve with necessary experimental range boxed in red; adapted from [66]. The y-axis represent pressure for both air and water. The x-axis represents pump discharge flow rate. The blue curves represent operating air pressure plots, and the orange curves represent the flow rate of air required.

2.3. Pump Drive Mechanism

One of the improvements to the previous investigation [15] is the added control over the pump stroke volume and pulse frequency. The diaphragm was directly connected to a motor drive mechanism. This motor drive mechanism is programmed to provide a heart waveform at a defined pulse frequency. Figure 2.6 below shows an image of the final pump drive mechanism.

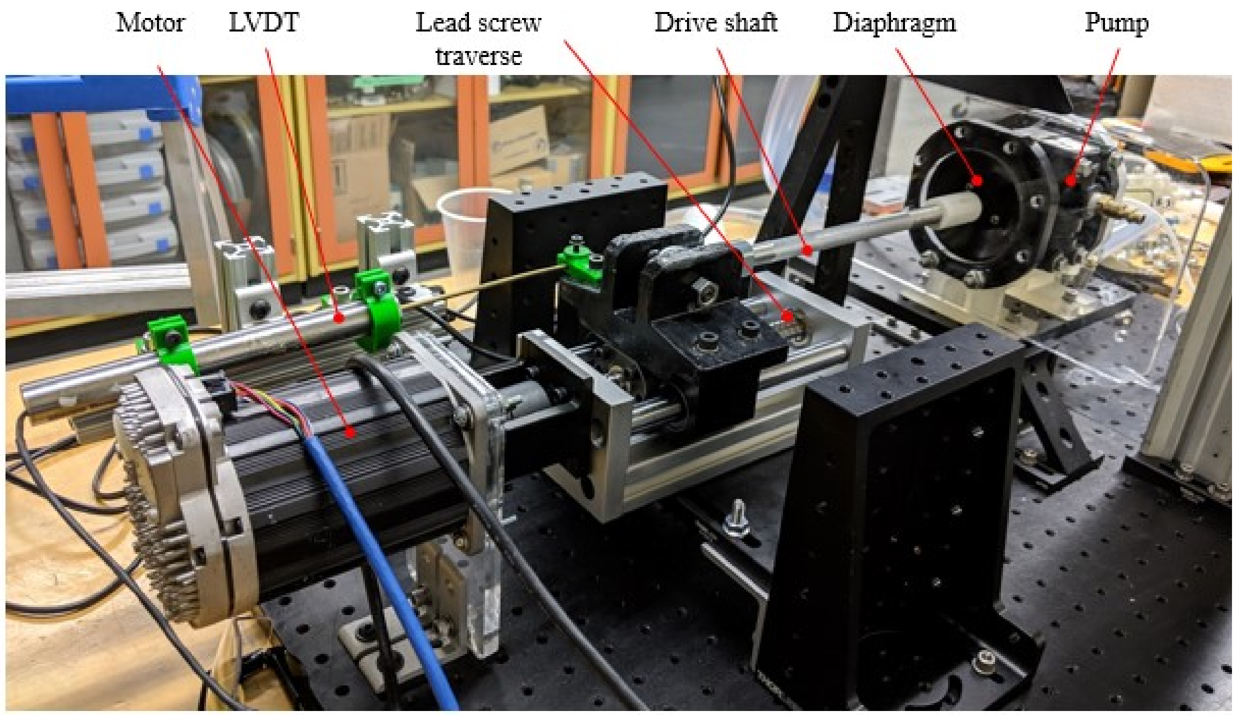


Figure 2.6: Labelled image of pump drive mechanism

The drive mechanism is driven by an integrated servo motor (Clearpath SCHP ELSA, Teknic, Inc.). The motor position is controlled using software API programmed in C++[1]. The motor is

[1] Code author: Jake Hadfield, Department of Mechanical Engineering, University of Alberta

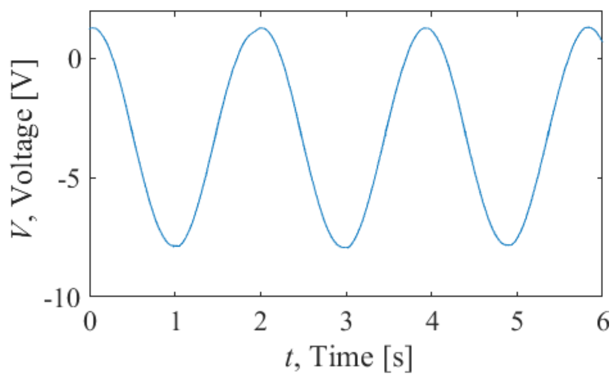
programmed to produce the desired pulsatile heart waveform. The code to run the motor to generate the heart waveform can be seen in Appendix B.

The pump is driven by a lead screw traverse (100mm /3.9inch CNC Linear Sliding Table, Walfront), with the specifications of the traverse shown in Table 2.3. The lead screw transforms rotational motion to linear motion. The conversion of distance travelled and pump displacement is 1:1. The traverse is directly attached to the connecting rod driving the pump diaphragm. The specifications of the traverse are show in Table 2.3 below.

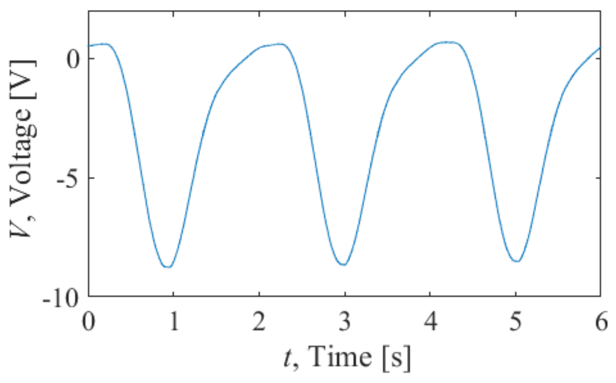
Table 2.3: Specifications of the lead screw traverse used to transform the rotary motion of the control motor into linear motion

Diameter	16 mm
Effective stroke	100 mm
Pitch	5 mm

The distance traveled by the traverse is recorded using a Linear Variable Differential Transformer (LVDT). The LVDT is an electromechanical transducer which converts the rectilinear movement of the traverse into a voltage signal captured by the DAQ. Figure 2.7 shows a voltage vs time plot of both a sine and heart waveform function outputted by the motor to the traverse, measured by the LVDT based on distance traveled by the traverse. The heart waveform function is used for this experiment to simulate the pulsatile nature of the heart. As shown in Figure 2.6, the LVDT housing is secured to the stationary motor housing, and the core is coupled to the drive shaft of the pump. The LVDT data is used as a measurement of pump displacement Since the pump and traverse operated on a 1:1 system, an increase in LVDT displacement corresponds directly to an increase in pump fluid displacement and vice versa



(a)



(b)

Figure 2.7: Plot of voltage data corresponding to motor rotation showing (a) sine vs (b) heart waveform generated by the motor

2.4. Compliant Components

The two compliant components in the flow loop consist of the trileaflet valve and the compliant tube. At the outlet of the pump is the trileaflet valve to prevent backflow during the suction stroke. Downstream from the trileaflet valve is the compliant tube simulating the human aorta. Both the trileaflet valves and the compliant tubes are cast from silicone using 3D printed molds. The molding procedure for both components are the same, the only difference being the mold used. This section is split into three parts. The first describes the molding procedure for the trileaflet valve, which also applies to the compliant tube. The following two sections describe the valves and compliant tubes respectively in detail, as well as their mold design.

2.4.1. General Molding Procedure

The molding procedure explained in this section is for the silicone valves. The compliant tube was molded using the same procedure. The molds are designed in SolidWorks and 3D printed in the FormLabs Form 2 printer. The specifics of each mold design are described in section 2.4.2 Outlet Valve. Three different silicones were tested for the trileaflet valves. The silicones used were Ecoflex 00-50, Dragon Skin 10, and SYLGARD 184. Their specifications are listed in Table 2.4. The same mold was used to create each type of valve, regardless of silicone type.

Table 2.4: Silicone specifications

	Ecoflex [67]	Dragon Skin [68]	SYLGARD 184 [17], [20]
100% modulus	82.7 kPa	151.7 kPa	1316-2972 kPa
Shore hardness	Shore 00-50	Shore 10A	Shore 44-50A
Pot life	18 minutes	45 minutes	1 hour
Cure time	3 hours	16 hours	24 hours
Curing temperature	25 °C	25 °C	23 °C

Only the Ecoflex and Dragon Skin silicones were used for the compliant tube. The compliant tube did not have a SYLGARD 184 case due to the brittleness of the SYLGARD 184. The Ecoflex and Dragon Skin silicones are both manufactured by Smooth-On. Both present enough flexibility for demolding after curing. Thus, the same mold could be used for both tubes. The SYLGARD tube, due to its brittleness, could not be removed from the mold without damage.

The mold was first designed using SolidWorks, and the SolidWorks Plastics package used to simulate the silicone filling process of the designed mold. The SolidWorks simulation was used as a predictor of the mold filling process and was used to determine optimal locations for air vents. Detailed engineering drawings of the mold designs can be found in Appendix A.

All three silicones are two-part silicones. Ecoflex and Dragon Skin are supplied with parts A and B, while Sylgard 184 has a base and a curing agent. For the Ecoflex and Dragon Skin silicones, parts A and B were mixed at a volume ratio of 1:1. For the Sylgard 184, a base to curing agent volume ratio of 10:1 is followed. A release agent (Ease Release™ 200, Mann) was first applied to the inner surfaces of the negative mold, and outer surfaces of the positive mold. Once mixed, the Ecoflex has a pot life of 18 minutes, the Dragon Skin has a pot life of 45 minutes, and Sylgard has a pot life of 4 hours. This mixture was then placed into a vacuum chamber to remove air bubbles. The silicone was then drawn into a 100 mL syringe housed in a custom-built syringe pump. Due to the viscosity of the silicone, commercial syringe pumps could not produce the required force to displace the syringe plunger. A custom syringe pump was built using a motor-traverse mechanism. The silicone was then injected into the mold through Luer lock connections. Figure 2.8 shows a labelled image of the silicone injection process. Once the mold was fully filled and silicone had begun to extrude through the air outlets, the Luer lock connections were closed using Luer caps. The silicone then cured at each material's specified temperature and cure time, then removed from the mold.

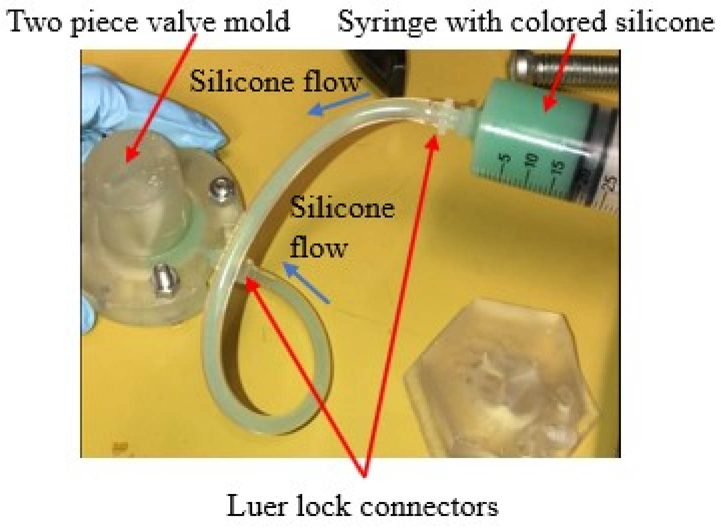
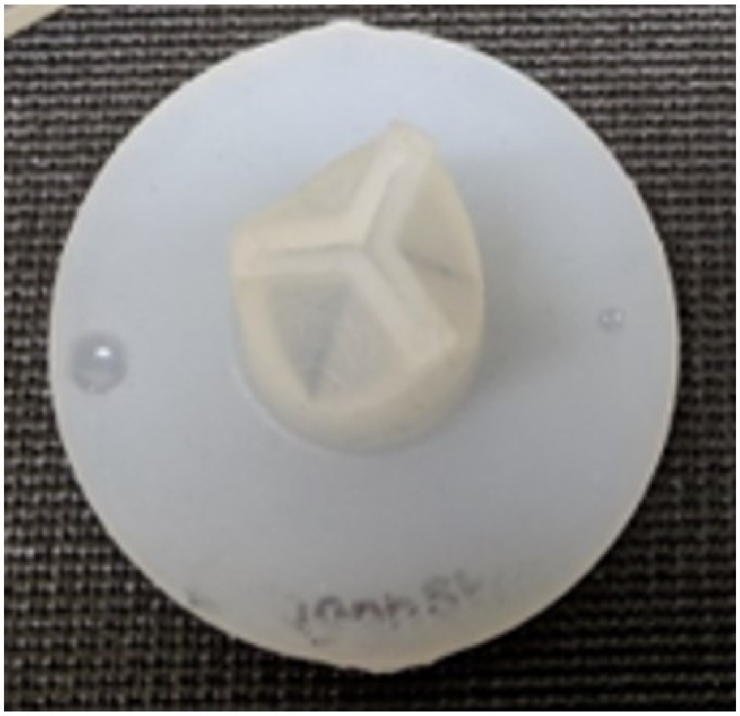
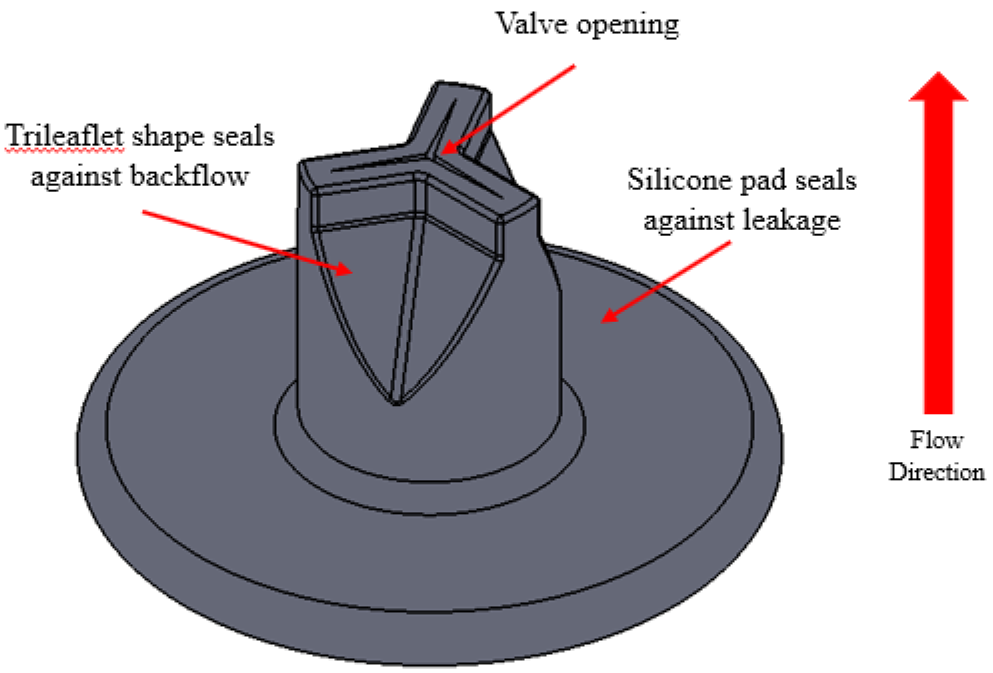


Figure 2.8: labelled image of the silicone injection process. Silicone flow direction is labelled using blue arrows and molding components labelled using red arrows.

2.4.2. Outlet Valve

The human heart has a trileaflet valve at its outlet. Figure 2.9 (a) shows an image of the molded silicone valve used in the pump. The original ball valves of the AODD pump were removed and replaced with silicone trileaflet valves. Figure 2.9 (b) shows an annotated isometric view of the silicone trileaflet valve solid model.


(a)


Valve opening
Trileaflet shape seals against backflow
Silicone pad seals against leakage
Flow Direction
(b)
Figure 2.9: Images of (a) molded silicone trileaflet valve and (b) anotated solid model of silicone valve

The pump outlet has a diameter of 12.7 mm (0.5 in), smaller than the experimental aortic diameter of 19.05 mm (0.75 in). A valve seat was manufactured to sit between the pump outlet and the valve. The valve seat functions as both a location for a Luer lock connector to attach a pressure transducer, and as a transitional length to reduce entrance effects between the pump exit and the valve inlet. Figure 2.10 shows engineering drawings of the Dragon Skin valve. The valve diameter was chosen based on human dimensions and experimental need. A human aortic valve leaflet has a thickness range of 0.35-3.5 mm at the leaflet tip and 0.25-0.70 mm at the leaflet center [72]. Due to the molding process, leaflet thickness depends on the space between the positive and negative mold. The silicone will have issues entering spaces approximately less than 1 mm, causing holes in the final molded valve. Successfully molded leaflets that are too thin will collapse when subjected to pump pressures. Through testing trials, a valve leaflet thickness of 2.5 mm was chosen for its structural integrity against pump pressures while lying in the general range of physiological thicknesses. The internal diameter of an aortic valve has a range between approximately 17.8-23.8 mm [54], [73]. Since the experimental aorta was molded at internal diameter of 19.05 mm, the valve and its valve casing attachment to the pump were designed to match closely while allowing for valve leaflet thickness. A 17.4 mm valve internal diameter was then chosen, which leads to an external diameter of 22.4 mm. The casing has an internal diameter of 25 mm, with a neck transitioning to 19.05 mm to match with the internal diameter of the aorta.

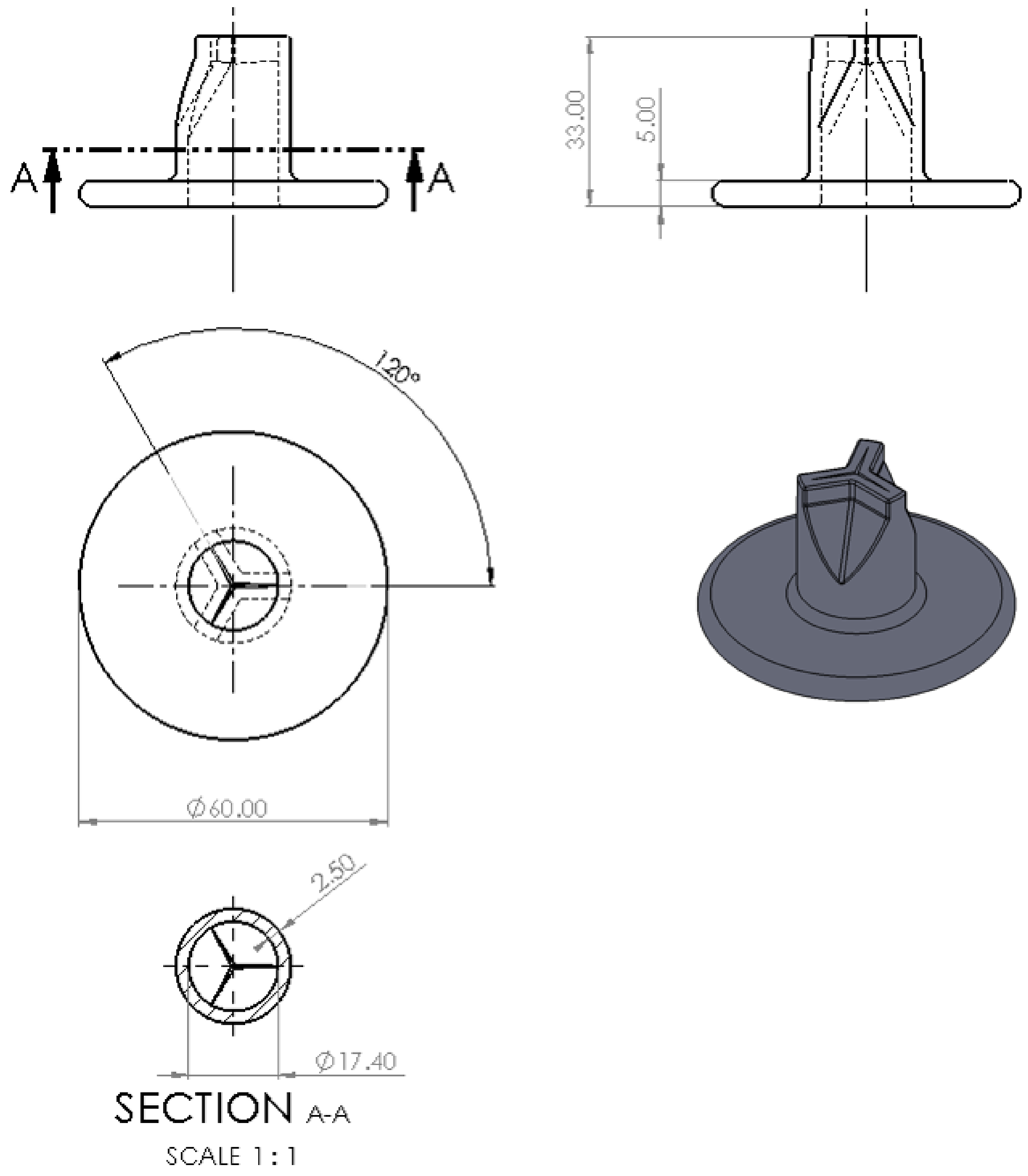


Figure 2.10: Dimensioned engineering drawing views of Dragon Skin valve

Three different silicones were tested when making the trileaflet valve. Identical molds were used for the casting of each valve, ensuring the same dimensions of each. Figure 2.11 shows annotated images of the 3D printed positive (left) and negative (right) mold, and Figure 2.12 and Figure 2.13 shows the engineering drawing exploded view and dimensioned orthographic views of the molds respectively. The valves were tested by running the then in-progress experimental setup with a heart waveform. All valves of the three difference materials could withstand the pressure conditions of the system without collapsing from backflow. Due to the ease of casting multiple replacement valves, Smooth-On silicones were preferred. Since the Dragon Skin is a stiffer silicone with a higher 100% modulus, it was less likely to damage or collapse with prolonged use than Ecoflex. Therefore, the Dragon Skin valve was chosen and used for all experimental cases.

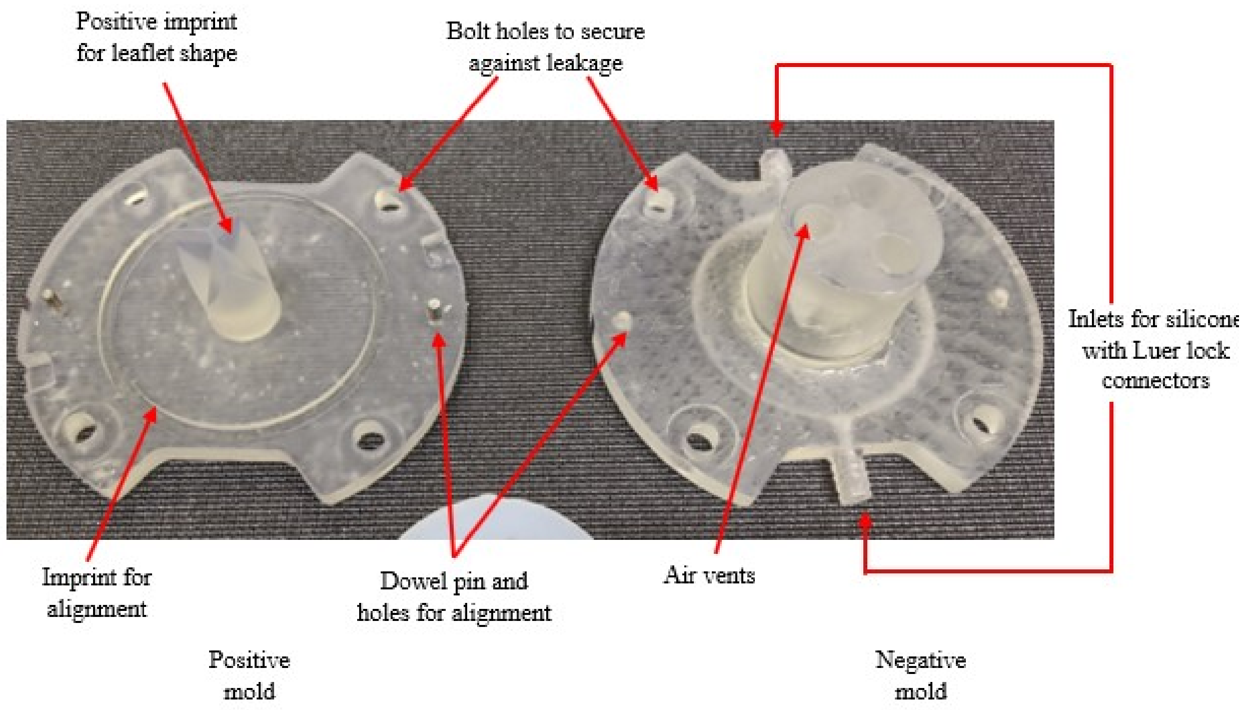
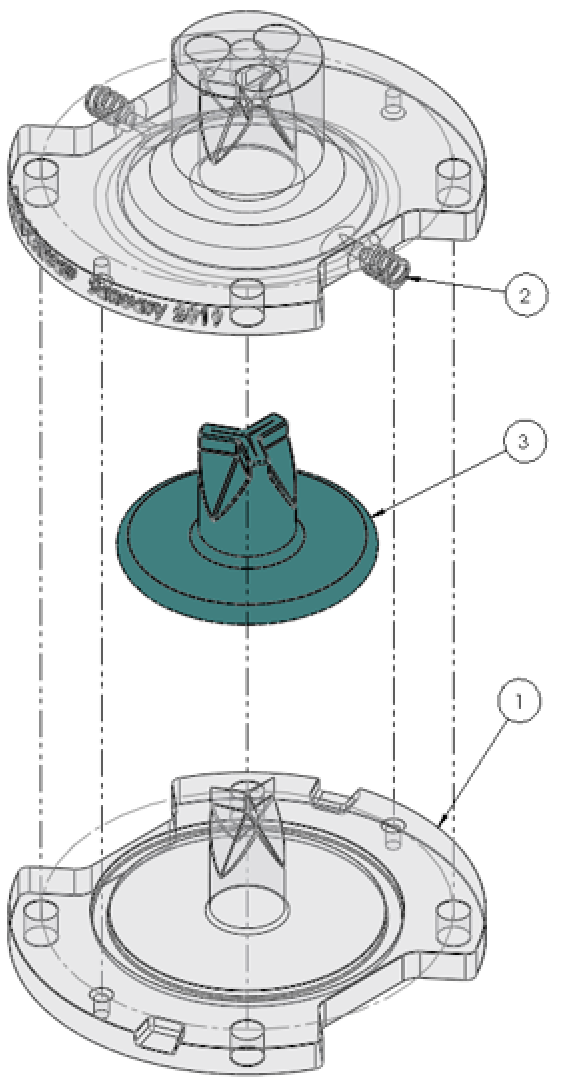
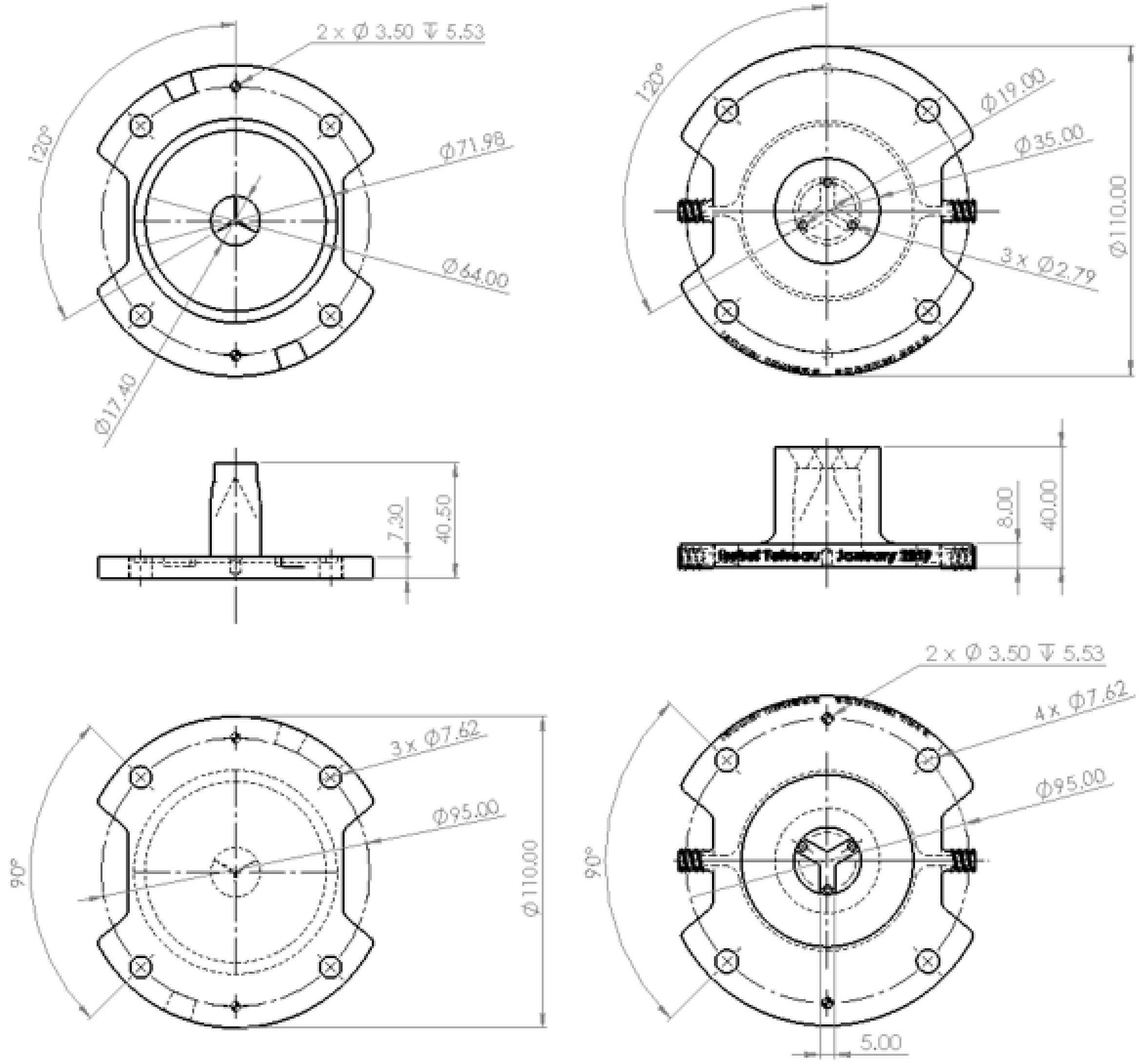

Positive imprint for leaflet shape
Bolt holes to secure against leakage
Inlets for silicone with Luer lock connectors
Imprint for alignment
Dowel pin and holes for alignment
Air vents
Positive mold
Negative mold

Figure 2.11: Annotated image of 3D printed trileaflet valve silicone positive (left) and negative (right) mold.



ITEM NO.	PART NAME	QTY.
1	Valve Positive Mold (Bottom)	1
2	Valve Negative Mold (Top)	1
3	Silicone Trileaflet Valve	1

Figure 2.12: Engineering drawing of trileaflet valve mold assembly exploded view



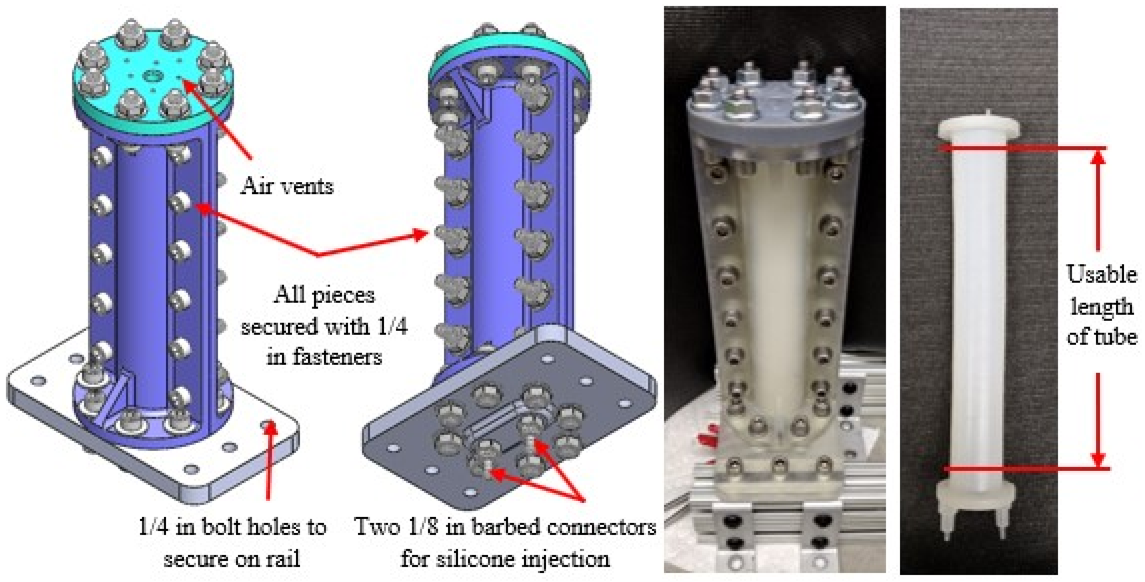
Positive Mold (Base) (a)

Negative Mold (Top) (b)

Figure 2.13: Dimensioned engineering drawing of trileaflet valve mold, where (a) is the positive mold (base) and (b) is the negative mold (top)

2.4.3. The Compliant Tube

The human aorta is compliant [14]. Previous research has shown that adding a compliant section to the pump loop improved pump performance [15]. However, the full effect of a compliant section has yet to be explored, since only one compliant tube was tested. Silicone rubbers demonstrate a similar elastic response to human arteries [23]. The Ecoflex and Dragon Skin silicones from Smooth-On have been selected for the casting of different lengths of compliant tubes. Figure 2.14 below show annotated images of the aorta mold solid model in subfigure (a), an image of the assembled tube mold during curing in subfigure (b), and the demolded Dragon Skin cast tube before it is cut down to its usable length in subfigure (c).



Air vents
All pieces secured with 1/4 in fasteners
1/4 in bolt holes to secure on rail
Two 1/8 in barbed connectors for silicone injection
Usable length of tube

(a) (b) (c)

Figure 2.14: Images of (a) compliant tube mold during the curing process and (b) the raw demolded Dragon Skin compliant tube

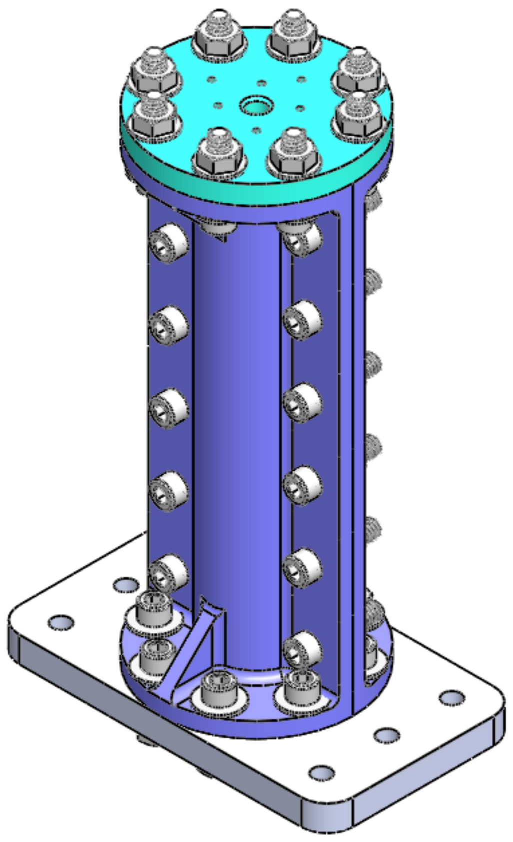
The human aorta has a diameter of approximately 0.75-0.9 in [69]–[71], therefore 0.75 in (19.05 mm) was the chosen diamter due to ease of keeping the flow loop consistent. The lengths tested were a multiplier of the diameter. ×3, ×5, and ×7 the diameter gave approximately even lengths of

60, 100, and 140 mm respectively. Table 2.5 below lists the compliant tube specifications. Using the chosen aorta size, an aorta mold was designed to manufacture the maximum length of tube, 140 mm. The molded tubes are then cut down to the size required.

Table 2.5: The specifications of the compliant tube

Silicones used	Ecoflex 00-50	Dragon Skin 10
Internal diameter	19.05 mm (0.75 in)	
Lengths	60, 100, 140 mm	
Wall thickness	4 mm	
100% modulus	82.7 kPa	151.7 kPa

Figure 2.15 shows an isometric and exploded view of the aorta mold assembly. Figure 2.16 shows the dimensioned orthographic views. The full drawing package of the aorta mold is included in Appendix A. The aorta mold is made up of 5 major components. The main body of the mold is two identical side pieces labeled as item #2 in the exploded view. The internal diameter of the main body when assembled is 23.05 mm (0.907 in). In the center of the main body is a 19.05 mm (0.75 in) outer diameter Delrin tube core. The silicone fills between the outer surface of the Delrin tube and the internal surface of the main mold body, forming the specified 19.05 mm internal diameter aorta with 4 mm wall thickness. The Delrin tube is machined on both ends with alignment features for both the top piece and base plate to allow precise alignment within the center of the mold. The top piece has eight staggered air vents at the top to allow air to escape during silicone filling. The bottom plate has two 1/8 barbed tube fitting for attachment of tubes with silicone feed from the syringe pump. There are 1/4 in bolt holes located on the bottom plate to fasten to rails below the mold to hold the assembly upright during molding. All pieces are secured together with standard 1/4 in fasteners via bolt holes included in the sides and the top and bottom of the mold assembly.



ITEM NO.	PART NAME	QTY.
1	Aorta mold base plate	1
2	0.125" barbed tube fitting	2
3	Aorta mold side piece	2
4	Aorta mold top piece	1
5	0.75" OD Delring tube core	1
6	0.25" flat washer	16
7	0.25" socket head cap screw	28
8	0.25" flange nut	28

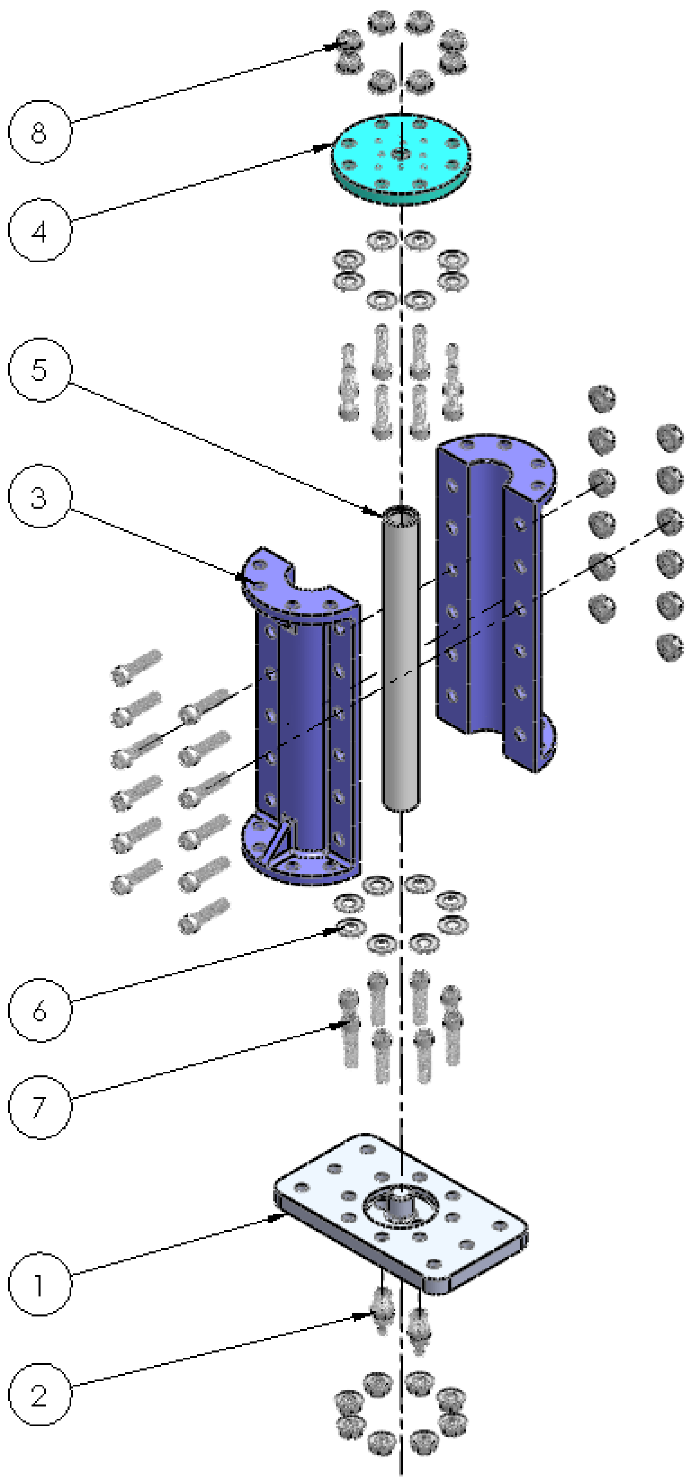


Figure 2.15: Aorta mold assembly isometric view (left) and exploded (right) view with bill of materials included

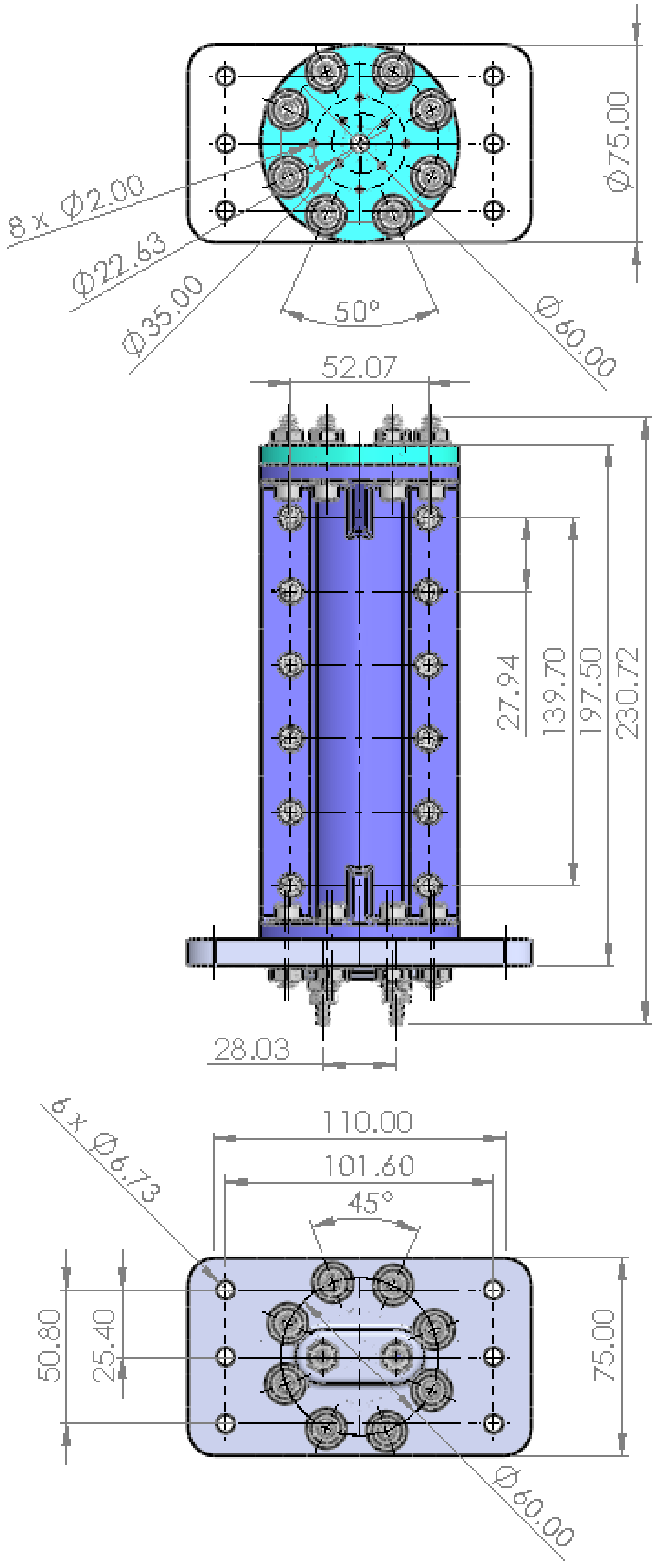


Figure 2.16: Dimensioned orthographic view of aorta mold assembly. Full drawing package of aorta mold parts included in Appendix A.

2.5. Pressure-Chamber

The compliant tubes required containment within an air pressure-chamber to control its compliant response. Without regulation, backpressure within the system caused the tubes to distend to the point of damage. With air pressure regulation of the tube distension, effect of mock aorta material on pump energy can be studied. Figure 2.17 below shows an image of the pressure-chamber with the compliant tube contained within. The isometric and exploded view are shown in Figure 2.18. Figure 2.19 shows a dimensioned orthographic engineering drawing of the full pressure-chamber assembly, as well as the clear acrylic chamber.

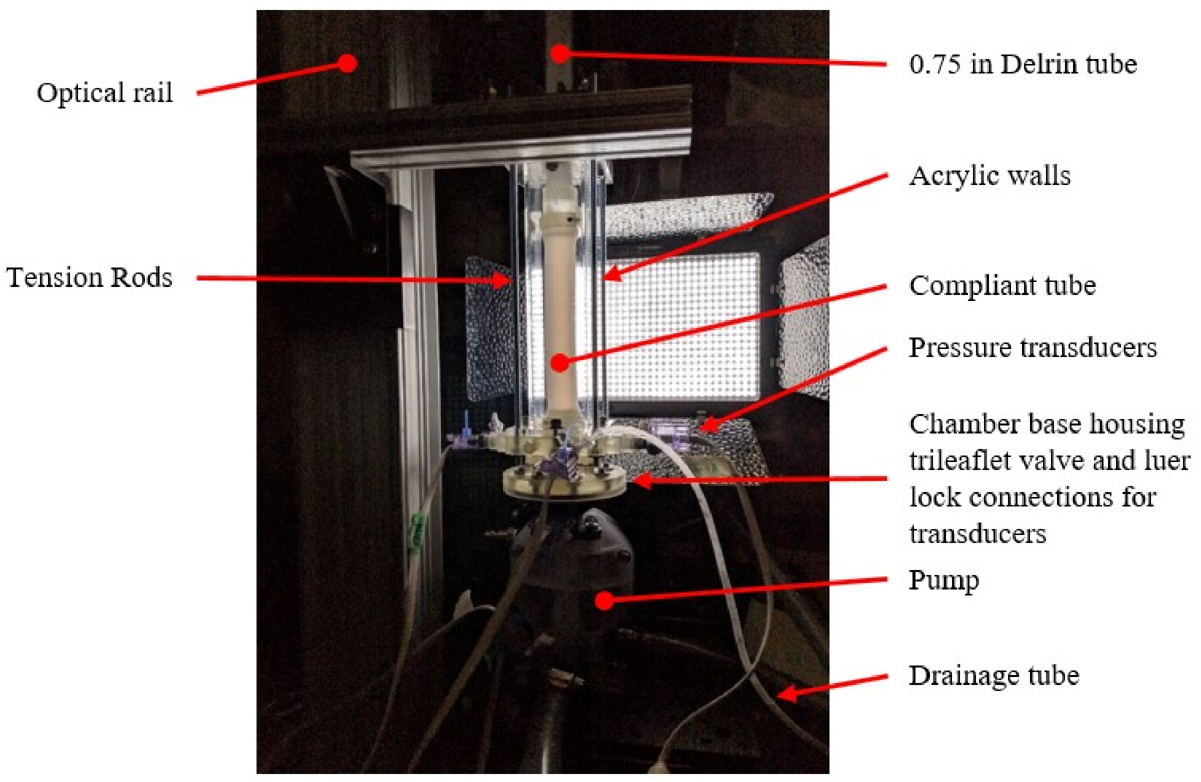
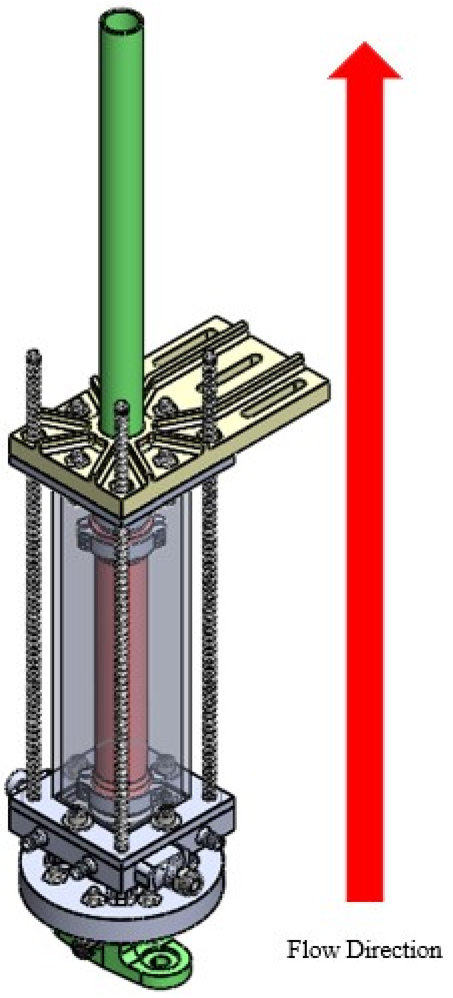
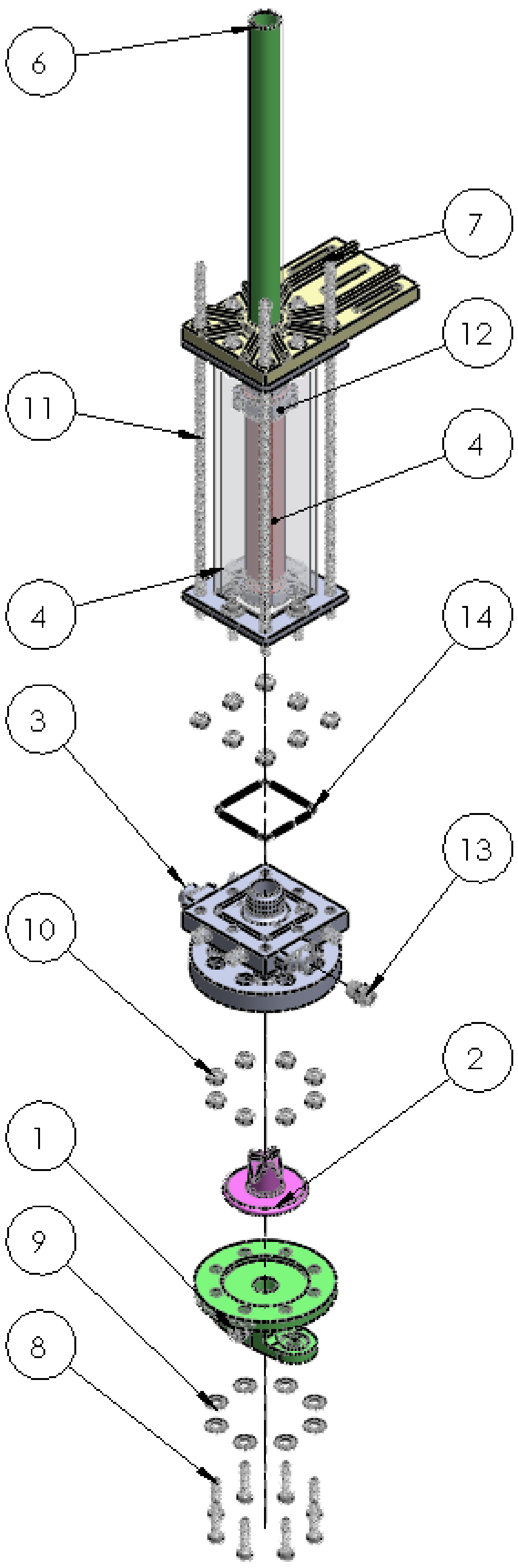


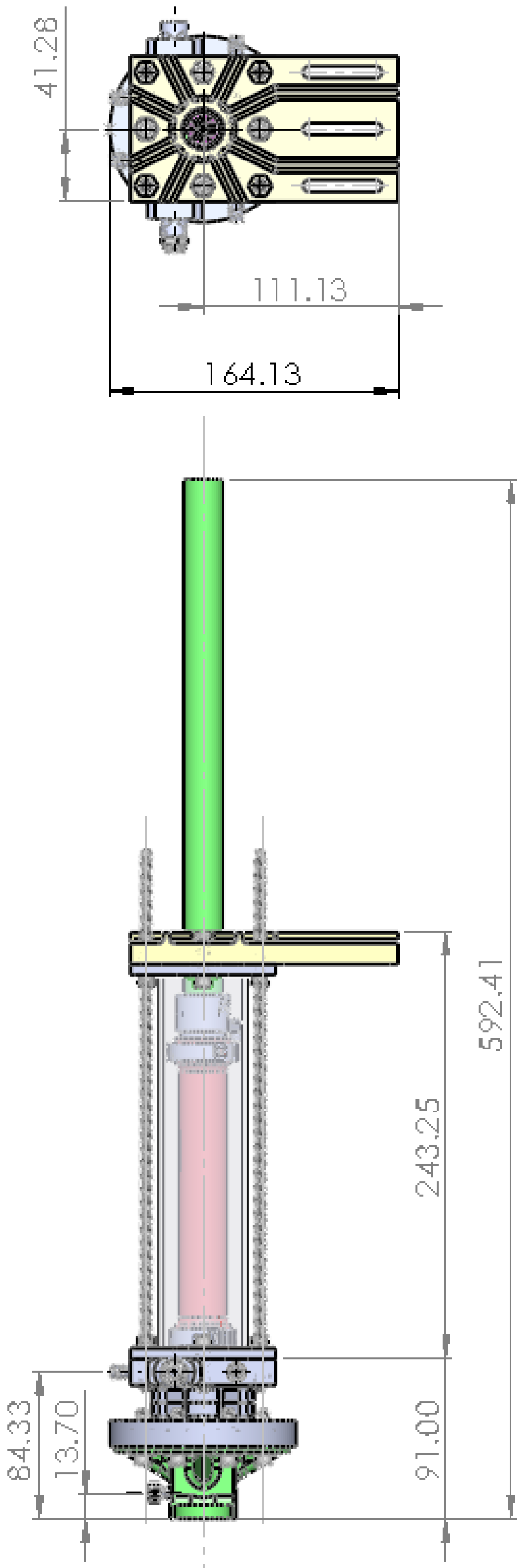
Figure 2.17: Pressure-chamber (a) labeled image and (b)

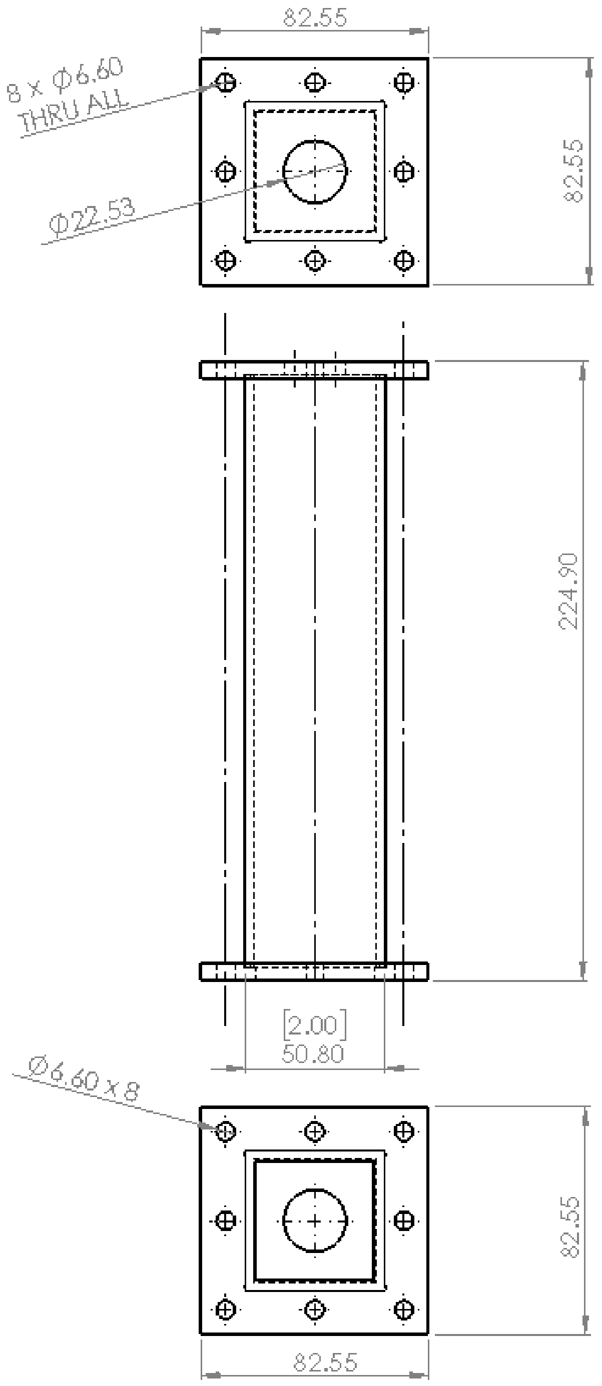




ITEM NO.	QTY.	Part Name
1	1	Pump Connection Plate
2	1	Trileaflet Valve
3	1	Valve Casing
4	1	Assembly - Acrylic Chamber
5	1	Silicone Aorta
6	1	Delring tube
7	1	Square Flange Cap
8	16	0.25" Socket Head Bolt Short 1" long
9	16	Washer for 0.25" Bolt
10	20	0.25" Hex Nut
11	4	Threaded Rod - 0.25"
12	4	C-Clamp
13	1	5779K109 Push-to-connect tube fitting for air
14	1	Valve Casing O-ring

Figure 2.18: Pressure-chamber isometric view (left) and exploded (right) view with bill of materials



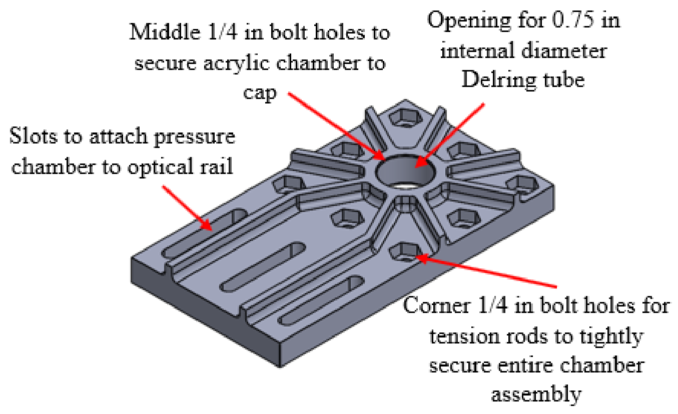


(a) (b)

Figure 2.19: Dimensioned engineering orthographic drawing of (a) pressure chamber assembly and (b) acrylic chamber

The clear viewing window of pressure-chamber where the compliant tube is housed is built using a 2×2 inch square acrylic tube. Sealing the chamber at the top is a 3D printed square flange cap. The valve casing acts both as a seal for the base of the chamber as well as a housing for the trileaflet valve. The entire pressure-chamber assembly is clamped tightly using four tension rods at each corner. O-rings are located at both ends of the acrylic tube to prevent leakage of air to the surroundings. A 3D printed bracket secures the pressure-chamber upright against an optical rail to stabilize and mount the system.

Figure 2.20 shows an annotated isometric view of the square flange cap solid model. Figure 2.21 shows the dimensioned orthogonal engineering drawing views. The 3D printed square flange cap houses a 12 in long and 0.75 in diameter Delrin tube secured by an O-ring to prevent leakage of air from the chamber to the surroundings. The outlet of the compliant tube is secured on one end of the Delrin tube using 3D printed clamping pieces inside the pressure-chamber. Using this length of tube, different lengths of the aorta can be clamped to the tube inside the pressure chamber by extending and retracting the length within the chamber. The exit of the Delrin tube extends outside the pressure-chamber and is connected to the rest of the flow loop via a 3D printed barbed piece with a Luer lock connector for a pressure transducer.


Middle 1/4 in bolt holes to secure acrylic chamber to cap
Opening for 0.75 in internal diameter Delring tube
Slots to attach pressure chamber to optical rail
Corner 1/4 in bolt holes for tension rods to tightly secure entire chamber assembly

\

Figure 2.20: Annotated isometric view of square flange cap

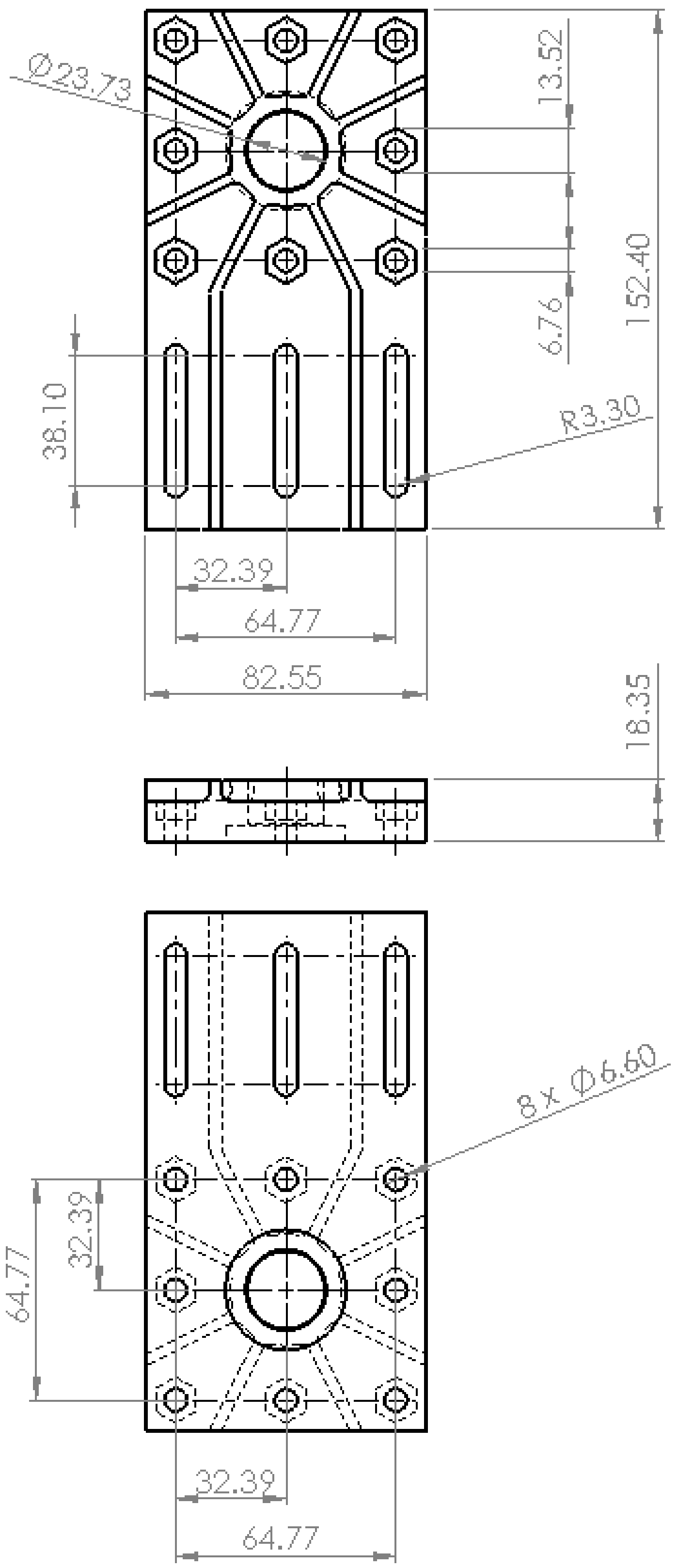


Figure 2.21: Dimensioned orthogonal engineering drawing views of square flange cap

Figure 2.22 shows an annotated isometric view of the valve casing solid model, with important features labeled. Figure 2.23 shows dimensioned engineering orthogonal drawing views of the valve casing. Detailed engineering drawings are show in Appendix A. The valve casing is a modular section containing the trileaflet valve attached to the pump outlet. A barbed 0.75 in connector protrudes from the top of the valve casing on which the compliant tube is secured using 3D printed clamping pieces. There are four Luer lock connectors printed onto the side of the valve casing, two of which branch off directly from the flow-loop. The first is connected to a pressure transducer for measurements at the entrance to the compliant section. The second allows for the attachment of a drainage tube to allow ease of draining the system. The third Luer lock connection allows a pressure transducer to measure the air pressure within the chamber. The fourth is an unused backup connection to the internal air pressure within the chamber. There are two air outlet connections built in for internal air pressure control, where one is a backup that is unused under normal operating conditions. After the valve casing is printed, the air outlet opening is threaded to fit a standard 1/4 NPT pipe size, 18 threads per inch push-to-connect fitting. Air is delivered to the pressure-chamber first through a reservoir pressure tank (IT20ASME, Industrial Air), a pressure monitor (PRX 21213, ProStar), a pressure regulator (68027-46, Cole-Parmer), and finally to the internal chamber. The pressure regulator adjusts the pressure within the chamber to the specified setting. The pressure-chamber has been tested for up to 34.5 kPa chamber pressure without leakage. Therefore, experimental chamber pressure ranges from 0-34.5 kPa in increments of 3.4 kPa.

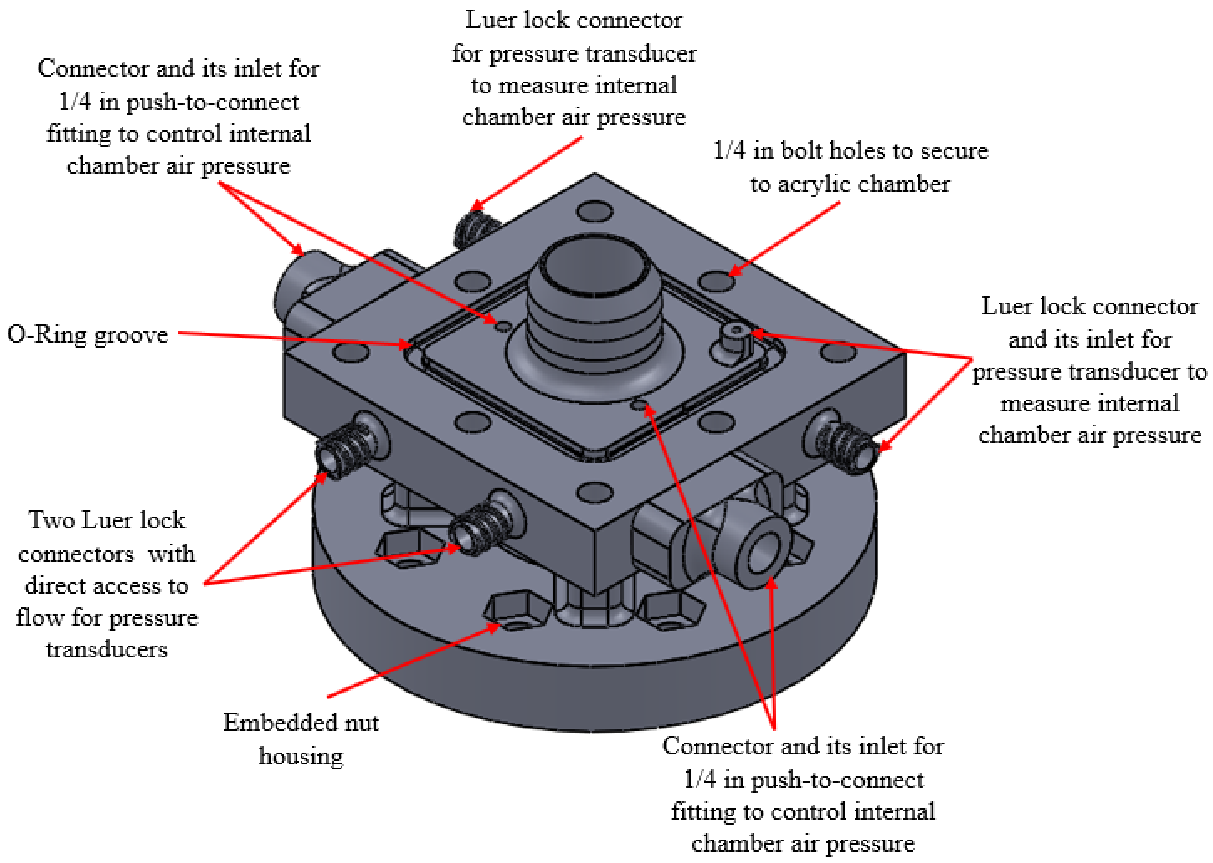

Connector and its inlet for 1/4 in push-to-connect fitting to control internal chamber air pressure
Luer lock connector for pressure transducer to measure internal chamber air pressure
1/4 in bolt holes to secure to acrylic chamber
O-Ring groove
Luer lock connector and its inlet for pressure transducer to measure internal chamber air pressure
Two Luer lock connectors with direct access to flow for pressure transducers
Embedded nut housing
Connector and its inlet for 1/4 in push-to-connect fitting to control internal chamber air pressure

Figure 2.22: Annotated isometric view of valve casing solid model

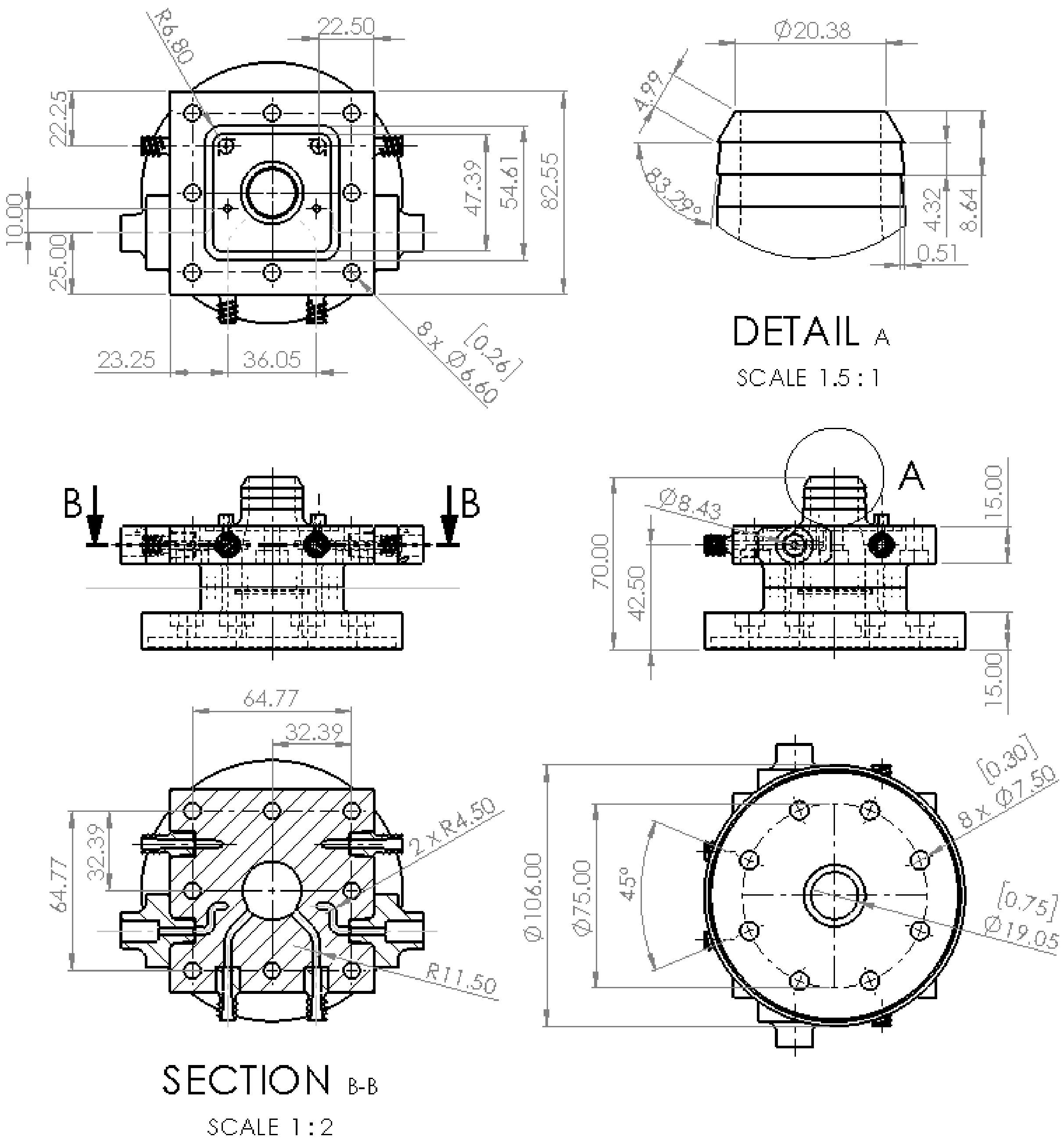


Figure 2.23: Dimensioned engineering orthographic drawing of pressure chamber assembly

Figure 2.24 shows annotated views of the pump connection plate solid model, with important features labeled. Figure 2.25 shows the dimensioned engineering orthographic views. The 3D printed pump connection plate was designed as a connecting piece between the large AODD pump (PD05P-ARS-PUU, ARO®/Ingersoll-Rand Inc.) outlet and the valve housing. Two slots with washer indents are located on the connection plate feet to secure to the pump outlet slots using 1/4 in nuts and bolts. At one side of the connection plate is a Luer lock connector for a pressure transducer to measure pump outlet flow pressure. At the top of the piece is a plate containing a circular indent to fit the silicone valve, where the silicone pad at the base of the valve acts as a seal against flow leakage. Around the valve indent are 1/4 in bolt holes to allow attachment to the valve housing, which has matching bolt holes in its base.

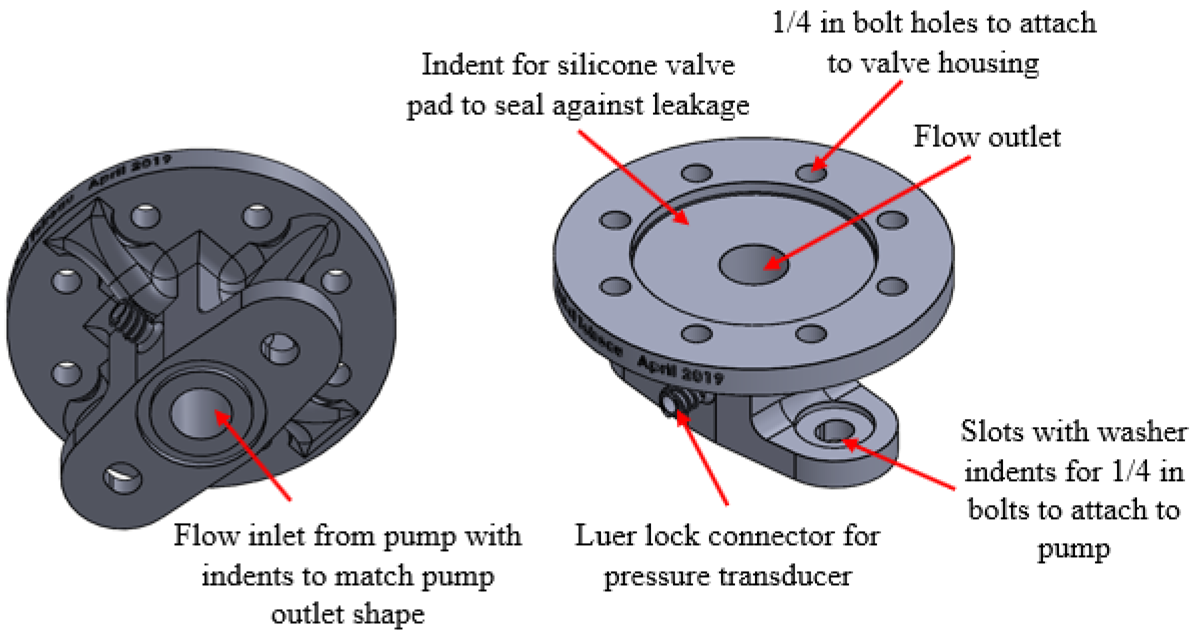

1/4 in bolt holes to attach to valve housing
Indent for silicone valve pad to seal against leakage
Flow outlet
April 2019
Slots with washer indents for 1/4 in bolts to attach to pump
Flow inlet from pump with indents to match pump outlet shape
Luer lock connector for pressure transducer

Figure 2.24: Annotated views of pump connection plate solid model

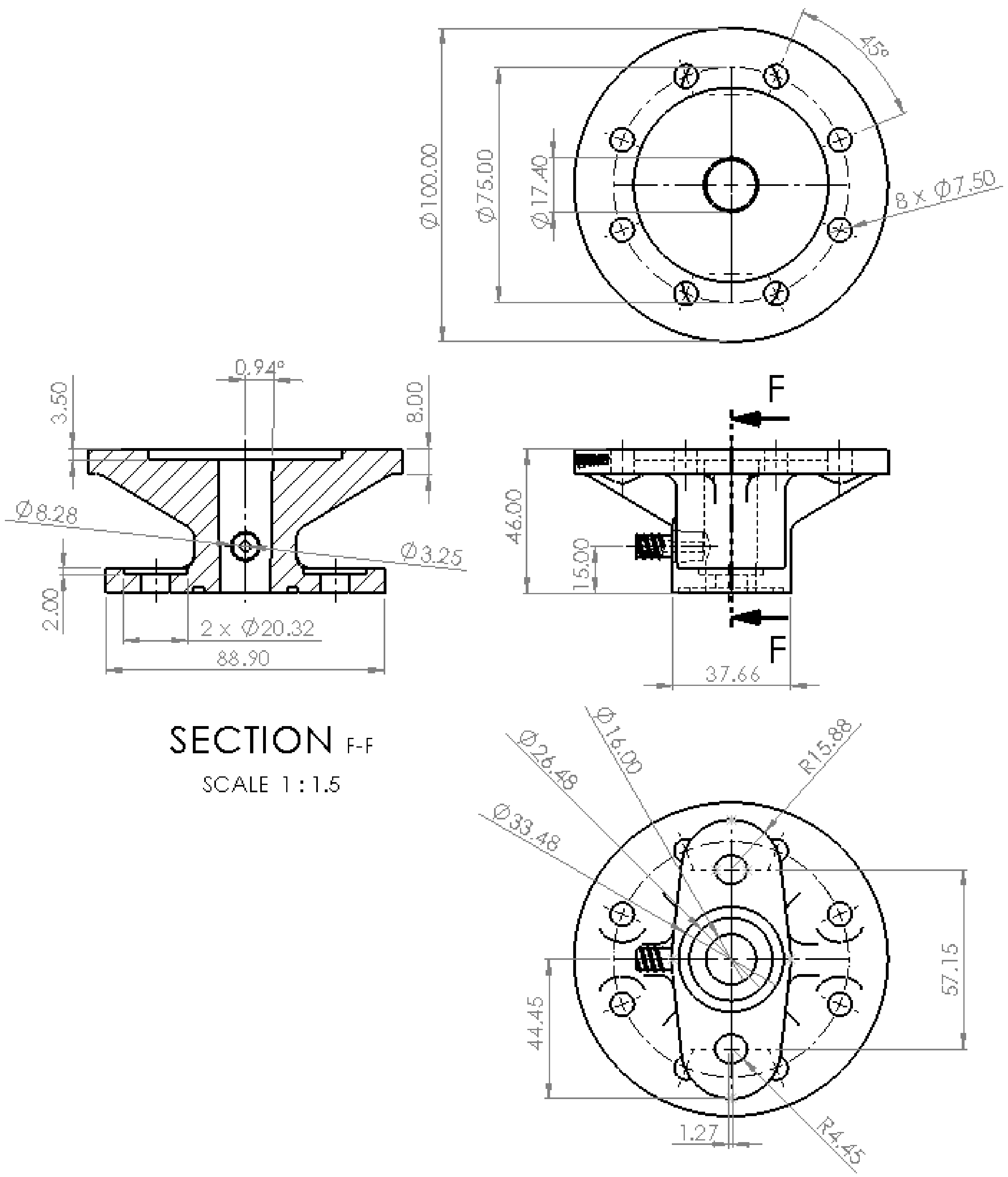


Figure 2.25: Dimensioned orthogonal engineering drawing views of pump connection plate

2.6. Reservoir and Backpressure Pump

To simulate the body conditions, a backpressure is required to simulate vascular afterload, providing resistance to the flow. The aortic valve also requires backpressure to close consistently. The hydrostatic head pressure was not significant enough with only a static reservoir located 1 m above the pulsatile pump. A centrifugal pump (Rotaflow Centrifugal Pump 20-970, Jostra AB) was used to aid the simulation of backpressure conditions of the human body. This is the same backpressure pump is used in the clinical EVHP system. This device pumps against flow direction while allowing fluid to slip through. The amount of backpressure used in experimentation is defined and characterized by the pump rotational speed in rpm. Five backpressure speeds from 0 rpm to 2000 rpm are used, increasing in increments of 500 rpm. This is converted to fluid pressure in kPa, corresponding to 0-15.66 kPa. The pressure at the pump inlet as the reference point, since it is located at the base of the flow loop.

2.7. Data Acquisition

Due to the need to coordinate the data timing from the pressure transducers (Edwards® Truwave Disposable Pressure Transducers, Edwards Lifesciences Corporation), high-speed camera (acA800-510um, Basler AG) images, camera trigger signal, and LVDT for the traverse/pump movement, a custom GUI was written in MATLAB for the data capture process. The code can be found in Appendix C.

Figure 2.26 is a schematic of the data collection set-up. All data is collected at a DAQ (USB 6009, National Instruments) at a capture rate of 6000 Hz. The DAQ can receive analog input (AI) signals from 8 channels numbered AI0-AI7. AI0 is connected to the function generator, which sends a square waveform signal at 200 Hz to the high-speed camera. Images are captured by the camera on the rising slope of the signal. This signal is also sent to the DAQ to coordinate timing of distension images with pressure signals. AI1 is unused. AI2 is connected to the LVDT signal, which records the pump displacement. AI3 is connected to the pressure transducer collecting data for air within the pressure-chamber. Pressure transducers located at the pump inlet, outlet, aorta inlet, outlet, and internal pressure-chamber are connected from AI4 to AI7 respectively. The pressure transducers have a working pressure range of -50 to 300 mmHg, or -6.7 to 40 kPa.

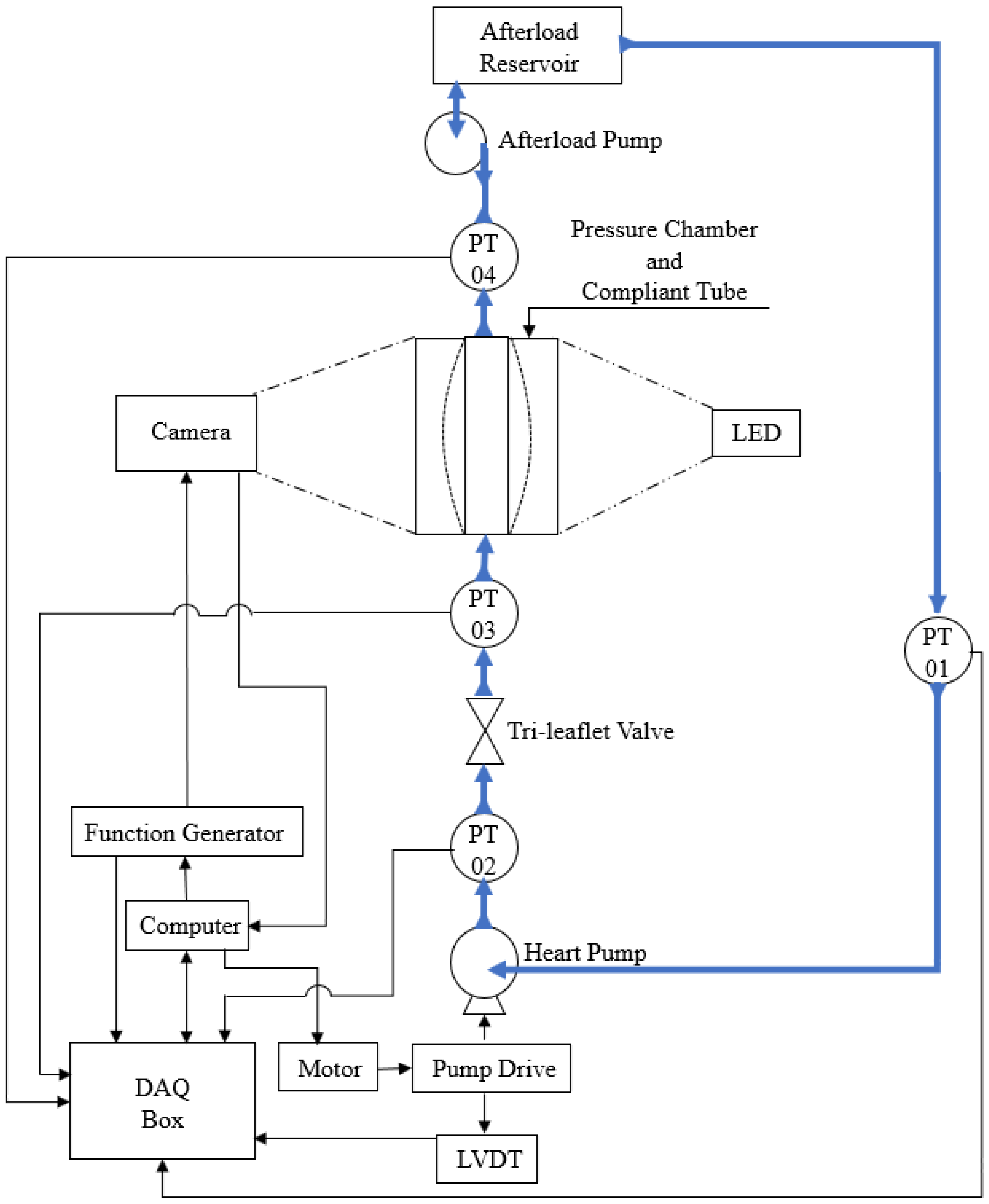


Figure 2.26: Schematic of data acquisition set-up

2.7.1. Timing

Timing is a key component of the data collection process. The images captured by the high-speed camera must be matched to the pressure data. Some images of the compliant tube not under the effects of the pulsatile cycle was needed as a comparison image for the distension images once the pulsatile cycle is active. Figure 2.27 below shows plots of the raw voltage data gathered by the DAQ over the entire data collection cycle, and Figure 2.28 shows a close-up of the digital 200 Hz signal used to trigger the camera cycle.

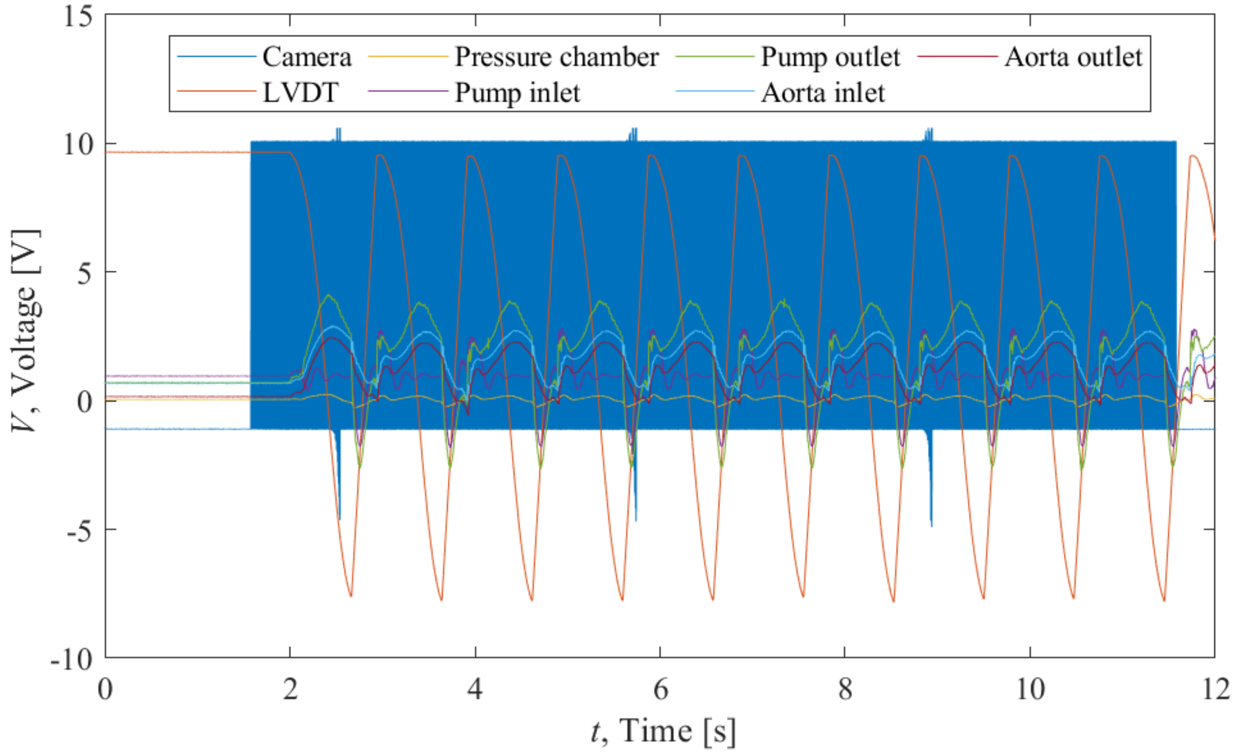


Figure 2.27: Plot of voltage data from DAQ for the complete data set

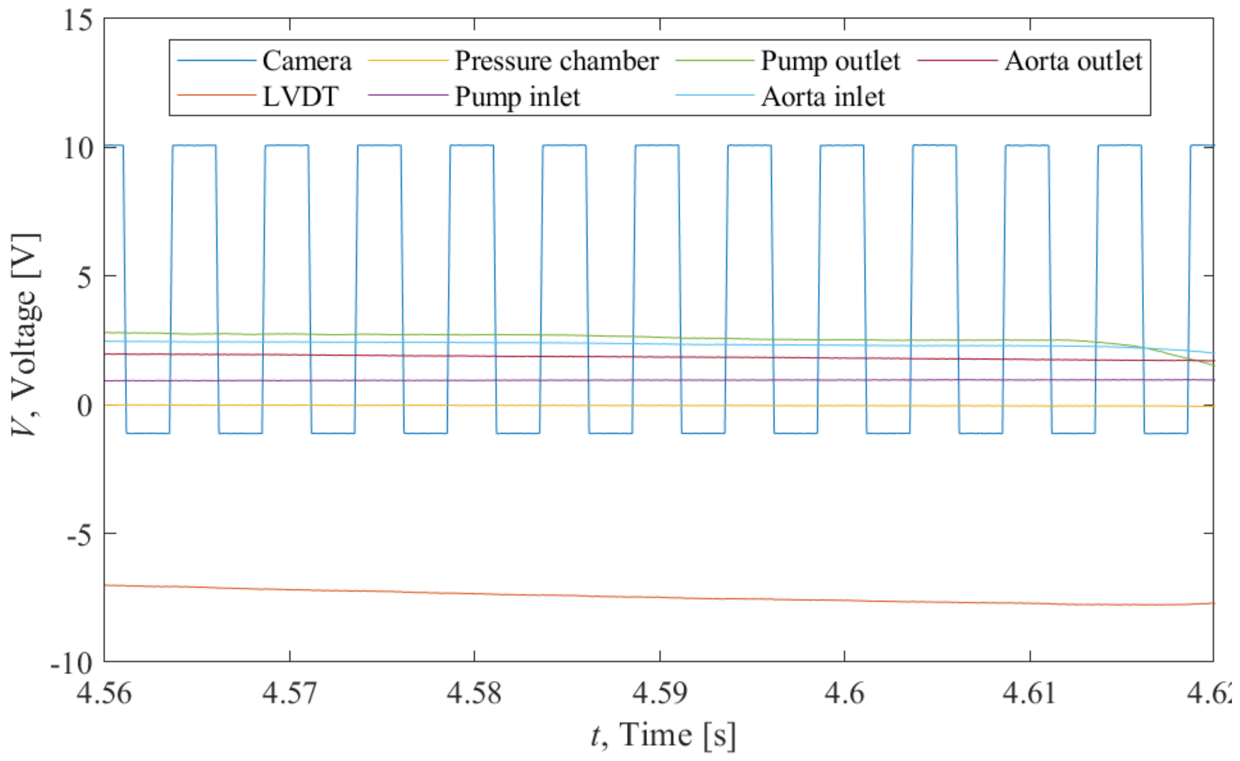


Figure 2.28: Plot of voltage data from DAQ zoomed in to show camera trigger function

Once the capture data button is pressed in the data gathering GUI, the counter starts at $t = 0$ s. The computer first sends a signal to the function generator to start generating the 200 Hz signal for the camera trigger. The signal is also captured by the DAQ. The initialization time for the MATLAB code, and the time it takes for the function generator to begin generating the data lasts approximately 1.5 s. At this time, images of the compliant tube not under pulsatile forces are gathered for 0.5 s. At approximately $t = 2.0$ s the MATLAB code sends out a signal to the motor to output the desired pulse frequency and stroke volume for the pump. The pump stroke distance is gathered by the LVDT attached to the traverse and the voltage signal sent to the DAQ. Once the pulsatile cycle begins, the DAQ gathers the voltage output from the pressure transducers located throughout the cycle. The DAQ gathers data until time $t = 12$ s. MATLAB then sends a signal to the DAQ to stop data collection and saves the captured data to a log file.

2.8. Limitations

2.8.1. System Vibrations

A large contributor to system vibration is the pump shaft's sudden change of direction at either end of its stroke. In some circumstances, the pressure-chamber visibly moves within the camera image frame. This may have led to larger variations in the tube diameter imaged by the high-speed camera.

2.8.2. Pressure Transducer Measurement Range

The pressure transducers (Edwards® Truwave Disposable Pressure Transducers, Edwards Lifesciences Corporation) have a working pressure range of -50 to 300 mmHg, or -6.7 to 40 kPa. In locations with high pressure spikes and drops, the measured pressure can reach values outside of the working pressure range. For example, during the valve closure event the measured pressure at the pump outlet can decrease to lower than -10 kPa. However, since the area of interest used in data processing is at the aorta inlet and outlet, the measured pressure at the pump outlet is used as a comparison only. Downstream of the valve (aorta inlet and outlet) the pressure variations are less dramatic and the majority stay within the working pressure range. For values which fall outside of the working range, extrapolation of calibration data is used.

2.8.3. Chamber-Pressure Fluctuations

Figure 2.29 below shows the chamber-pressure variations over the course of two cycles for the pressure regulator settings of 0 and 6.9 kPa. Due to the small internal volume, the chamber-pressure fluctuates with the changing volume of the compliant tube within. The fluctuating pressure will affect the distension of the experimental aorta. Overall, the chamber-pressure remains within a constant range of the pressure regulator settings and cycles according to aorta distension across pump cycles.

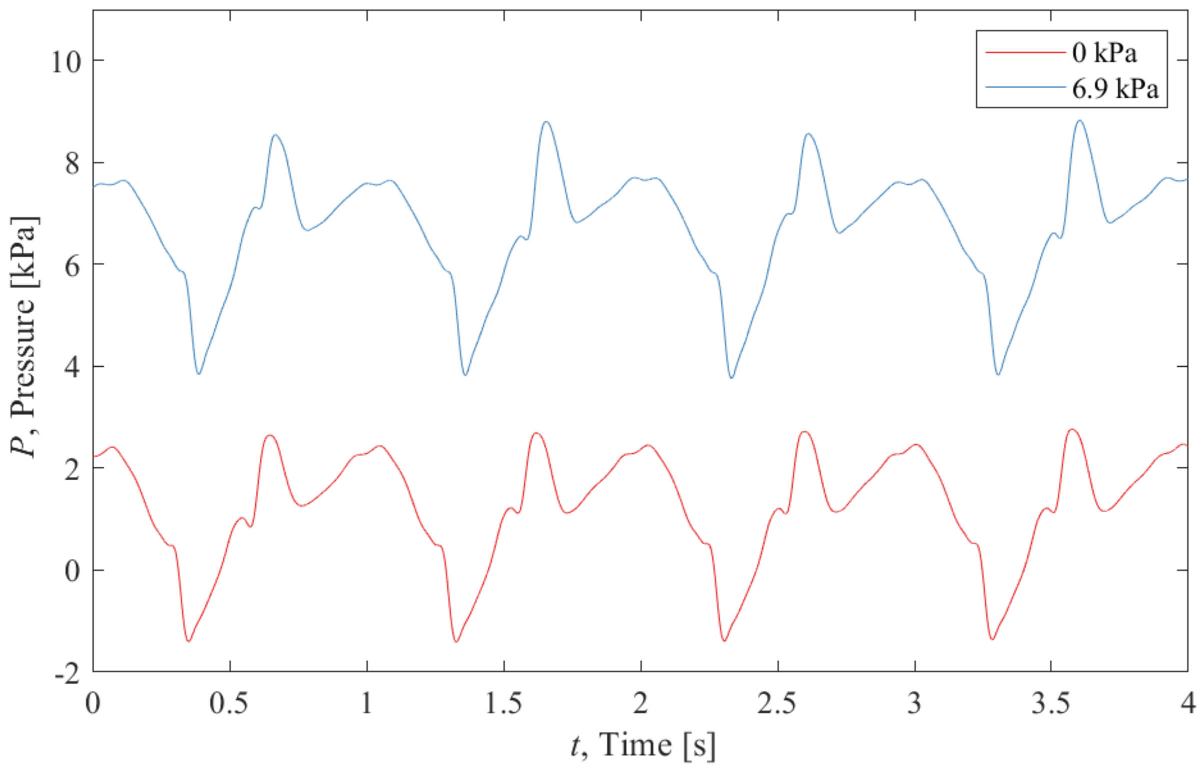


Figure 2.29: Pressure-chamber pressure variations throughout pump cycles for internal chamber pressure of 0 (red) and 6.9 (blue) kPa

2.8.4. Aorta Outlet pressure transducer location

As discussed in Section 2.5, the aorta outlet pressure transducer is located at the end of a 12 in long Delrin tube. Due to the need to seal the pressure-chamber against leakage while also allowing the clamping of different lengths of aortas within, the top end of the aorta is clamped to the end of the Delrin tube, which extends out of the top of the pressure-chamber. There is an O-ring located at the exit of the Delrin tube in the pressure-chamber cap to prevent air leakage. With this configuration, the aorta outlet pressure transducer could not be placed inside the pressure-chamber near the exit of the aorta. Instead, it is located outside at the exit of the Delrin tube. The difference in static pressure between the two locations is subtracted during data processing. However, there may be some losses due to entrance effects at each connection point.

2.8.5. Pulsatile Pump Pressure Waveform

Figure 2.30 shows the labelled diagrams of Murgo's Type pressure waveforms, A, B, and C [25], [26] in subfigures (a), (b), and (c) respectively. Subfigure (d) shows the experimental pressure waveform, while subfigure (e) shows the full experimental range. Some major differences stand out between the Murgo's waveforms and the experimental waveform. The systolic ejection time is significantly longer in the experimental waveform than the Murgo waveforms, and the diastolic relaxation time is shorter. The pressure range of the experimental waveform reaches approximately 0 mmHg during diastole, and higher than 150 mmHg during the systolic peak. The anacrotic notch is much more prominent in the experimental waveform than either Murgo's Types A or B. Given the significance of the systolic peak in comparison to the small reflected wave peak before the anacrotic notch, the experimental waveform resembles a Murgo's Type A. There is a small peak at the very end of diastole, which occurs due to the valve leaves flapping due to system vibrations.

Although the experimental pressure waveform differs from the desired pressure waveforms, the waveform itself is consistent after the initial start-up time of the pump at approximately 4 s onwards (subfigure (e)). Since the purpose of this investigation is to compare the effect of different experimental parameters on pump energy using the same pump, the consistency of the pump to output a pulsatile waveform resembling a heart waveform was deemed sufficient. Future work will include modifying the pump and pump drive section to produce a more accurate representation of a heart waveform.

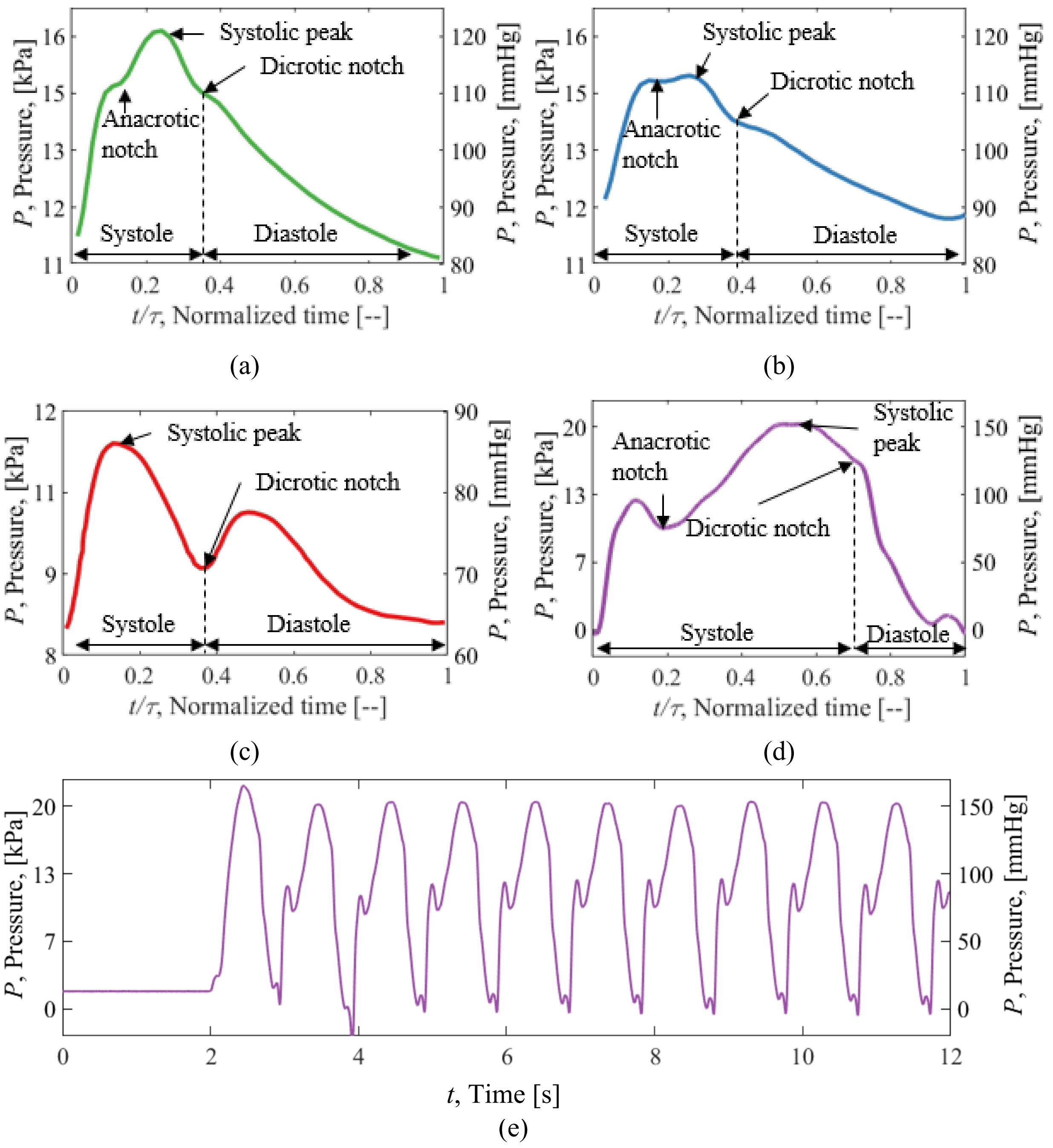


Figure 2.30: Labelled plots of three Murgo's Type waveforms, (a) Type A, (b) Type B, and Type C (c). (d) is a labelled plot of the experimental pressure waveform, with the full experimental range show in (e). Adapted from [25] for Types A and C, and [26] for Type B

2.9. Summary

This chapter outlines the experimental set-up for data acquisition. Each major components of the flow-loop contribute to the simulation of the left side of an EVHP system. The pump and its pump drive mechanism were designed and built in-house to replicate the human heart. A molded silicone tri-leaflet valve at the pump outlet was manufactured and used to prevent backflow during the pump suction stroke, functioning in a similar manner to the aortic valve in the human heart. A pressure-chamber was built to house compliant tubes of varying lengths and material in order to test varying compliance parameters on the pulsatile system. The chamber pressure controls the degree of distension for each compliant tube. Varying backpressure conditions are introduced to the system using a centrifugal pump to generate flow resistance. The varied system parameters of pump pulse frequency, pump stroke volume, compliant tube, compliance chamber pressure, and backpressure are compared to determine the overall effect on pump performance. Data acquisition include pressure data at select flow loop locations, high-speed imaging to determine compliant tube distension, and pump stroke distance to determine volumetric flow rate. The collected data are processed in Chapter 3 and used to calculated pump energy. The results of varying system parameters on pump performance are discussed in Chapter 4.

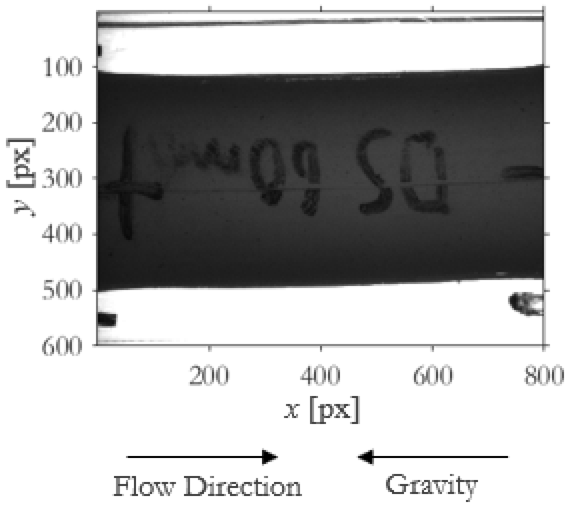
Chapter 3. Data Processing Methodology

3.1. Introduction

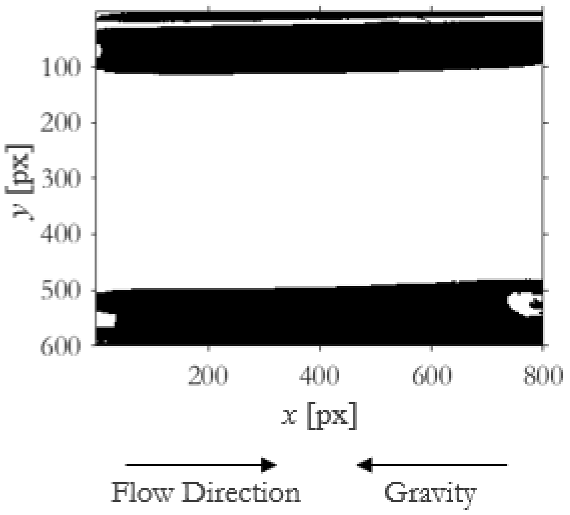
Tube distension, pump stroke distance, and pressure were the three experimental conditions monitored during experimental testing. An in-house code (MATLAB, The MathWorks Inc.) was used to process high-speed camera images of the tube and determine tube expansion with respect to time. Pressure data was filtered after acquisition to remove noise. Pump stroke distance was measured using a LVDT and converted to the system volumetric flow rate. Pump energy was calculated using the processed pump stroke distance and pressure results. The equations for pump energy are adapted from previous work undertaken by Cameron [15].

3.2. Processing of Imaging Data

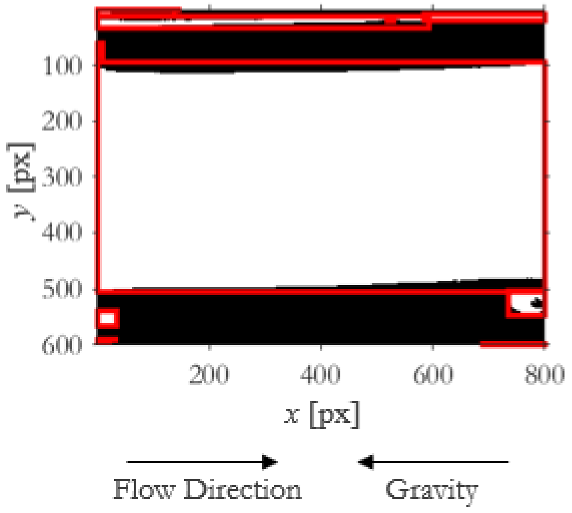
Tube distension was calculated using images captured at 200 Hz by a high-speed camera (acA800-510um, Basler Inc.). An in-house MATLAB code (provided in Appendix G) is used to process the images. Figure 3.1(a) shows an image of the original camera data captured of the tube with the fluid at rest. The tube is backlit to enhance the contrast between the tube edges and the background. There is extraneous information within the original image, such as the edges of the pressure-chamber. This must be removed before edge detection can be used on the tube. Figure 3.1(b) shows the binarized image, which maximizes the contrast at edges. Figure 3.1(c) shows an image after MATLAB edge detection is applied to find the bounding boxes of each object within the image. Figure 3.1(d) shows the final cropped image used for determining tube distension.



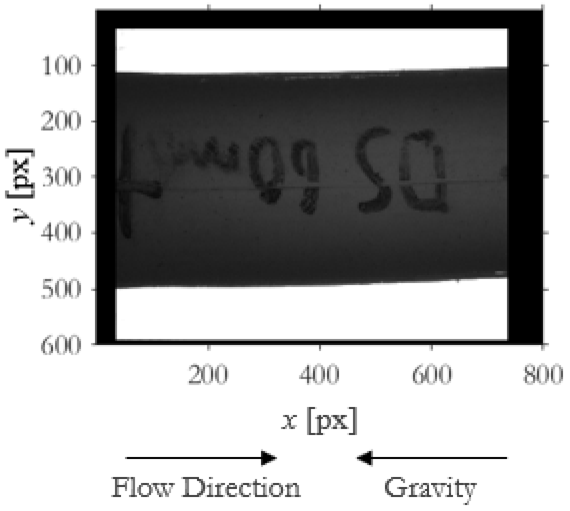
(a)



(b)



(c)



(d)

Figure 3.1: Image processing steps starting from (a) initial image, (b) binarized image, (c) edge detection to find regions to crop, and (d) cropped image

A peak intensity detection code was then used on the cropped image to determine the tube edge locations in *y* across length *x* of the tube for every image collected during the experiment run time. Figure 3.2 shows an example of image intensity gradient *dI* as a function of tube width *y*. The red boxes highlight the location of maximum peak intensity, where the tube edges are found. Once the location of the two edges of the tube are found, the width of the tube along its entire length *x* for every image captured is calculated. The tube width is then saved in a 3-dimensional matrix as a function of both time and its position along the length *x* of the tube

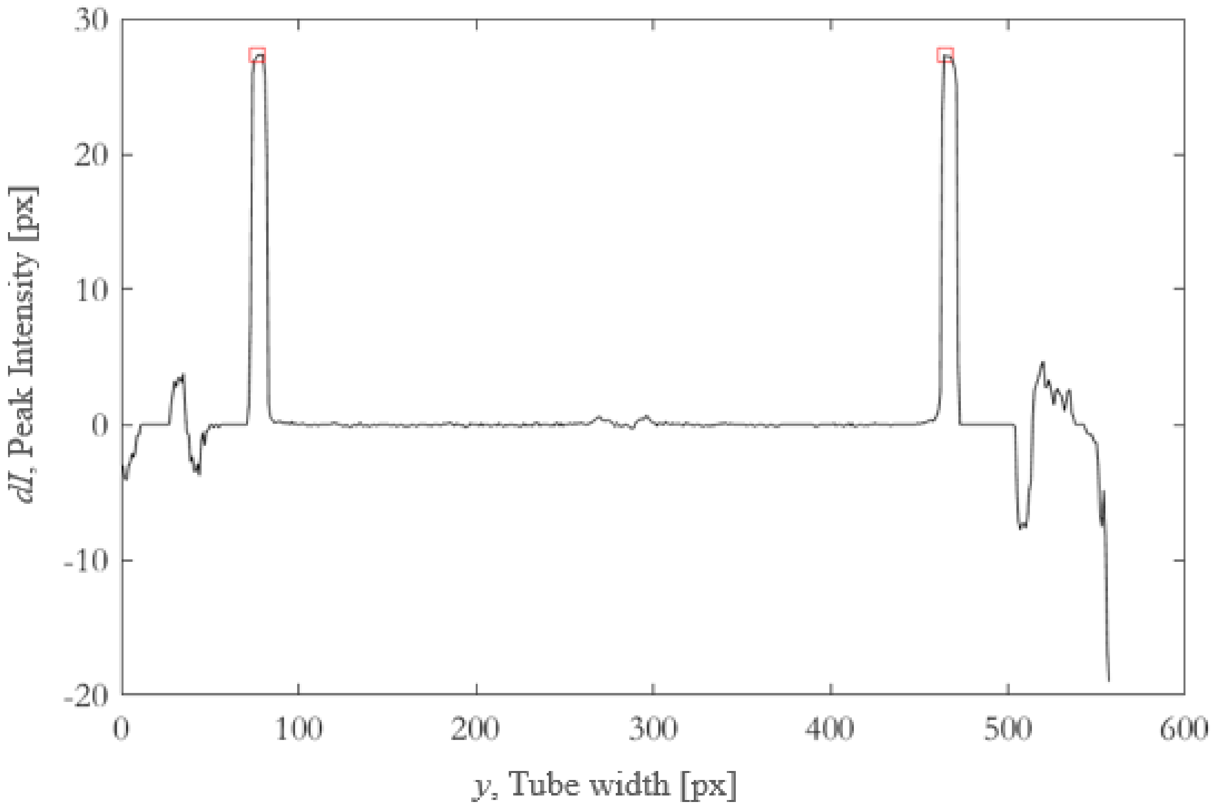


Figure 3.2: Plot of peak intensity locations along one tube vertical pixel axis for one image

3.2.1. Aorta Percent Diameter Distension, $\frac{\Delta D}{D}$

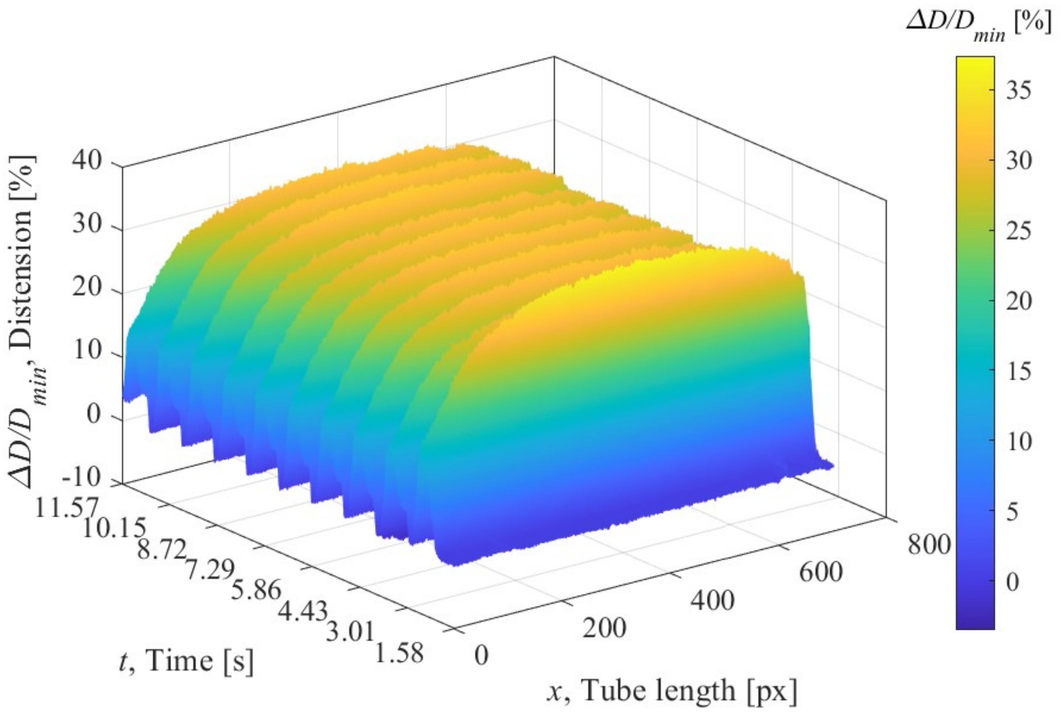
To determine the amount of diameter distension the aorta undergoes, the percent diameter distension, $\frac{\Delta D}{D}$ [%], is found. Equation 3.1 shows the calculation for $\frac{\Delta D}{D}$:

$$\frac{\Delta D}{D} = \frac{D(t) - D_{min}}{D_{min}} \times 100\% \qquad 3.1$$

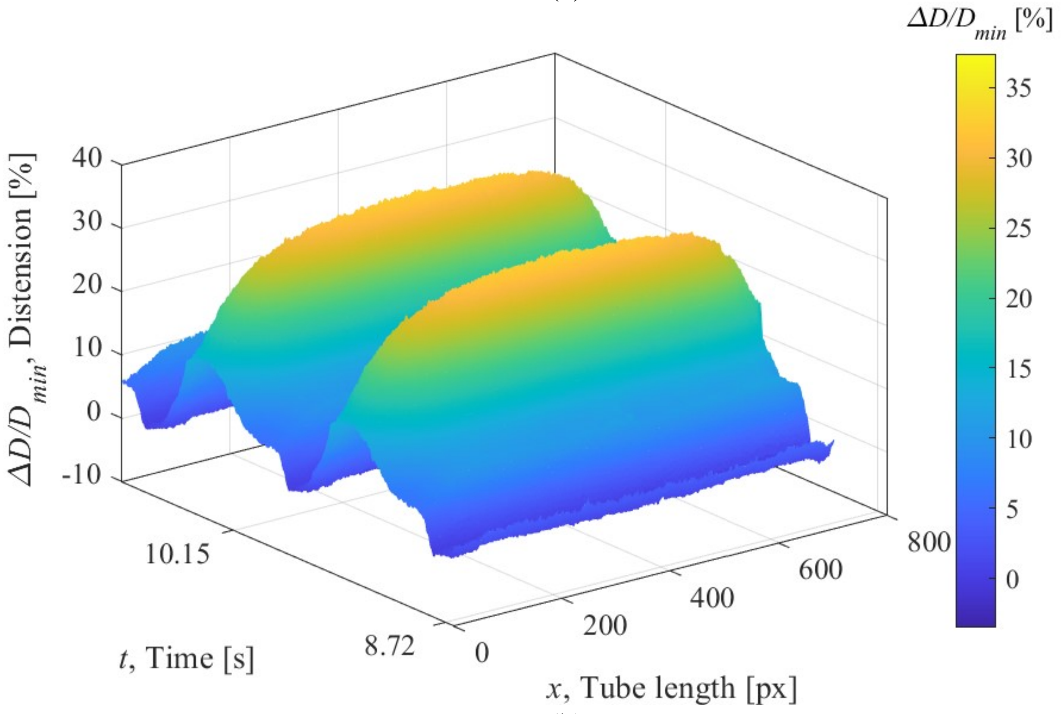
where:

$\frac{\Delta D}{D}$	=	Diameter distension in percent
$D(t)$	=	Aorta diameter at camera time t
D_{min}	=	Minimum aorta diameter, averaged over 8 cycles.

When the experimental system is at rest, the mock aorta collapses under high chamber-pressure. Once the system is running, the mock aorta retracts back to a non-collapsed minimum diameter within a certain chamber-pressure range. The mock aorta once again collapses when the chamber-pressure reaches a higher threshold. Since a collapsed mock aorta during system rest is not a good indicator of the mock aorta minimum diameter, the percent diameter distension is calculated using the minimum mock aorta diameter averaged over 8 cycles. Once $\frac{\Delta D}{D}$ is calculated, this is saved in a 3-dimensional matrix as a function of both time and its position along the length *x* of the tube. The MATLAB code to calculate minimum aorta diameter is listed in Appendix H. Figure 3.3 shows the 3-dimensional surface plot of $\frac{\Delta D}{D}$ for (a) one full experimental case, and (b) over two cycles. The mock aorta distension is greater and more consistent along the center of the tube. The distension is lower at each end of the mock aorta due to the clamping mechanism on each end. Figure 3.4 shows the plot of $\frac{\Delta D}{D}$ against time at the 400 px strip down the width of the mock aorta. In the initial phase of pump ejection, the mock aorta diameter expands sharply before hitting a small plateau, then increases to the peak of expansion. Following the pump return stroke, the mock aorta diameter relaxes to its minimum diameter before the cycle repeats.

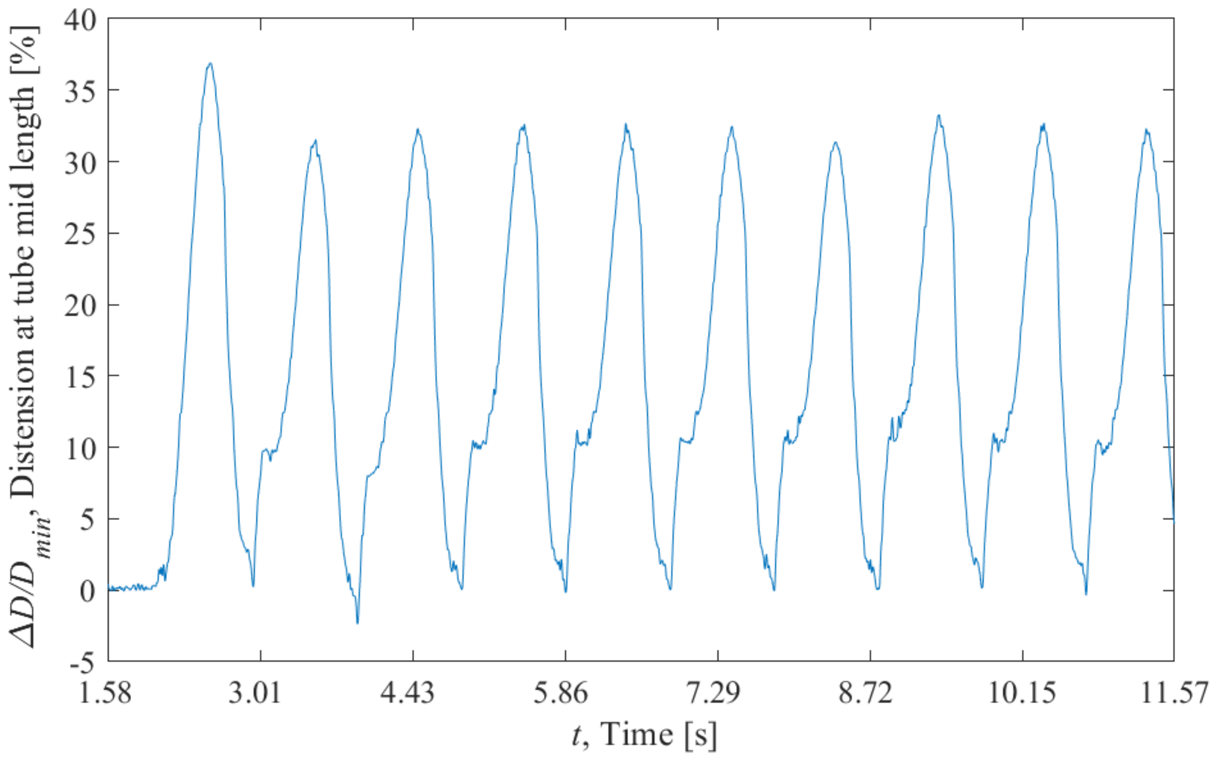


(a)

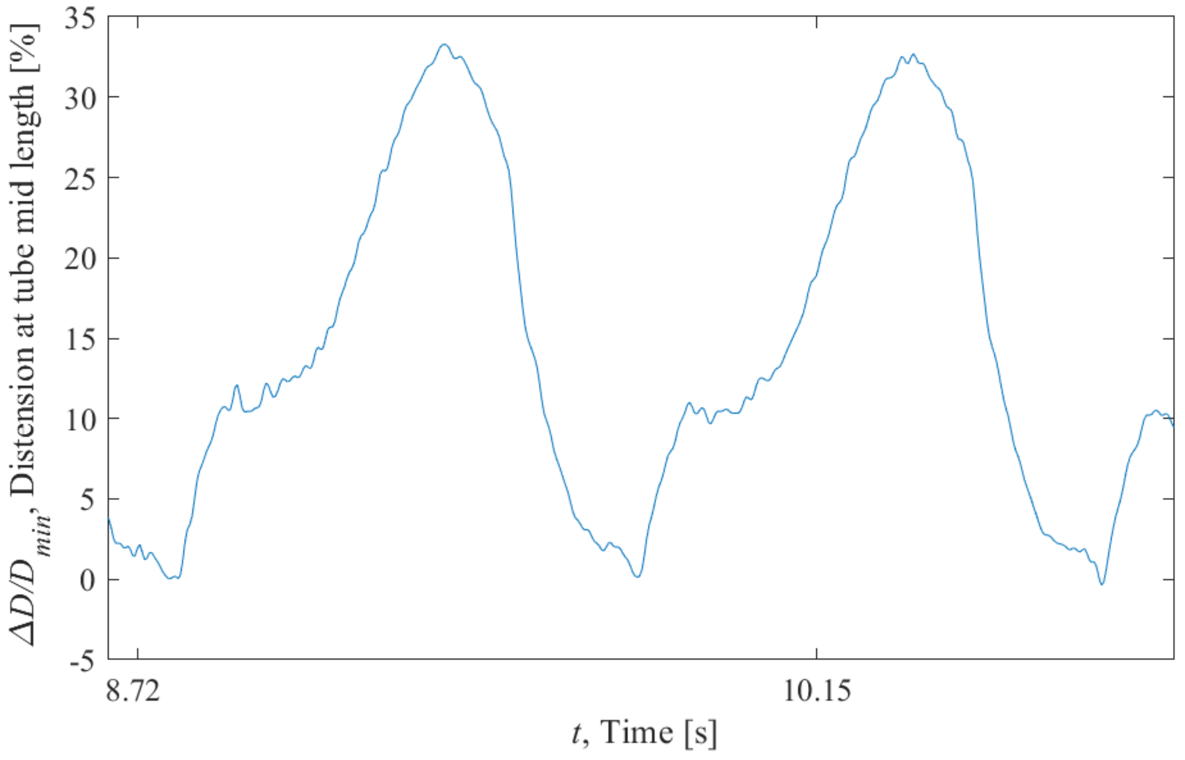


(b)

Figure 3.3: Plots of tube distension over time shown as 3D mesh plots of (a) full experimental time and (b) zoomed in to show 2 cycles



(a)



(b)

Figure 3.4: Plots of tube distension over time at tube mid length of (a) full experimental time and (b) zoomed in to show 2 cycles

3.3. Processing of Pressure Data

As discussed in Chapter 2, pressure data was gathered using pressure transducers (Edwards® Truwave Disposable Pressure Transducers, Edwards Lifesciences Corporation) at the following four locations: pump inlet, pump outlet, aorta inlet, and aorta outlet (see Figure 2.1 in Chapter 2). As discussed in Section 2.7, the voltage data from the pressure transducers were collected using the DAQ for 12 seconds at 6,000 Hz. The pressure transducers have a working pressure range of -50 to 300 mmHg, or -6.7 to 40 kPa.

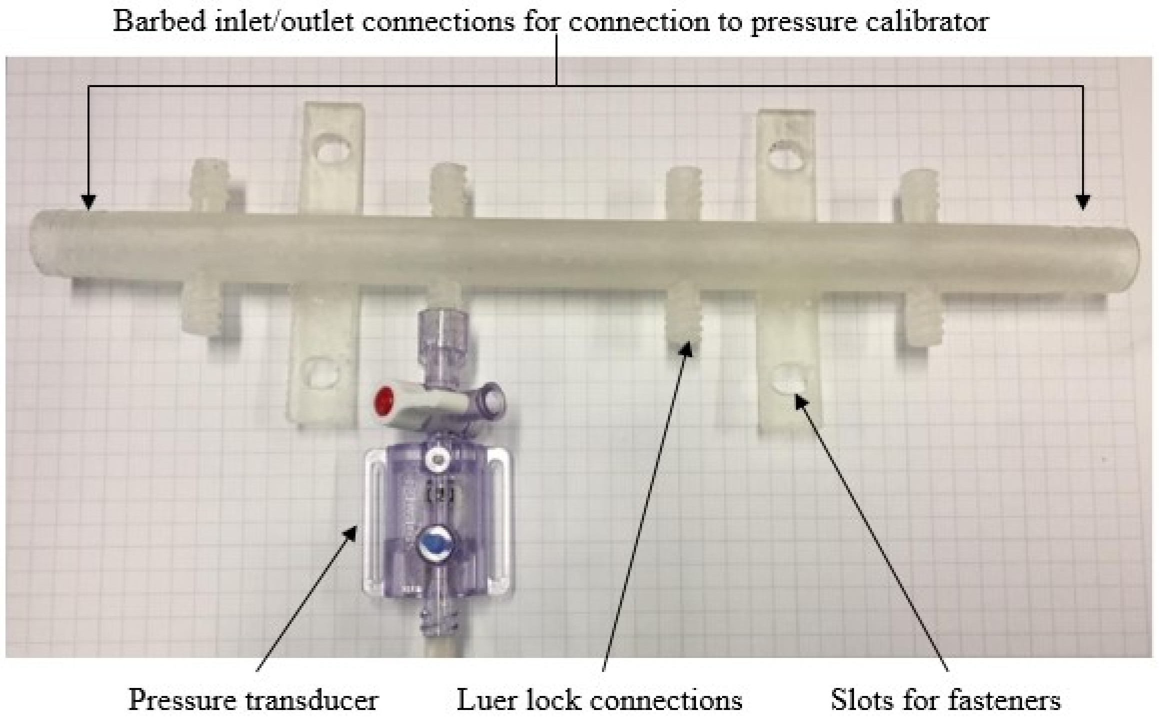

Barbed inlet/outlet connections for connection to pressure calibrator
Pressure transducer
Luer lock connections
Slots for fasteners

Figure 3.5: Pressure transducer calibration device with pressure transducer attached

An in-house 3D printed device was used to determine the calibration values for the pressure transducers. An image of the calibration device and set-up is shown in Figure 3.5. Multiple Luer lock connections ensure all four transducers used in the experiments can be calibrated at the same time. A pressure calibrator (DPI 603, GE Druck) was used to supply pressure from 0 kPa to 40 kPa in increments of 5 kPa. The voltage data was collected for 5 seconds and averaged. The pressure as a function of voltage plots are shown in Figure 3.6. A linear relationship was found, and the derived equations for slope and intercept used to convert the voltage values to pressure are listed in the sub-figures of Figure 3.6 for the different individual pressure transducers. The code used to convert the voltage values to pressure can be found in Appendix D.

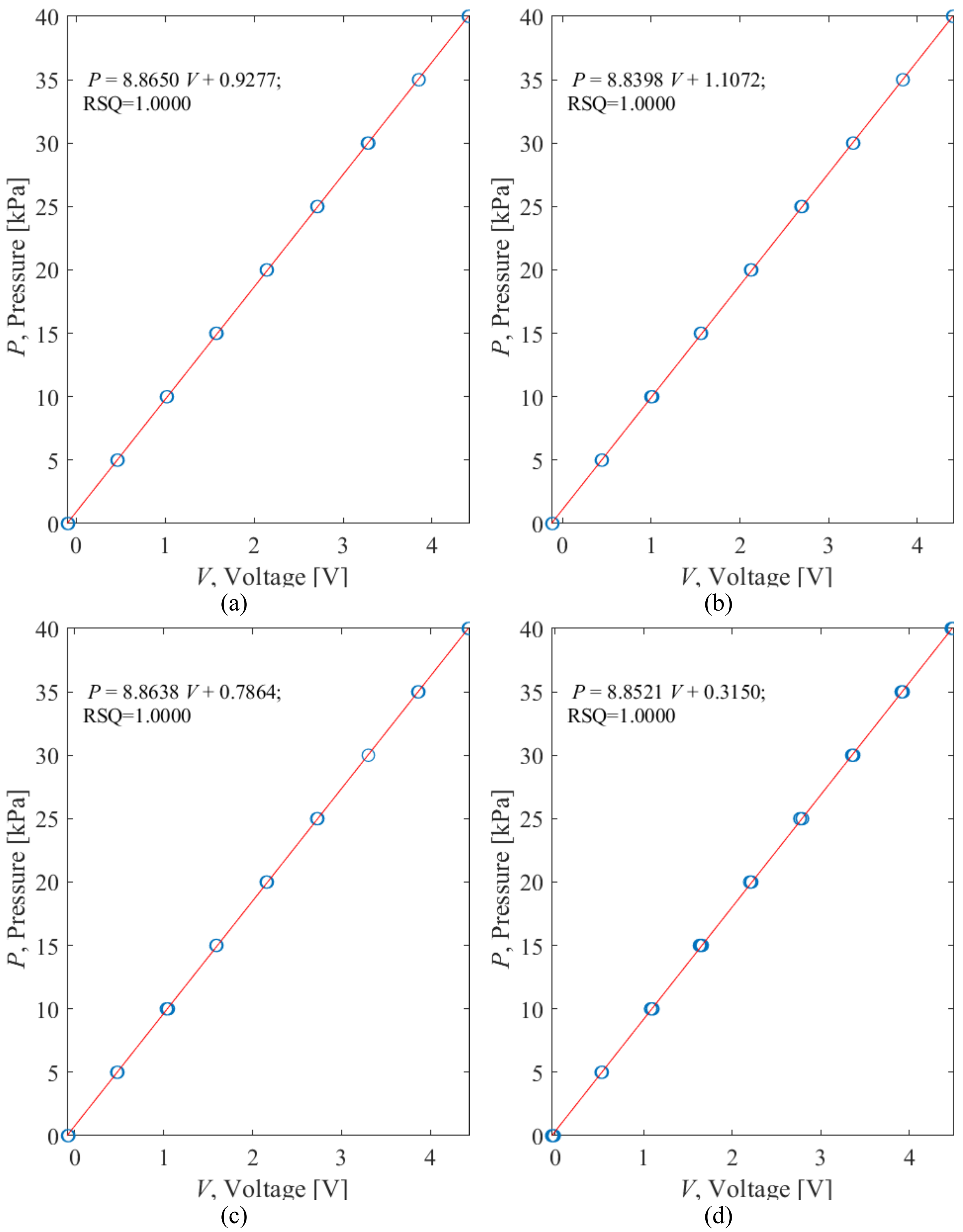
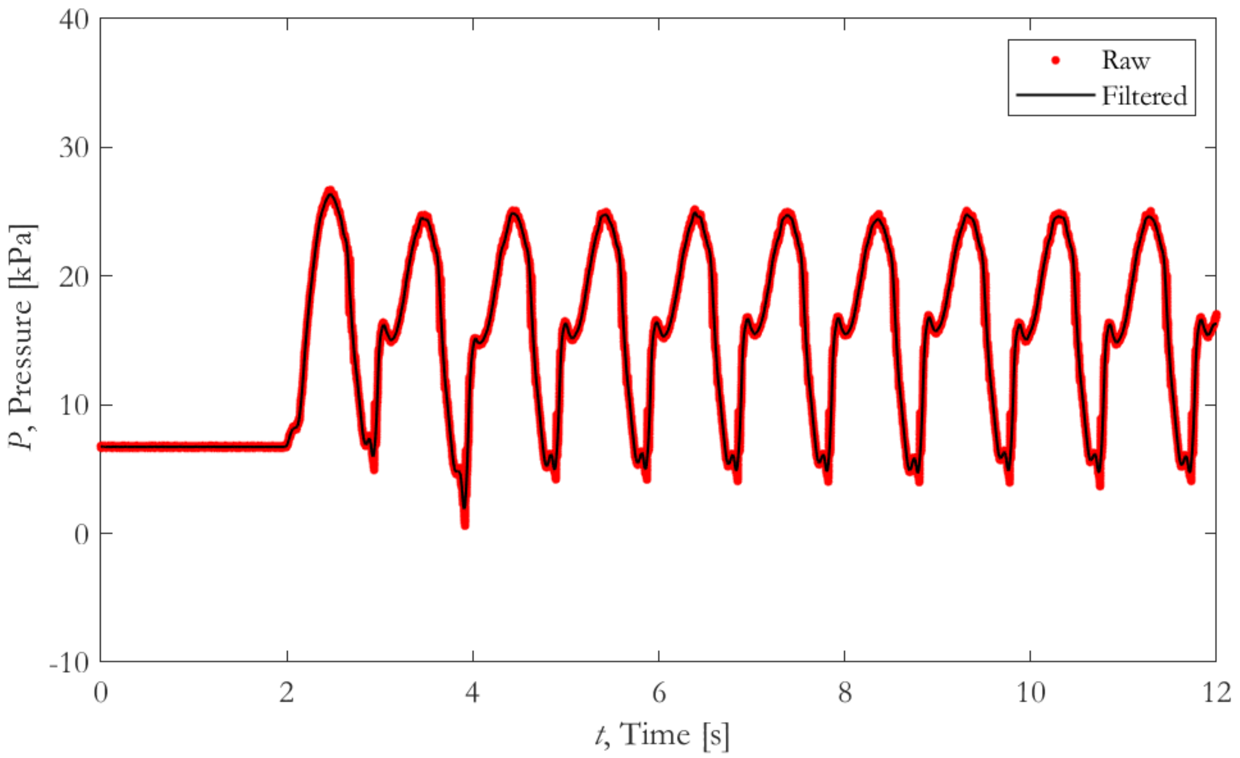
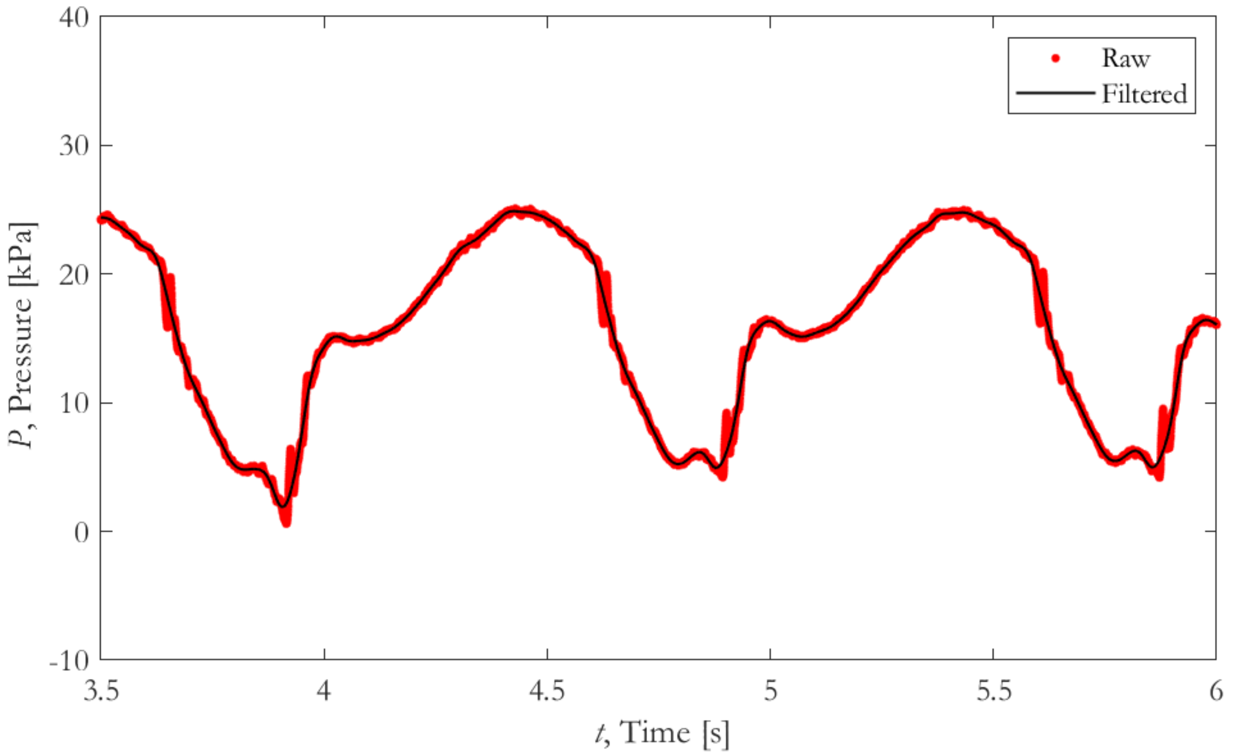


Figure 3.6: Pressure transducer calibration data for transducer located at (a) pump inlet, (b) pump outlet, (c) aorta inlet, (d) aorta outlet

To remove noise from the pressure data, a zero-phase filtering technique written in MATLAB was used to filter the pressure waveforms and provided in Appendix H. A 6th order Butterworth filter with a 20 Hz cutoff frequency gave the closest representation of the pressure response. Figure 3.7 shows the filtered waveform plotted against the raw data points at the aorta inlet for the entire 12 second data collection time range in Figure 3.7(a), as well as zoomed in for two cycles in Figure 3.7(b). Due to the high sampling rate of data collection, the filtered pressure waveforms are expressed as continuous lines.



(a)

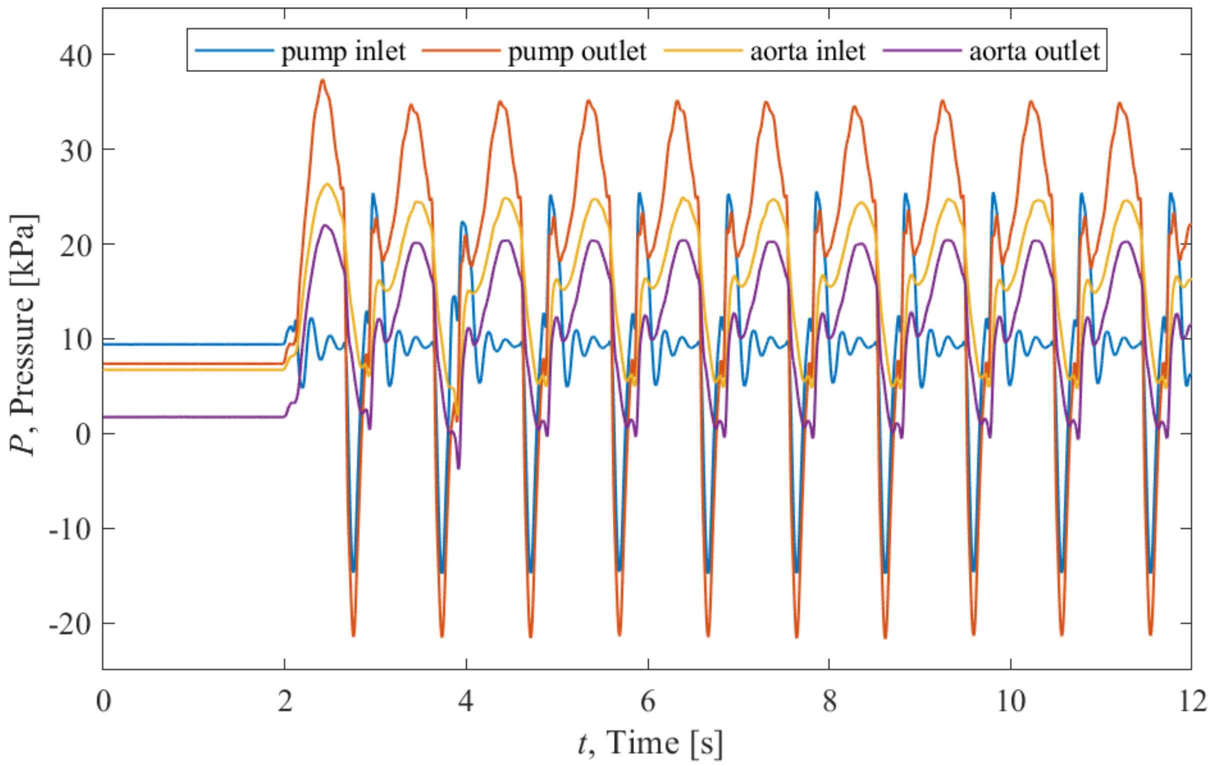


(b)

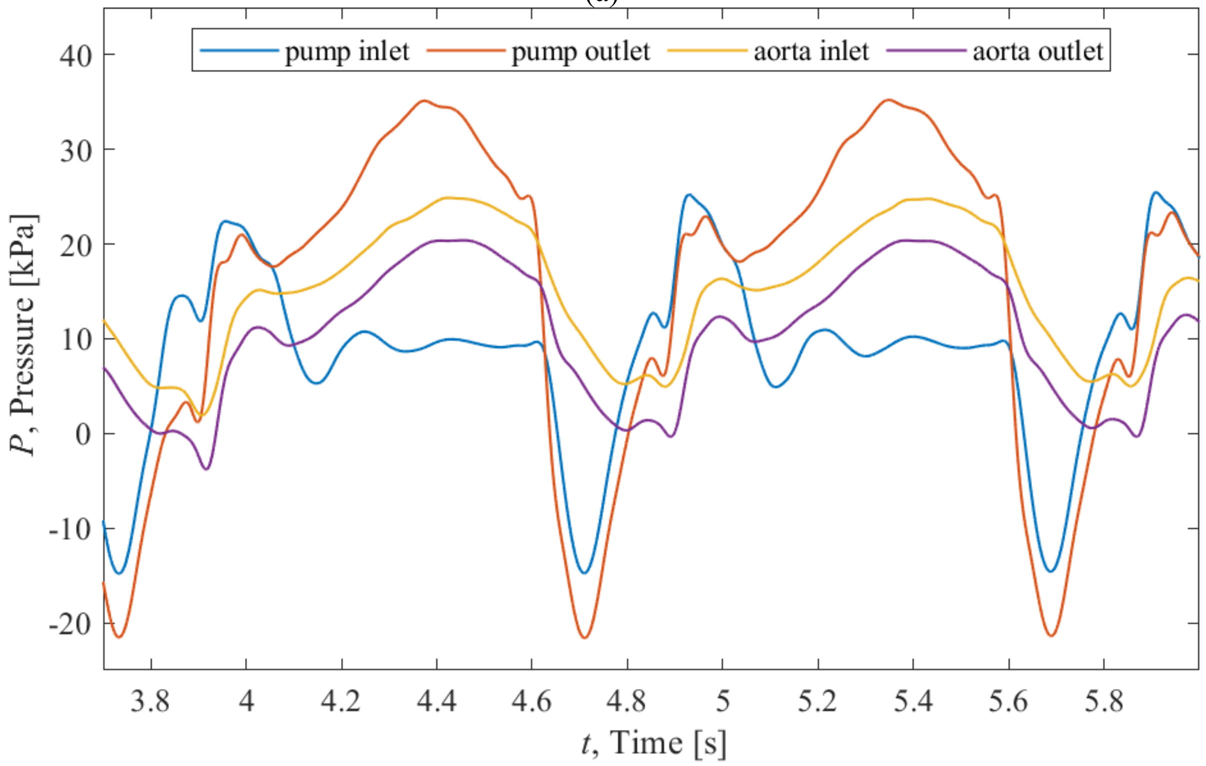
Figure 3.7: Plot of raw vis filtered pressure at the aorta inlet over (a) the full experimental time range, and (b) two cycles

Figure 3.8 shows pressure measurements at all four transducer locations over the full experimental time range in sub-figure (a) and zoomed for two cycles in sub-figure (b). The initial readings up to time *t* = 2 s are the static pressure values before the pump is turned on. The main beaks and valleys of the waveform are visible in at all four locations. The further downstream, the more dampened the waveform intensity becomes. The pressure pulse can be seen to move through the system with respect to time. There is a noticeable phase difference in the waveforms when looking at the occurrences of the major peaks and valleys. At the pump inlet the largest valley occurs approximately 0.1s before the same valley is reached at the aorta outlet. The same phase shift can be found for the first and second major peaks of the waveforms as well.

During valve closure, a negative pressure below the -6.7 kPa lower limit of the pressure transducers is reached. This is most prominently seen at the pump inlet and outlets. Given the consistency of the pressure readings across all cycles, as shown in Figure 3.8 (a), it is assumed the linear transducer calibration relationship is sufficient for determining general trends and energy change across experiments.



(a)



(b)

Figure 3.8: Plot of filtered pressure at all flow loop locations over (a) the full experimental time range and (b) zoomed in for two pump cycles

3.3.1. Pressure Unit Conversions

There were three different pressure units in use within the system. Flow pressure measured in kPa, system backpressure controlled via centrifugal pump rotation speed in rpm, and chamber-pressure controlled by a pressure controller in psi. These are converted the SI unit kilopascals (kPa) for consistency. It will also be useful to know these conversions in millimeters mercury (mmHg), which is a common unit of measurement for pressure in the medical field. Table 3.1 and Table 3.2 show the original pressure measurement units and their corresponding values in kPa and mmHg.

Table 3.1: Chamber-pressure unit conversions

Original unit: psi	kPa	mmHg
0.0	0.0	0.0
0.5	3.4	25.9
1.0	6.9	51.7
1.5	10.3	77.6
2.0	13.8	103.4
2.5	17.2	129.3
3.0	20.7	155.2
3.5	24.1	181.0
4.0	27.6	206.9
4.5	31.0	232.7
5.0	34.5	258.6

Table 3.2: Backpressure unit conversions

Original unit: rpm	Pressure difference between pump inlet and mock aorta outlet	
	kPa	mmHg
0	0.00 ± 0.0415	0.00
500	0.88 ± 0.0447	6.59
1000	3.71 ± 0.0437	27.84
1500	8.62 ± 0.0433	64.63
2000	15.66 ± 0.0462	117.43

As discussed in Section 2.6, the backpressure of the system is driven by a centrifugal pump that pumps against flow direction and allows some fluid to slip through. The speed of the pump in rpm sets the backpressure of the system. To convert the pump rotational speed to its corresponding pressure, the pressure transducer measurements when the flow loop is inactive is used. The system is in resting state at the specified centrifugal pump speeds for the first second of data capture before the motor is turned on. The pressure difference at the mock aorta inlet from the pump inlet is found and calculated in both kPa and mmHg. Pump inlet is the reference point, since it is located at the base of the flow loop. All other flow-loop components downstream of the inlet are of a higher elevation than the inlet. Therefore, all static pressure measurements downstream of the pump inlet will be lower than the pump inlet pressure when the backpressure pump is off. Taking the difference between the mock aorta inlet and pump inlet pressure will result in a negative pressure value when the backpressure pump is off. Increasing the rotation speed of the backpressure pump will increase the static pressure measurements at all measurement locations of the flow-loop, with the exception of the pump inlet. Since the trileaflet valve separates the pump inlet location from the downstream components of the flow-loop, the only pressure forces acting on the pump inlet location is the hydrostatic pressure head from the fluid reservoir, located approximately 1 m above the pump. Due to this, the backpressure kilopascal conversions are adjusted accordingly, setting the backpressure conditions to 0 kPa when the centrifugal pump rotation is 0 rpm.

Five thousand pressure data points at the pump inlet and mock aorta outlet are gathered during the system resting state, and the mean of their difference is found. Figure 3.9 shows the plot of fluid pressure in both kPa and mmHg against the centrifugal pump speed in rpm. A polynomial fit for the conversion is found and plotted in blue. Error bars are show on the plot at two standard deviations from the mean. The MATLAB code for the calibration can be found in Appendix G.

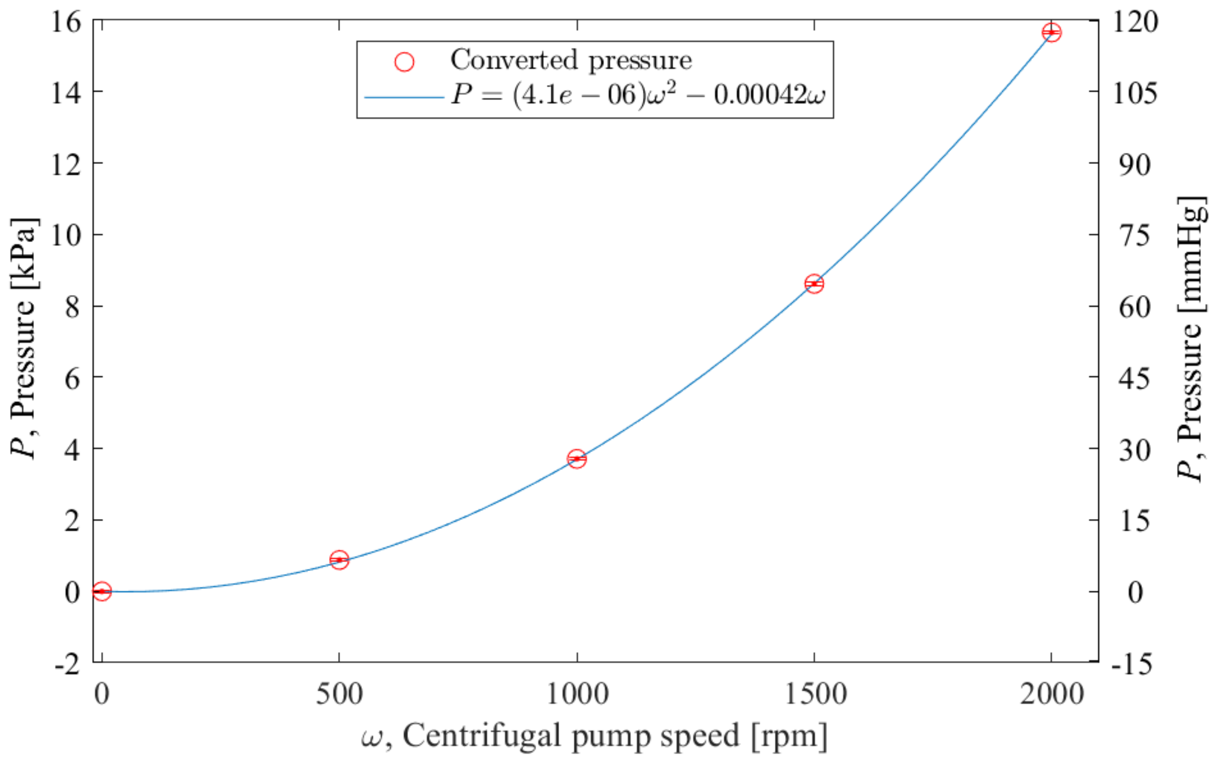


Figure 3.9: Plot of fluid pressure in both kPa on the left y-axis and mmHg on the right y-axis vs centrifugal pump speed in rpm. A polynomial fit for the conversion is found and plotted in blue. Error bars are show on the plot at two standard deviations from the mean.

3.4. Processing of Pump Stroke Distance

Since the pump drive shaft is directly coupled to the LVDT on the linear traverse, as described in Section 2.3, a 1:1 relationship of LVDT voltage measurement to pump drive shaft and diaphragm distance traveled is used. Figure 3.10 shows a plot of LVDT voltage and displacement calibration. An increase in LVDT displacement corresponds directly to an increase in pump fluid displacement. The pump forward stroke, systole in the cardiac cycle, is marked by a linear increase in LVDT travel distance. The pump return stroke, diastole in the cardiac cycle, is marked by the non-linear decrease in LVDT travel distance, accelerating in the beginning and decelerating at the end of the return stroke. Using the displacement distance x, $\frac{dx}{dt}$ is found and used to calculate volumetric flow rate, $\dot{Q}(t)$, used in the calculation for pump energy in Section 3.5.

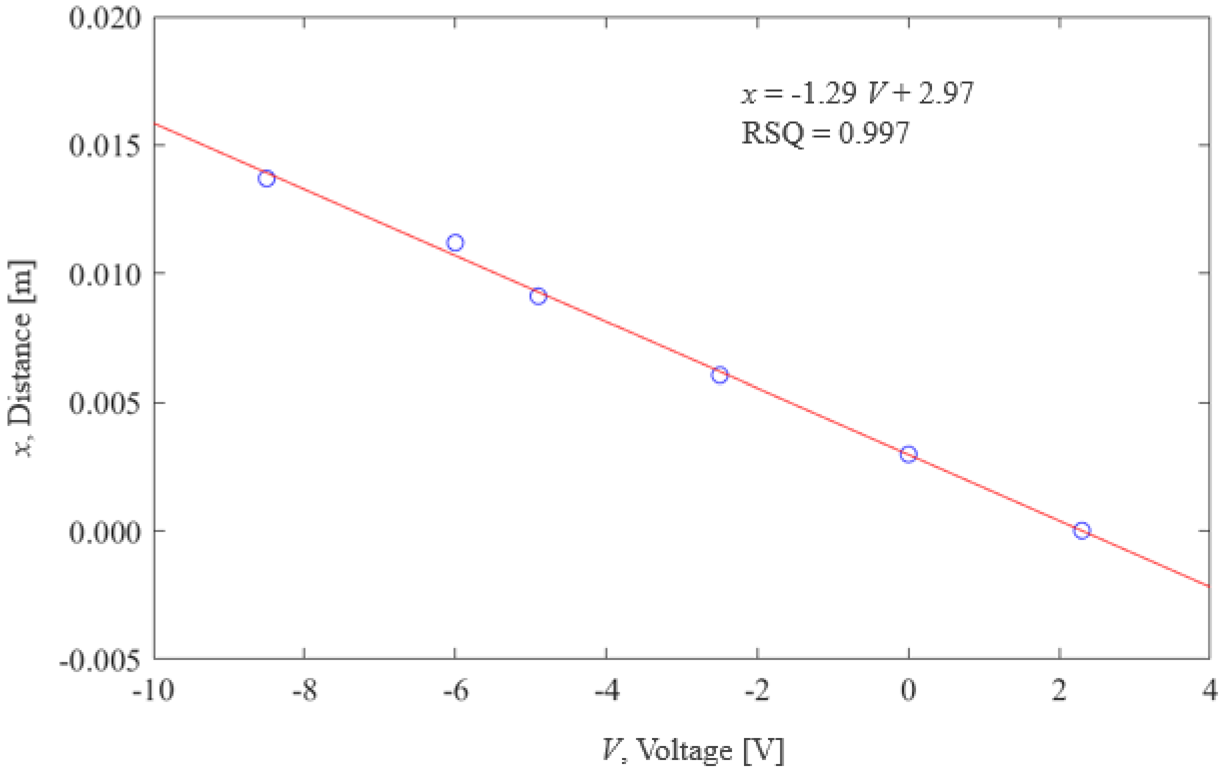


Figure 3.10: Plot of LVDT voltage and displacement calibration

The pump stroke travel distance was calculated using LVDT voltage data filtered using a 6th order Butterworth filter to remove noise. Figure 3.11 shows the raw vs filtered plot of pump stroke travel distance against time for three pump cycles. The filtered LVDT data is used for the calculation of volumetric flow rate, $\dot{Q}(t)$, in Section 3.5.

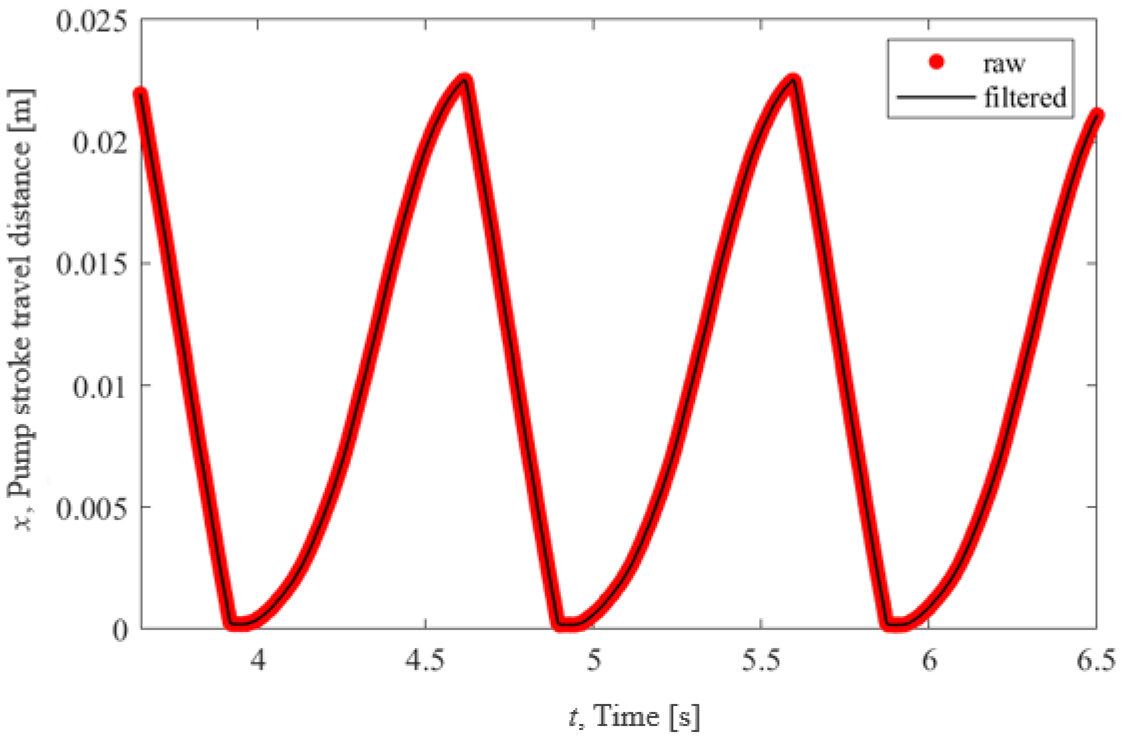


Figure 3.11: Plot of raw vs filtered pump stroke distance against time for three pump cycles

3.5. Volumetric Flow Rate, $\dot{\boldsymbol{Q}}(\boldsymbol{t})$

Q_{cycle} is found by averaging the total volume of water displaced by the pump over 10 cycles. Water is collected by running the pump at the specified pulse frequency for 10 cycles. The collected water is weighed and averaged over the number of cycles. Table 3.3 lists the Q_{cycle} values for each experimental pulse frequency case. Assuming the diaphragm pump area to remain constant, a linear relationship of $\frac{Q_{cycle}}{x_{cycle}}$ is used. The volumetric flow rate $\dot{Q}(t)$ can be determined at each point in the cycle, knowing the change in x with respect to time, $\frac{dx}{dt}$, measured by the LVDT and calculated in Section 3.4.

$$\dot{Q}(t) = \frac{Q_{cycle}}{x_{cycle}} \frac{dx}{dt} \tag{3.2}$$

where:

Q_{cycle}	=	Maximum volumetric flow per cycle [m^3/cycle]
x_{cycle}	=	Maximum pump displacement per cycle [m]
x	=	Pump displacement [m]

Table 3.3: Q_{cycle}, maximum stroke volume per cycle at each bpm

Pulse frequency [bpm]	Q_{cycle} [m^3/cycle]
60	73.6×10^{-6}
80	72.5×10^{-6}
100	66.5×10^{-6}
150	45.4×10^{-6}

Figure 3.12 shows the calculated $\frac{dx}{dt}$ value using a direct gradient between dx and dt. There is a large amount of artificial noise within the data. Therefore, two polynomial curves are fitted to the pump stroke travel distance for the calculation of $\frac{dx}{dt}$. As shown in Figure 3.13, a 2nd order polynomial function is fitted to the pump in-stroke, while a 5th order polynomial function is

fitted to the pump out-stroke. The resulting calculated $\frac{dx}{dt}$ and volumetric flow rate $\dot{Q}$ is shown in Figure 3.14.

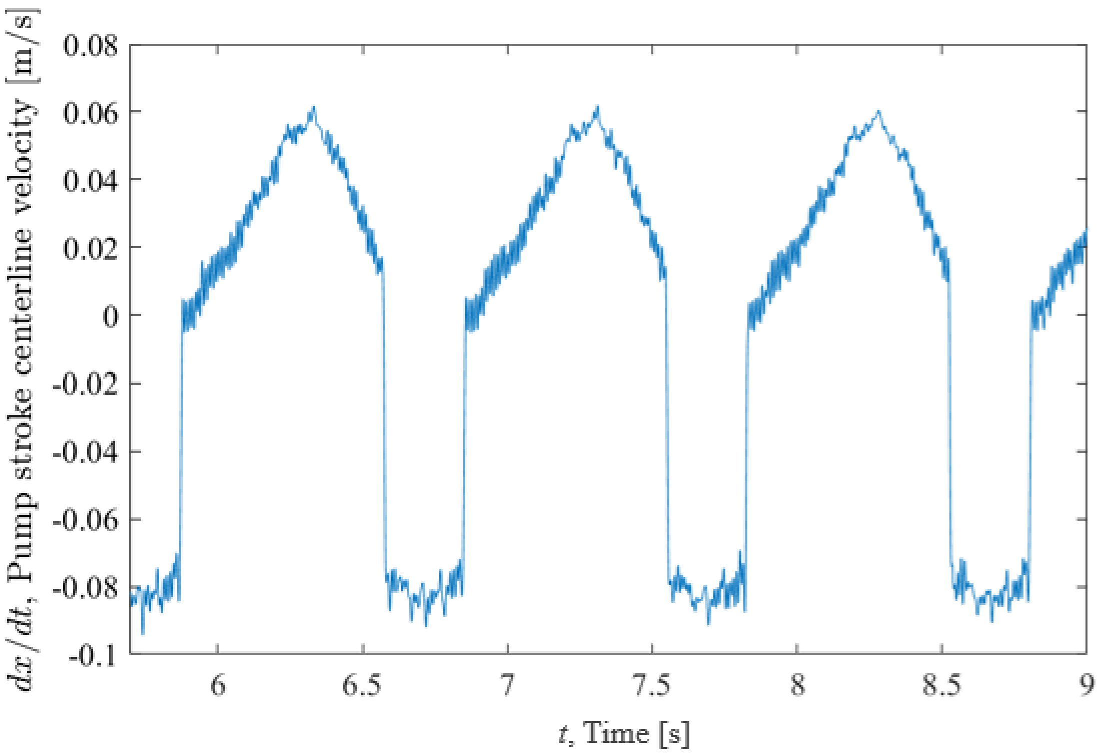


Figure 3.12: Centerline velocity as a function of time for three pump cycles at the aorta outlet

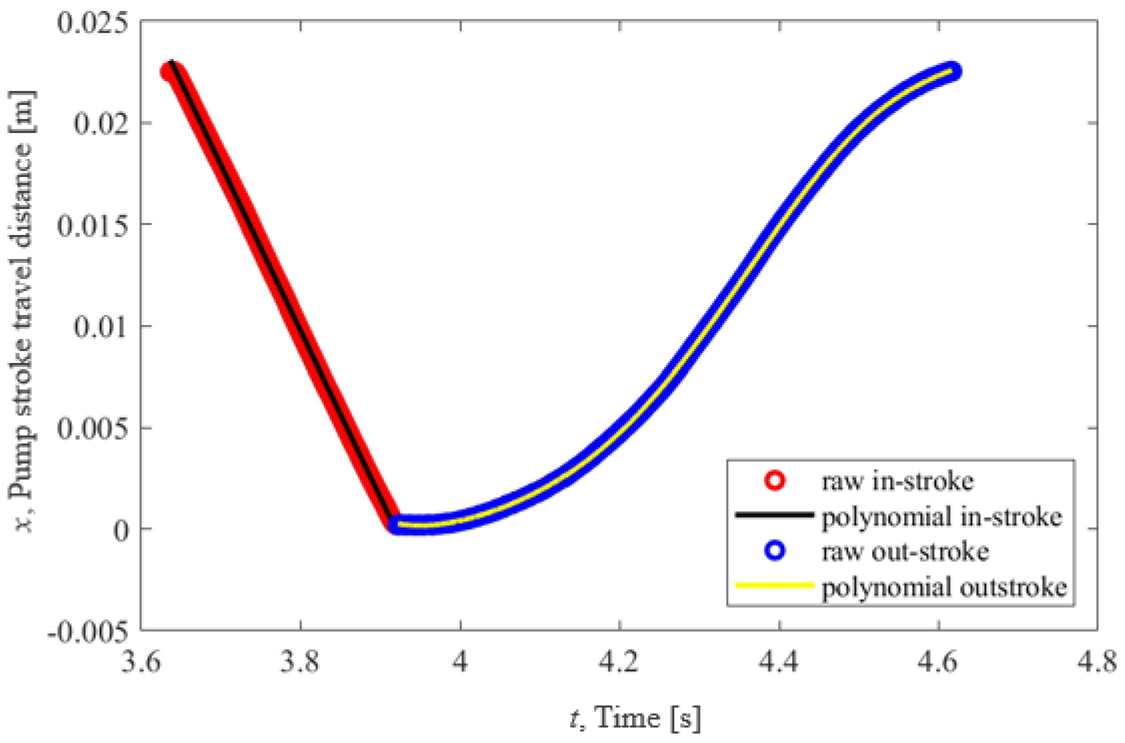


Figure 3.13: Approximating pump stroke travel distance using two polynomial expressions

Figure 3.14 shows a plot of the calculated pump stroke centerline velocity at the aorta outlet for three pump cycles. There is a definitive point in each cycle where the velocity changes sharply from positive to negative and vice versa, marking changing point between pump fill and ejection strokes.

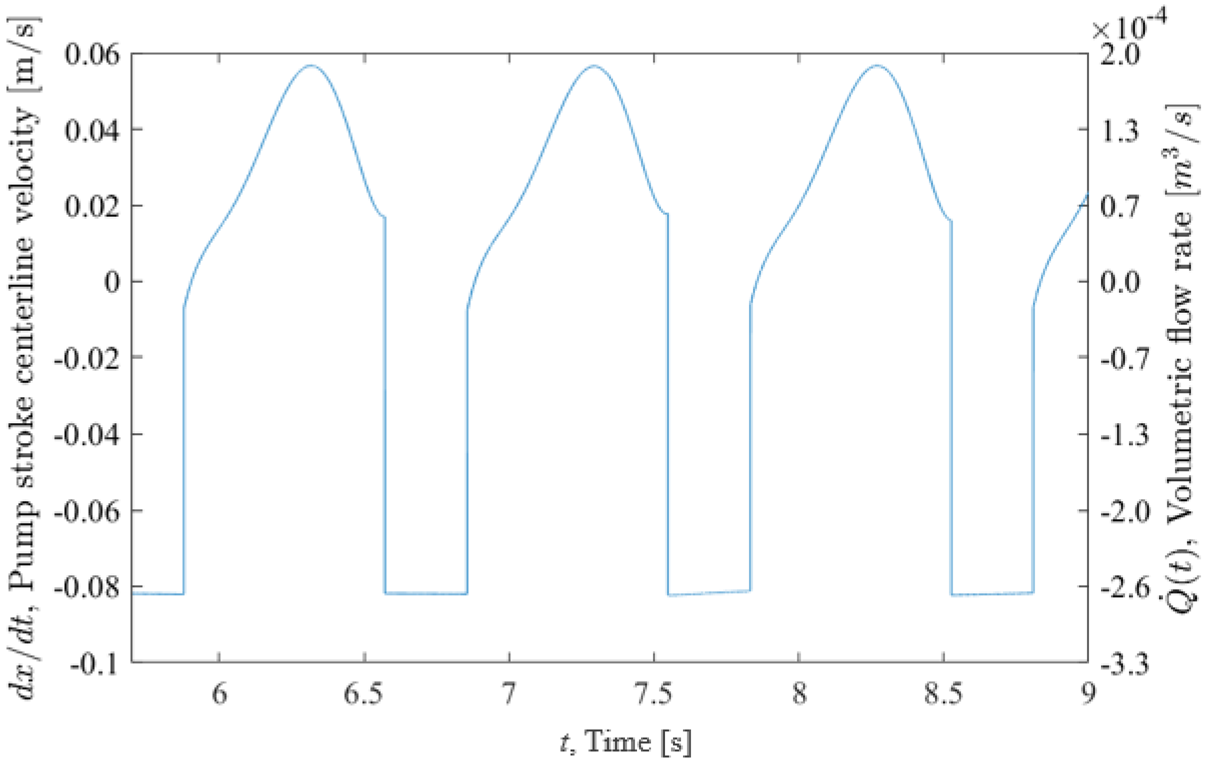


Figure 3.14: Volumetric flow rate and centerline velocity as a function of time for three pump cycles at the aorta outlet, using a polynomial expression to approximate the LVDT travel distance

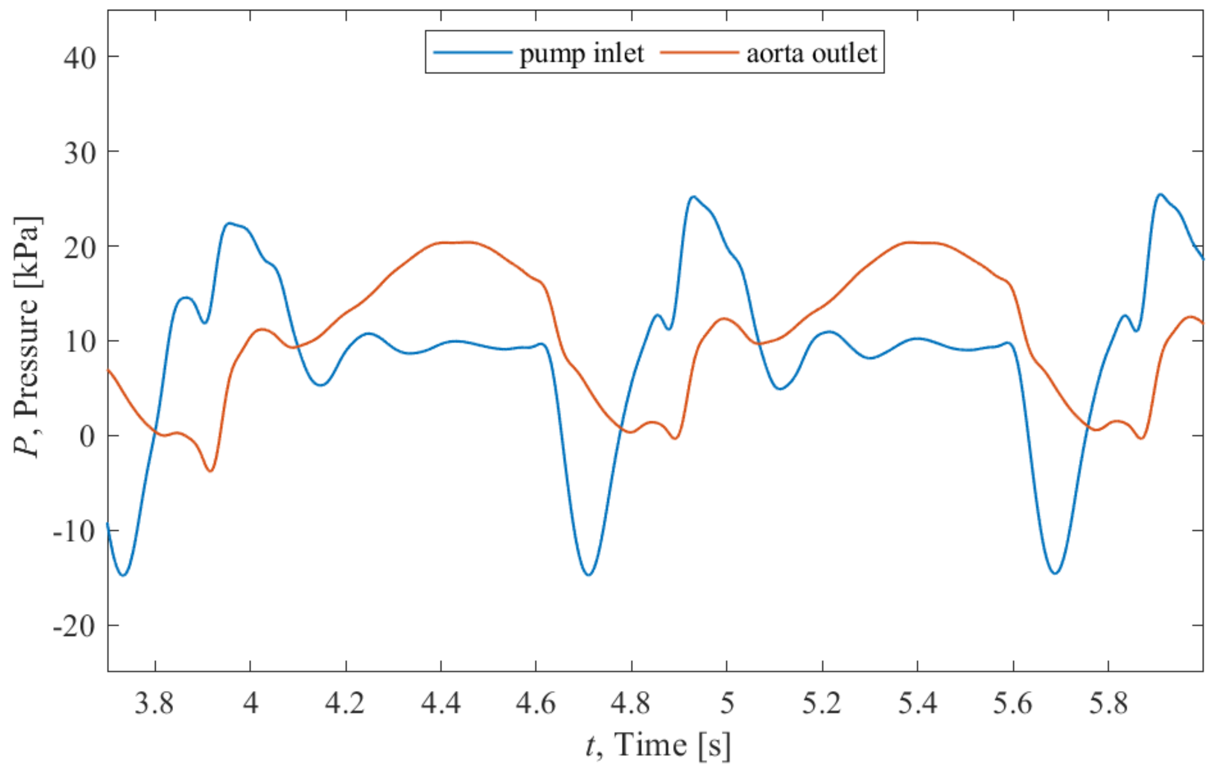
The calculation of average centerline velocity is shown in Equation 3.3.

$$\bar{v}_n(t) = \frac{\dot{Q}(t)}{A_n} \tag{3.3}$$

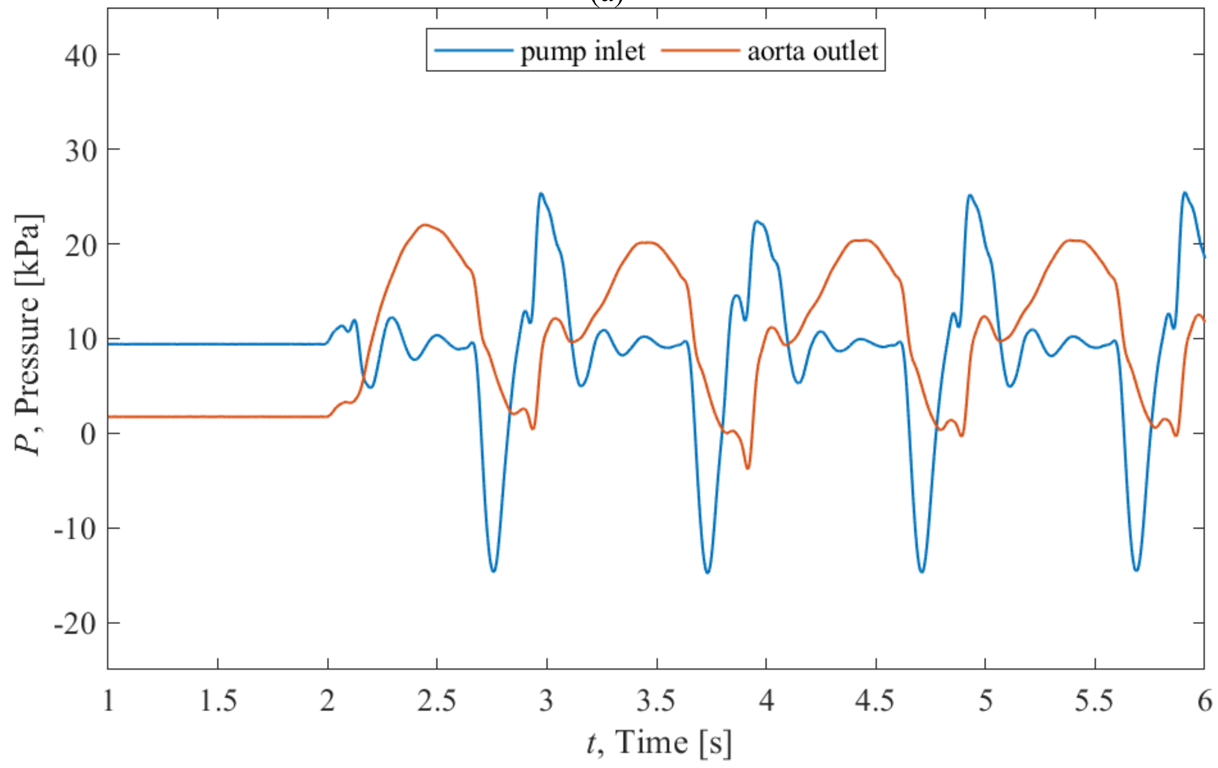
where:

A_n	=	Cross sectional area at location *n*

There will be a small amount of negative fluid centerline velocity as the fluid initially travels back before valve closure. This value was assumed to be negligible for the purpose of the pump energy calculation. Figure 3.15 (a) shows a plot comparing the pressure readings at the pump inlet and at the aorta outlet, adapted from Figure 3.8 in Section 3.3. At time t = 4.6 s to t = 5 s, the pump inlet pressure dramatically dips before rising sharply. The sudden pressure drop and subsequent sharp peak at the pump inlet can be attributed to the opening of the inlet valve as the pump shaft reverses direction into the in-stroke. During the outstroke at time t = 5 s to t = 5.6s, the pump inlet pressure drops to approximately 9.5 kPa with some dampening oscillations due to the nature of the ball valve. This value is comparable to the static pressure head measured in the initial seconds of data capture before the pump is turned on, as shown in Figure 3.15 (b). 9.41 kPa is the static pressure head at the location of the pump inlet valve. This indicates there is minimal fluid motion at the inlet side of the pump during the pump ejection-stroke. During this time, the aorta outlet pressure gradually increases to a maximum before relaxing slightly, following the ejection stroke of the pump. From time t = 5.6 s to 5.7 s, the aorta outlet pressure decreases, and plateaus from time t = 5.7-5.9. The plateau pressure matches the inlet static pressure shown in Figure 3.15 (b). The small rise and fall of the pressure within the plateau region can be attributed to the tri-leaflet valve closure. Therefore, for the purpose of this experiment, the fluid backflow is assumed to be negligible downstream of the pump outlet upon the return stroke of the pump. Figure 3.16 shows a plot both volumetric flow rate and centerline velocity as a function of time for three pump cycles using this assumption.



(a)



(b)

Figure 3.15: Pressure plot comparison at the pump inlet and aorta outlet

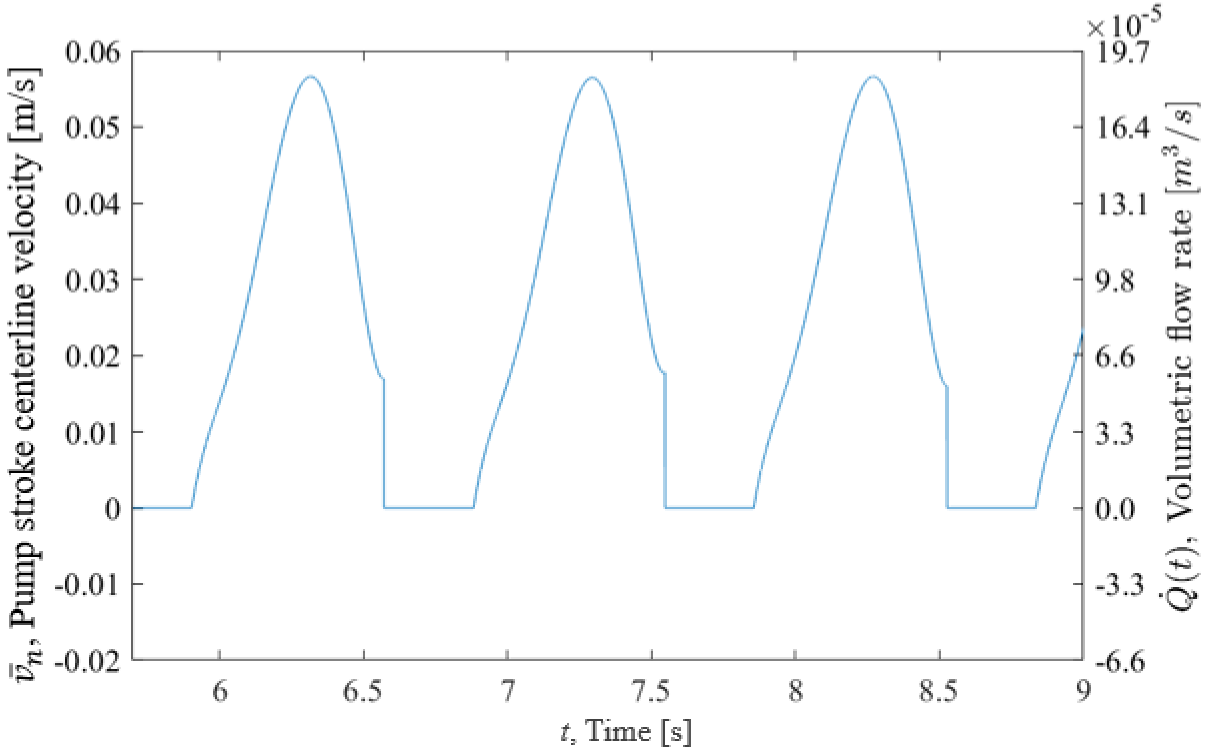


Figure 3.16: Adjusted volumetric flow rate and centerline velocity as a function of time for three pump cycles at the aorta outlet with the pump suction stroke values set to zero

3.6. Pump Energy Calculation

As discussed in Section 1.5, a modified Bernoulli equation (Equation 1.4) for the conservation of energy is broken down into its energy components. Equation 3.4 shows the pump energy calculated as the sum of these four energy components: pressure (Equation 3.5), kinetic (Equation 3.6), static (Equation 3.7), and local acceleration (Equation 3.8). Pump energy is calculated at all four pressure transducer locations: pump inlet, pump outlet, aorta inlet, and aorta outlet. The MATLAB code used to calculate pump energy is shown in Appendix J.

$$E_{pump} = E_{pressure} + E_{kinetic} + E_{static} + E_{local\ acceleration} \tag{3.4}$$

$$E_{pressure} = \int_0^{t_c} \dot{Q}(t) P_n(t)\,dt \tag{3.5}$$

$$E_{kinetic} = \frac{\rho}{2}\int_0^{t_c} \dot{Q}(t)\left(\bar{v}_n(t)\right)^2 dt \tag{3.6}$$

$$E_{static} = \rho g \Delta z \int_0^{t_c} \dot{Q}(t)\,dt \tag{3.7}$$

$$E_{local\ accel} = \rho L_{eff_n} \int_0^{t_c} \dot{Q}(t)\left(\frac{dv}{dt}\right)_n dt \tag{3.8}$$

where:

E_{pump}	=	Total pump energy per pump pulse cycle [J/cycle]	$\dot{Q}(t)$	=	Volumetric flow rate [m^3/s]
$E_{pressure}$	=	Pressure pulse energy	$P_n(t)$	=	Pressure waveform at location n [Pa]
$E_{kinetic}$	=	Kinetic energy	$\bar{v}_n(t)$	=	Average centerline velocity at location *n* [m/s]
E_{static}	=	Static energy	Δz_n	=	Height difference between preload reservoir and location n [m]
$E_{local\ accel}$	=	Local acceleration energy	L_{eff_n}	=	Height difference between location *n* and preload reservoir
n	=	Locations of energy calculation: Pump inlet, Pump outlet, Aorta inlet, Aorta outlet	$\left(\frac{dv}{dt}\right)_n$	=	Acceleration at location n calculated from $\bar{v}_n$ [m/s^2]
			ρ	=	Density of water; 1000 kg/m^3
			g	=	Acceleration of gravity; 9.81 m/s^2
t_c	=	Cycle time [s]			

The total pump energy E [J/cycle] at each location is calculated as an average of the pump energy per cycle. Seven cycles are used for each experiment case, due to seven being the maximum number of cycles the slowest pump speed of 60 bpm can consistently output. Each energy term is calculated as a function of $\dot{Q}(t)$, the volumetric flow rate [m^3/s]. $\dot{Q}(t)$ is calculated using Equation 3.2, form the maximum stroke volume per cycle, Q_{cycle} [m^3]. Since the final energy output is averaged over all cycles, an average Q_{cycle} value is used and assumed to be the same for all cycles.

Using the calculated $\dot{Q}(t)$ the pump energy is calculated. Figure 3.17 shows a plot of all energy types for each cycle for one experimental case. The energy terms vary slightly across cycles, but the differences are negligible in comparison to the final averaged total energy term.

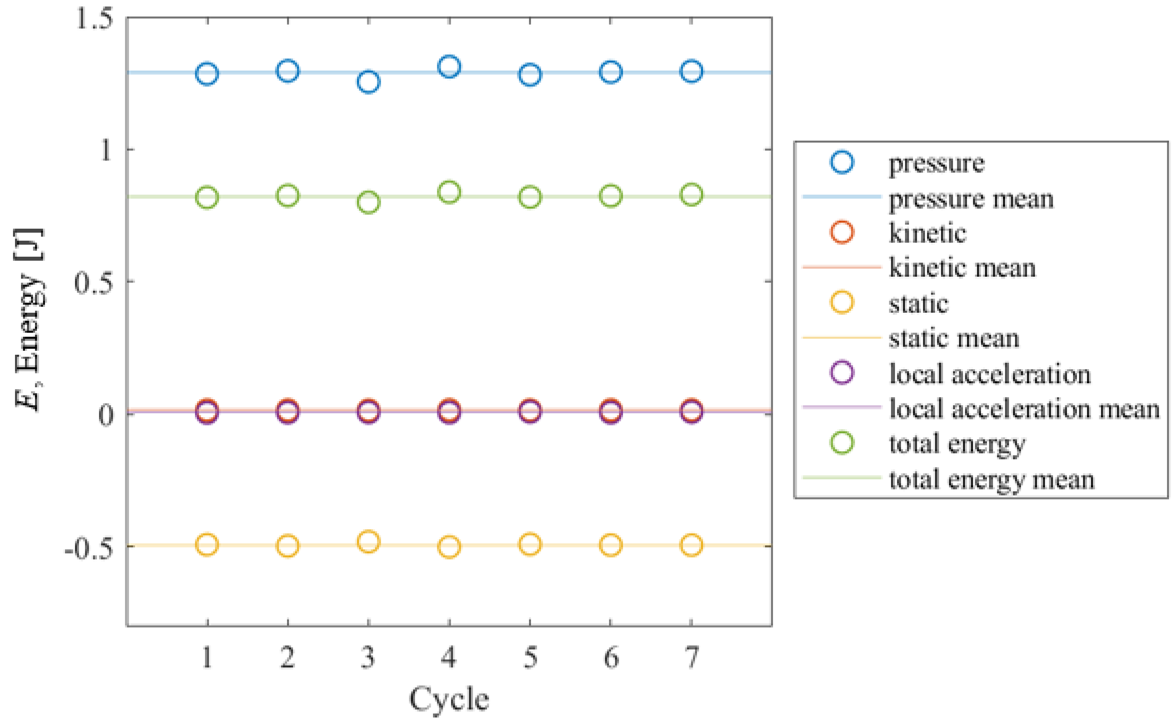


Figure 3.17: Plot of all energy types for one experimental case

This calculation of pump energy determines the conservation of energy at each pressure measurement point in the flow loop. These calculated values provides insight in energy losses in terms of pressure head loss throughout the system. Since a major goal of this investigation is to determine the effect of a compliant tube on the pump loop, the percentage change in energy at the aorta outlet from the aorta inlet is used as a method of comparison. Equation 3.9 calculates this percent energy change, *ΔE*. Detailed analysis of pump energy as a function of different experimental parameters are discussed in Chapter 3. The MATLAB code used to calculate pump energy is shown in Appendix K.

$$\Delta E = \frac{E_{aorta\ outlet} - E_{aorta\ inlet}}{E_{aorta\ inlet}} \times 100\% \qquad 3.9$$

where:

$E_{aorta\ outlet}$	=	Energy per cycle [J/cycle] at the aorta outletr
$E_{aorta\ inlet}$	=	Energy per cycle [J/cycle] at the aorta inlet

3.7. Limitations

Due to the limiting size of the pressure-chamber, the maximum distension of the compliant tube is the width of the chamber. The tube has an outer diameter of 1.06 in and the chamber has a 2x2 in square area. The maximum distension the tube can undergo is less than approximately 89%. This is further decreased due to the need to crop the image during image processing. The image processing technique relies on edge detection between the bright background and the darkened tube. If the tube expands to the walls of the pressure-chamber, which are also black in the image, then the distension of the tube cannot be accurately determined. Due to the limiting size of the pressure-chamber, too much backpressure will cause the aorta to expand to the chamber walls. Chamber-pressure controls the distension amount but will cause the tube to collapse when too high. Thus, for each backpressure case, there are data points for only the specific range of chamber-pressures the aorta can safely distend and retract without wall-contact or collapse. Therefore, these cases are removed from the experimental results during image processing. This results in incomplete data sets for comparison between cases in Chapter 4. Experimental cases using an Ecoflex aorta is heavily affected by case removal. Ecoflex, with a lower Young's modulus, is softer and therefore "wall-touching' of the tube with the pressure-chamber walls occurs more frequently than the stiffer Dragon Skin cases. The effects will be discussed in detail in Section 4.5.

Another limitation for the calculation of pump energy is the lack of information on the profile of velocity at the pressure measurement locations. The fluid velocity used in calculation is based off the pump stroke distance measured by the LVDT, using the assumption of a cylindrical pump chamber. An improvement to future work can incorporate Particle Imaging Velocimetry (PIV) at pressure measurement locations to have an accurate visualization of the velocity profile.

3.8. Summary

This chapter described the data processing methodology for the high-speed imaging data, pressure data, pump stroke distance data, and pump energy calculations. High-speed camera images of the compliant tube were used to determine compliant tube distension percentages across the full experimental data collection time range. Collected pressure data at different locations within the experimental pump loop was filtered and the results used for calculation of pump energy, as well as comparison of system response when different experimental conditions are introduced. Pump stroke distance data was collected and filtered to allow calculation of volumetric flow rate. Pump energy was calculated using the processed pressure and pump stroke distance. Certain experimental cases were discarded due to limiting size of the pressure-chamber not allowing full distension of the tube, as well as the silicone flexibility causing the tube to collapse under high chamber-pressures. The following chapter discusses the impact of varying system parameters on pump performance using the processed results.

Chapter 4. Impact of Varying System Parameters on Pump Performance

4.1. Introduction

This chapter discussed the effect of experimental parameters on pump energy and aorta elastic response, as well as the effect of aorta elastic response on pump energy. Section 4.2 discuss the impact of experimental parameters on pump energy. Pump energy and the percent energy decrease at the aorta outlet from the aorta inlet, is compared against backpressure, chamber-pressure, and pulse frequency/stroke volume per cycle. Note that pulse frequency and stroke volume are grouped together due to the torque limits of the motor driving the pump. With an increase in pulse frequency there is a decrease in stroke volume and vice versa. Section 4.3 discuss the impact of aorta properties on aorta elastic response. The effects chamber-pressure and aorta length on tube percent diameter distension are examined. Section 4.4 discuss the impact of aorta elastic response on pump energy. The relationship between percent energy decrease—at the aorta outlet from the aorta inlet—versus the maximum percent diameter distension are examined with respect to the stroke volume per cycle and pulse frequency. Section 4.5 discuss the limitations within the experimental results. Table 4.1 lists the common terms and symbols used in the discussion of experimental results.

Table 4.1: Terms and symbols used in results discussion

Term	Symbol	Definition
Backpressure	b	Backpressure applied to the system using the centrifugal backpressure pump. The pressure value is the difference from mock aorta inlet to the pump inlet.
Chamber-pressure	c	Chamber-pressure
Pulse frequency	f	Heart rate expressed in beats per minute [bpm], see also: bpm
Bpm	bpm	Beats per minute, describes pump pulse frequency
Mock aorta length	l	Length of mock aorta tested
Stroke volume	SV	Stroke volume per beat of heart pump
Material	m	Aorta material, Dragon Skin and Ecoflex silicones

4.2. Impact of Experimental Parameters on Pump Energy

4.2.1. Pump Energy at Aorta Inlet and Aorta Outlet, *E*

Table 4.2 shows the number of experimental cases for Ecoflex and Dragon Skin aortas separated by backpressure. The number of usable experimental cases is higher for Dragon Skin than Ecoflex, as discussed in Section 3.7. This is most apparent for the 15.66 kPa backpressure case, there are only five data points available for Ecoflex, but 49 data points available for Dragon Skin. Therefore, Dragon Skin was chosen as the main material for the focus of the investigation. Ecoflex cases, where available, are compared to Dragon Skin results to determine trend similarity. The experimental limitations due to discarding specific experimental cases are discussed in detail in Section 4.5.1.

Table 4.2: Number of experimental cases for Ecoflex and Dragon Skin aortas, separated by backpressure

Backpressure [kPa]	No. Ecoflex cases	No. Dragon Skin cases
0.00	16	45
0.88	12	37
3.71	13	41
8.62	10	45
15.66	5	49

Figure 4.1 and Figure 4.2 show plots of pump energy per cycle, E, at the aorta outlet versus backpressure. All plots of E vs backpressure are shown in Appendix L. The three aorta lengths are shown in different colors. The plots are separated by chamber-pressure (c), pulse frequency (f), and material (m) that are listed at the top of each figure. As discussed in Section 3.7, for each backpressure case, there are data points for only the specific range of chamber-pressures the aorta can distend and retract without wall-contact or collapse. The impact of missing cases is discussed in Section 4.5.1. When looking at all plots overall, an upwards trend exist for experimental cases with sufficient data points. The upward trend appears to follow an exponential pattern, increasing sharply from 3.71 kPa onwards.

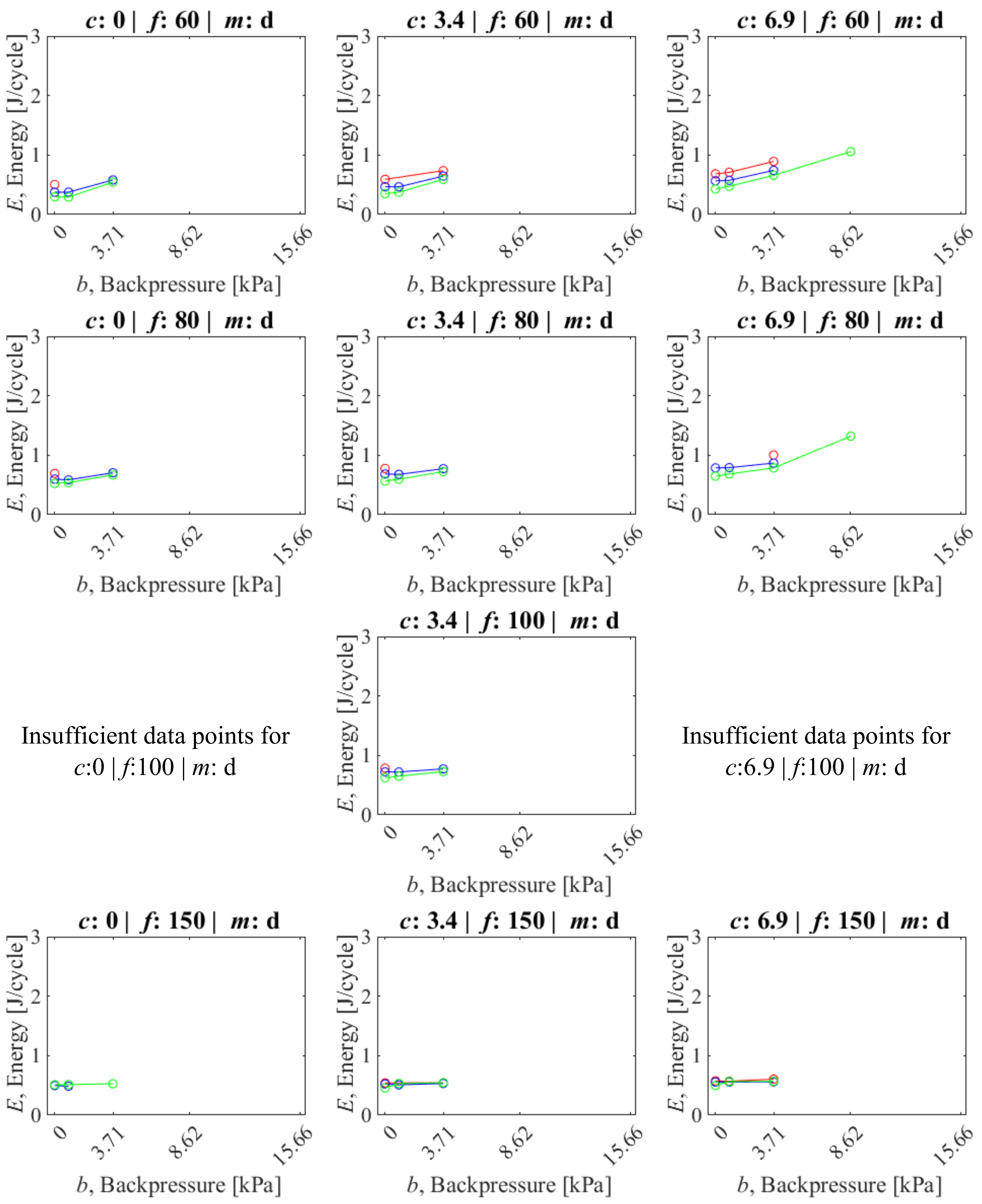


Figure 4.1: Plots of pump energy per cycle at aorta outlet vs backpressure. Chamber-pressure 0-6.9 kPa.
c: chamber-pressure [kPa]; *f*: pulse frequency [bpm]; *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

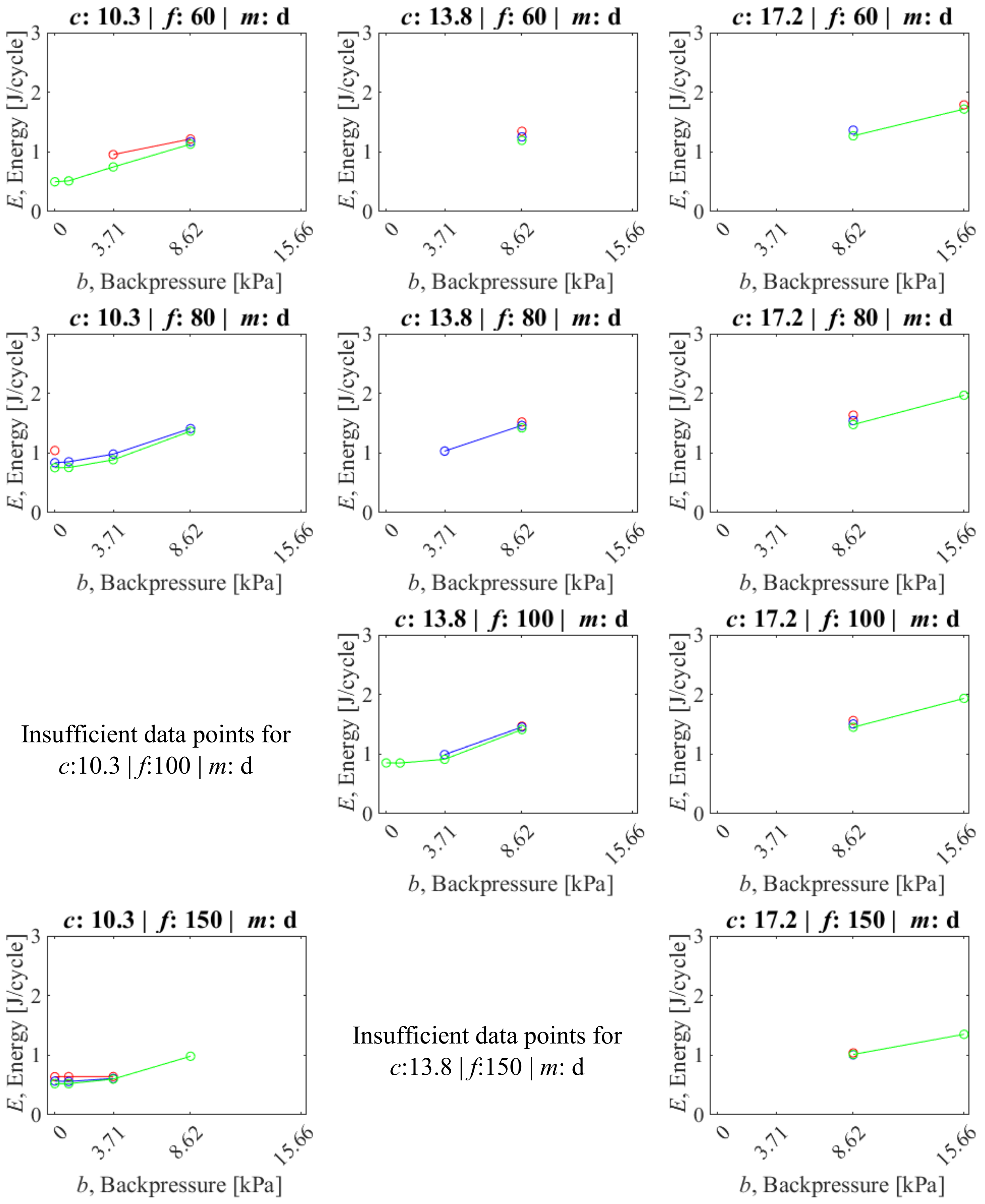


Figure 4.2: Plots of pump energy per cycle at aorta outlet vs backpressure. Chamber-pressure 10.3-17.2 kPa.
c: chamber-pressure [kPa]; *f*: pulse frequency [bpm]; *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

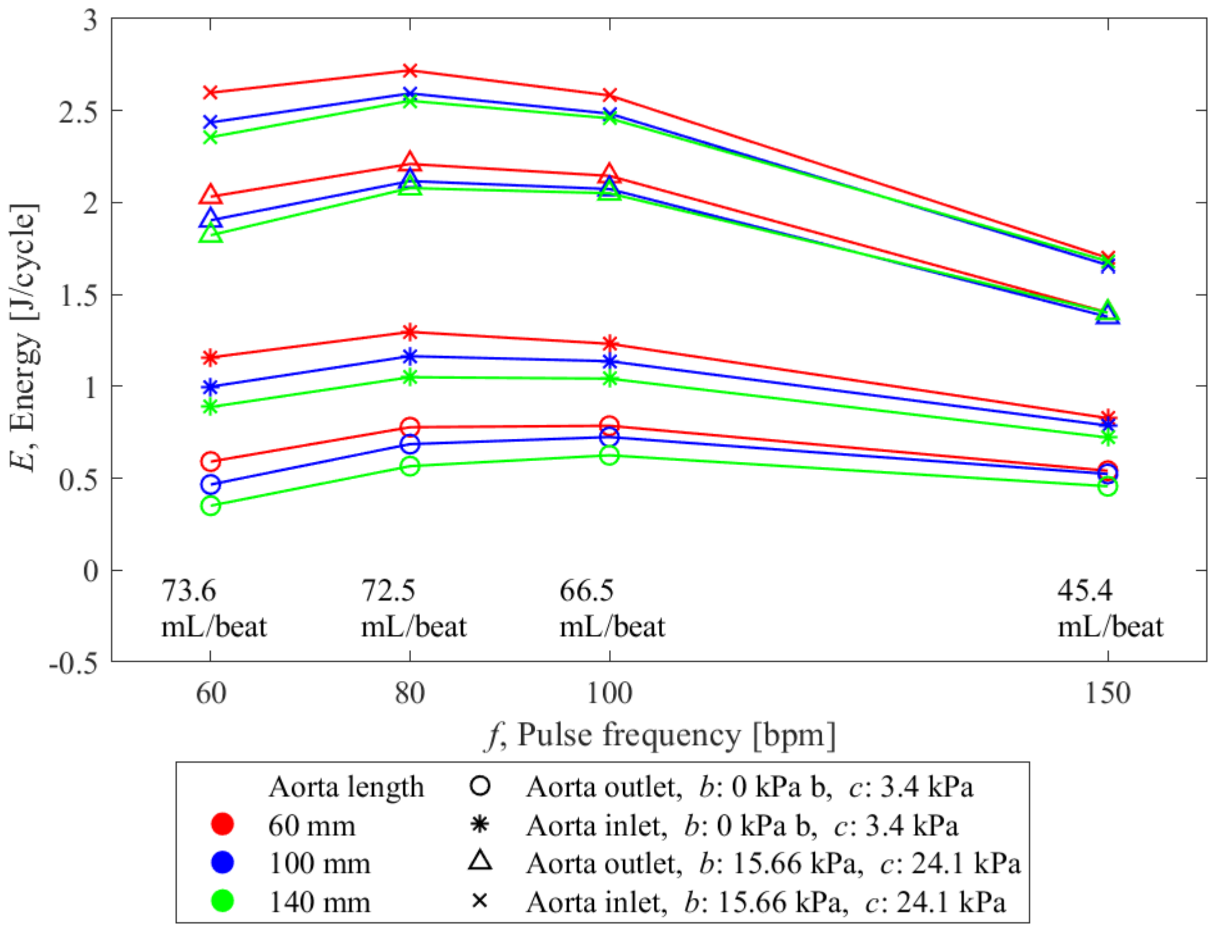


Figure 4.3: Plot of energy per pump cycle vs pulse frequency for both 0 kPa backpressure, 3.4 kPa chamber-pressure, and 15.66 kPa backpressure, 24.1 kPa chamber-pressure; at both aorta inlet and aorta outlet. Dragon Skin tube material.

Two full experimental sets were found for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 12.95 backpressure and 24.1 kPa chamber-pressure. Figure 4.3 shows the energy per cycle, *E* [J/cycle] of the two data sets, plotted against pulse frequency [bpm], at both the aorta outlet and aorta inlet test locations. All three aorta lengths are included in both data sets. There is a clear energy increase between the backpressure condition of 15.66 kPa from 0 kPa. At both ends of the experimental backpressure range, the energy values rise from 60 bpm to 80 bpm and decrease from 100 bpm to 150 bpm. Within each experimental set, the energy at the aorta inlet is higher than the aorta outlet among the same aorta length cases. At each test location, a longer aorta length corresponds to a lower energy value. As pulse frequency increases, there is also a decrease in the distribution of energy per cycle. This trend is investigated further by finding the percentage decrease in energy between the aorta inlet to the aorta outlet, in Section 4.2.2.

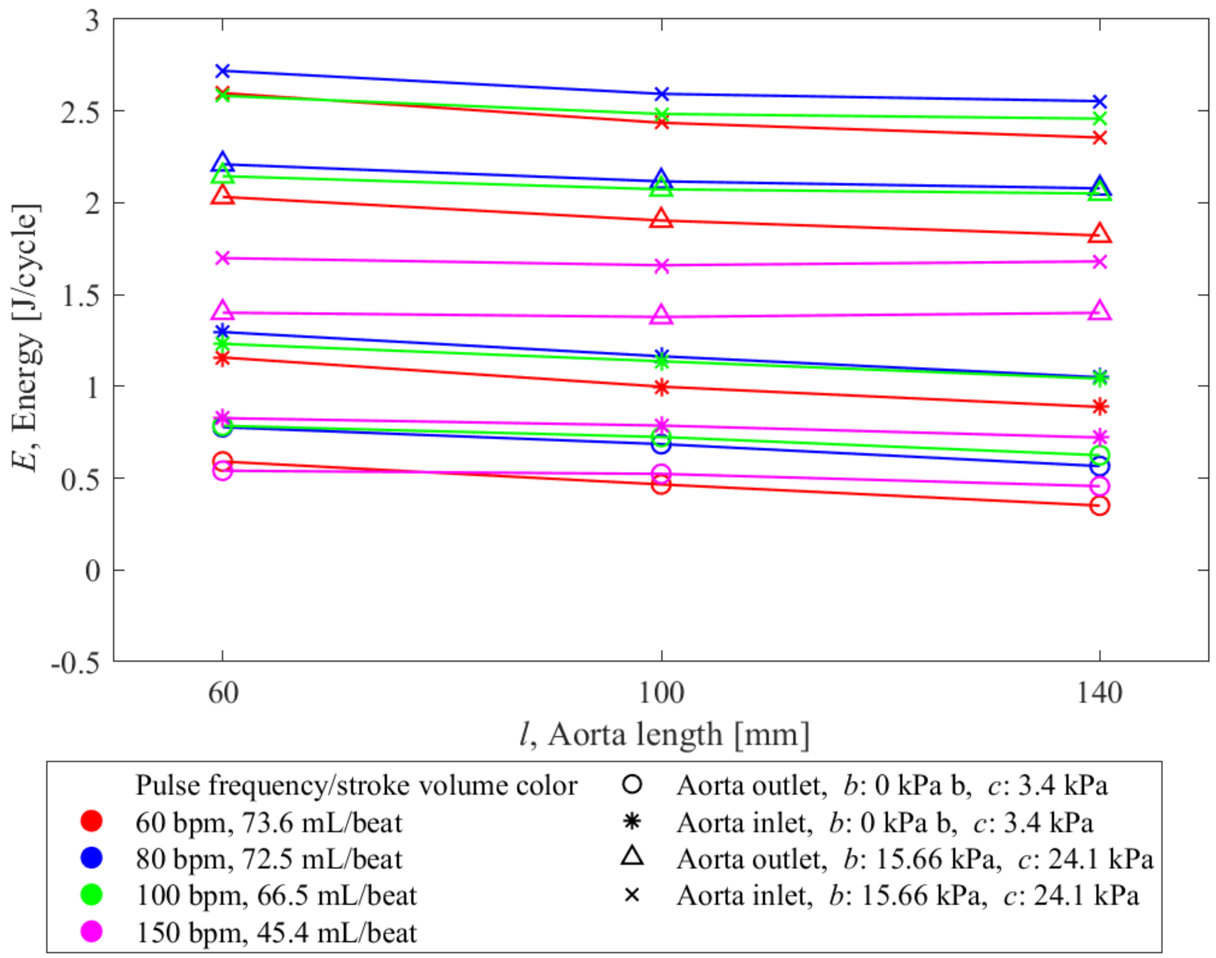
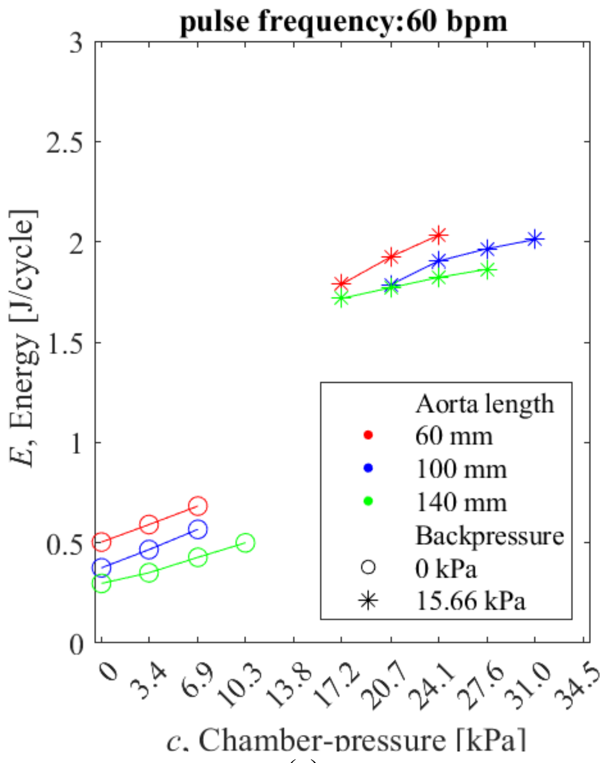


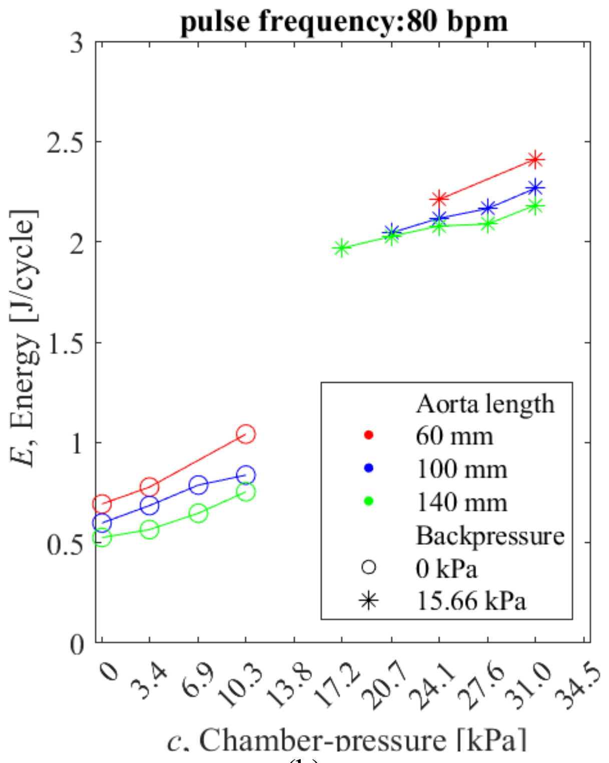
Figure 4.4: Plot of energy per pump cycle vs aorta length for both 0 kPa backpressure, 3.4 kPa chamber-pressure, and 15.66 kPa backpressure, 24.1 kPa chamber-pressure; at both aorta inlet and aorta outlet. Dragon Skin tube material.

Figure 4.4 shows the energy per cycle versus aorta length at both the aorta outlet and aorta inlet test locations. Data points at 0 kPa backpressure and 3.4 kPa chamber-pressure, and 2000 rm backpressure and 24.1 kPa chamber-pressure are included. At both ends of the experimental backpressure range, aorta length does not have as large an impact on the pump energy as pulse frequency/stroke volume seen in Figure 4.3. In general, most cases decrease in *E* slightly with increasing aorta length, an exception being the 150 bpm/45.4 mL/beat case, where *E* appears to be more constant.

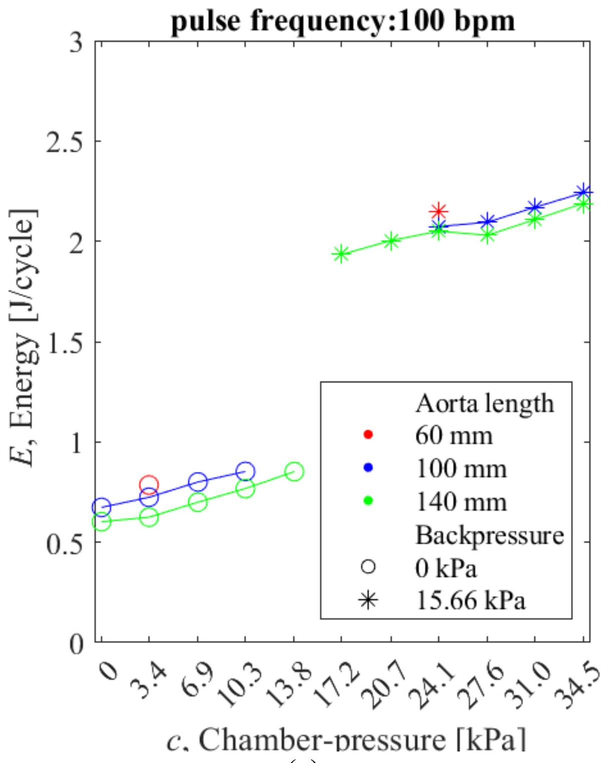
To visualize the effects of chamber-pressure, energy per pump cycle at the aorta outlet is plotted against chamber pressure. Figure 4.5 shows all pulse frequency cases from 60-150 bpm. Two sets of experimental data at backpressures of 0 and 15.66 kPa are compared. The aorta material is Dragon Skin. An upward trend of energy can be seen with increasing chamber pressure. Decreasing energy with increasing aorta lengths is again seen.



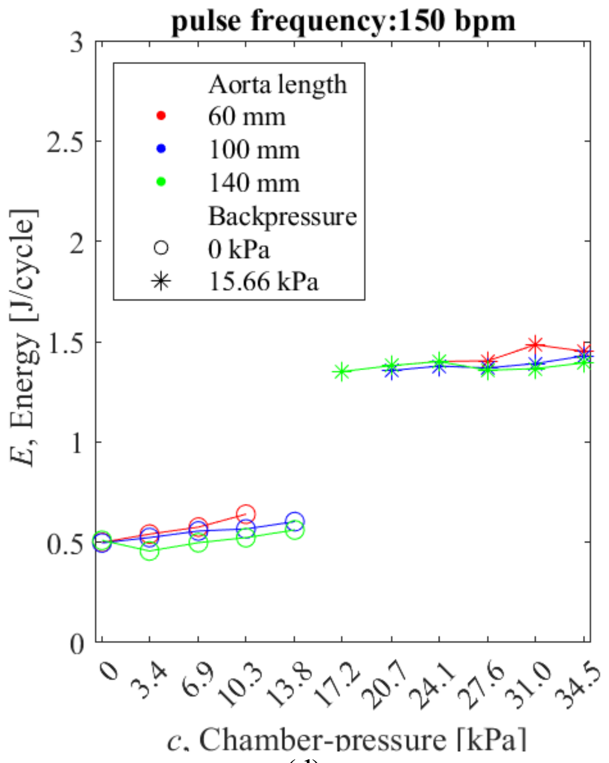
(a)



(b)



(c)



(d)

Figure 4.5: Plot of energy per pump cycle vs chamber-pressure for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Location: aorta outlet. Material: Dragon Skin.

4.2.2. Percent Energy Decrease from Aorta Inlet to Aorta Outlet, *ΔE*

To further visualize the effect of experimental parameters on pump energy, the percent energy decrease from the aorta inlet to the aorta outlet, *ΔE*, is found. Figure 4.6 and Figure 4.7 show some plots of percent energy decrease at the aorta outlet from the aorta inlet, *ΔE*, at the aorta outlet versus backpressure. All plots of *ΔE* vs backpressure are shown in Appendix M. The three aorta lengths are shown in different colors. The plots are separated by chamber-pressure, pulse frequency, and material. As discussed earlier in Chapter 3 Section 3.7 regarding discarding experimental cases, for each backpressure case, there are data points for only the specific range of chamber-pressures the aorta can distend and retract without wall-contact or collapse. When looking at all plots overall, the same trend of decreasing *ΔE* with increasing backpressure exist for experimental cases with sufficient data points.

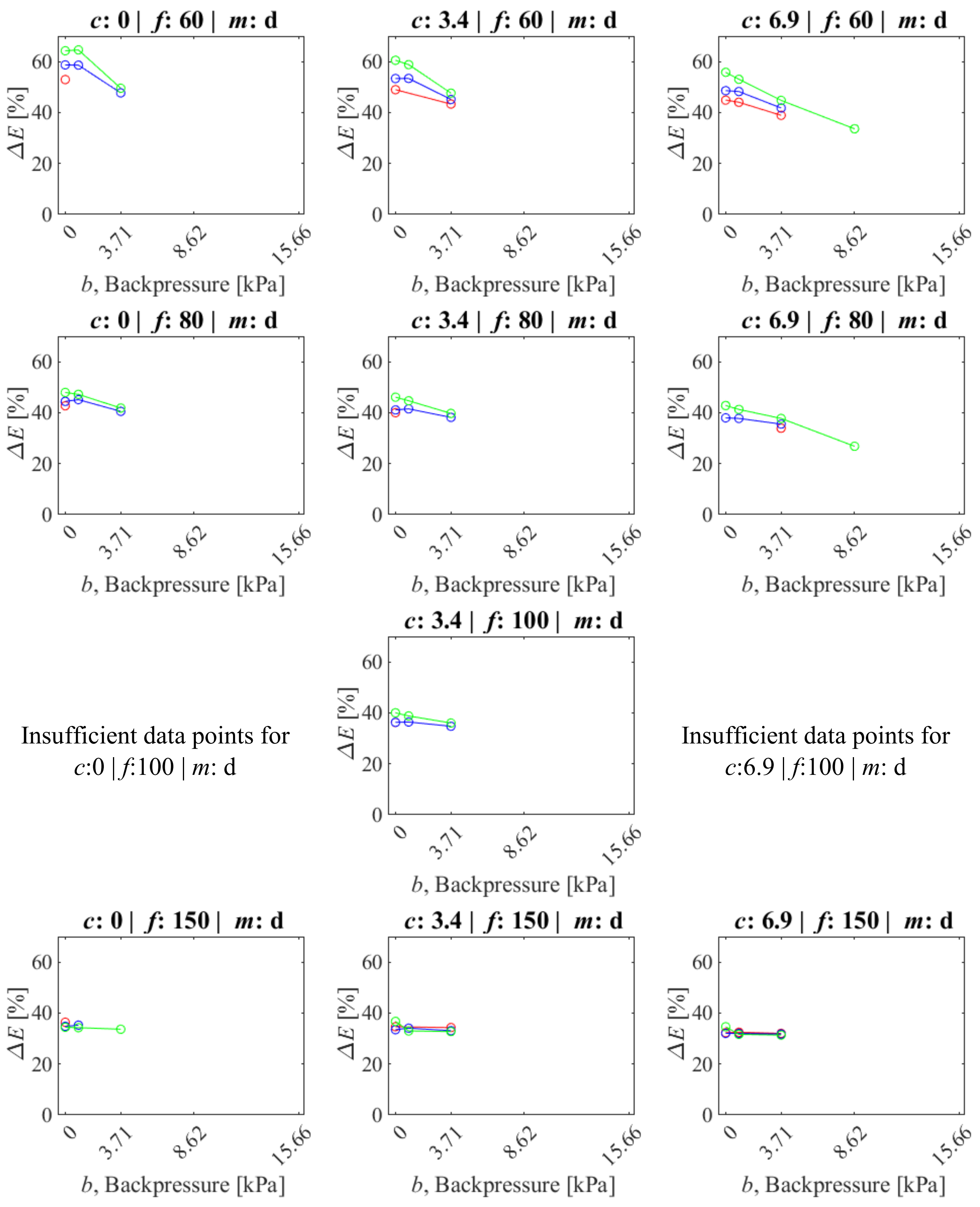


Figure 4.6: Plots of *ΔE,* percent energy decrease at the aorta outlet from the aorta inlet vs backpressure. Chamber-pressure 0-6.9 kPa.
c: chamber-pressure [kPa]; *f*: pulse frequency [bpm]; *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

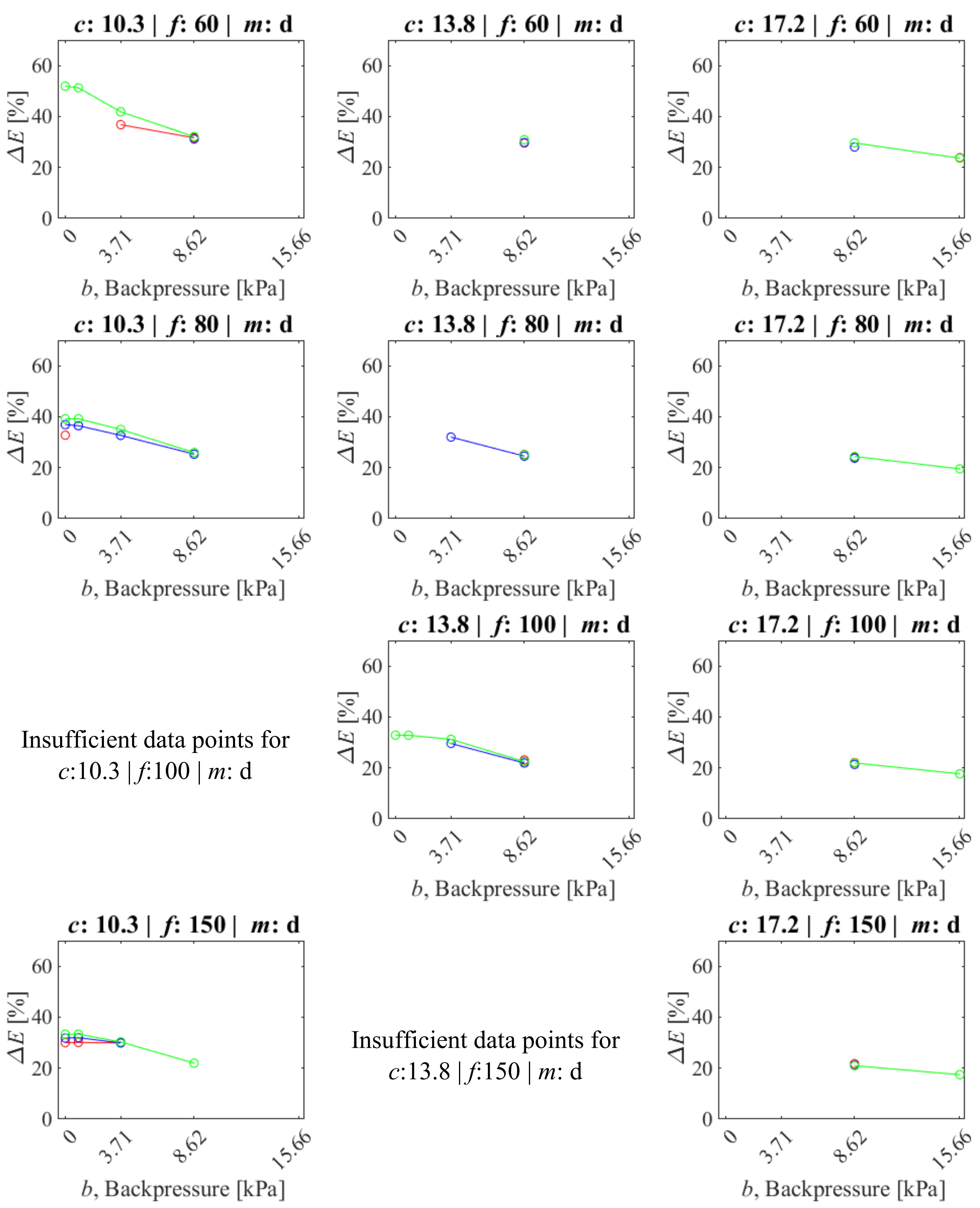
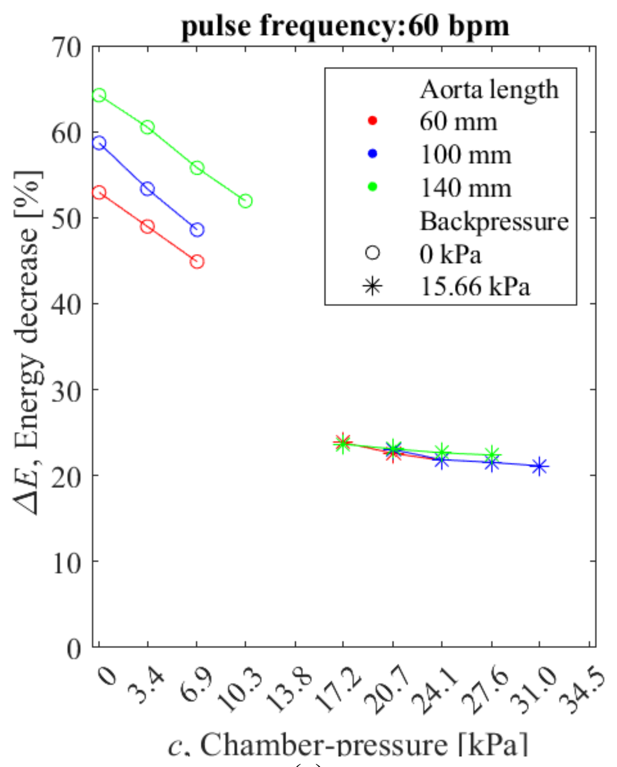
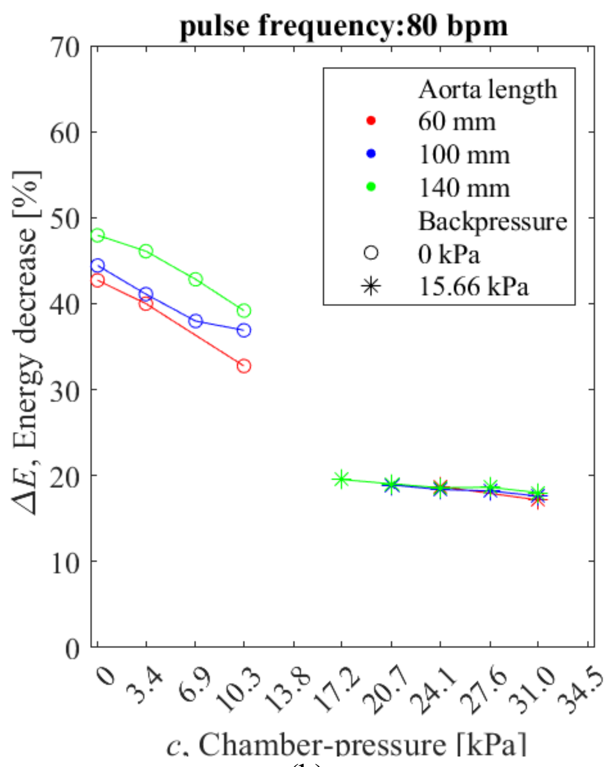


Figure 4.7: Plots of *ΔE,* percent energy decrease at the aorta outlet from the aorta inlet vs backpressure. Chamber-pressure 10.3-17.2 kPa.
c: chamber-pressure [kPa]; *f*: pulse frequency [bpm]; *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

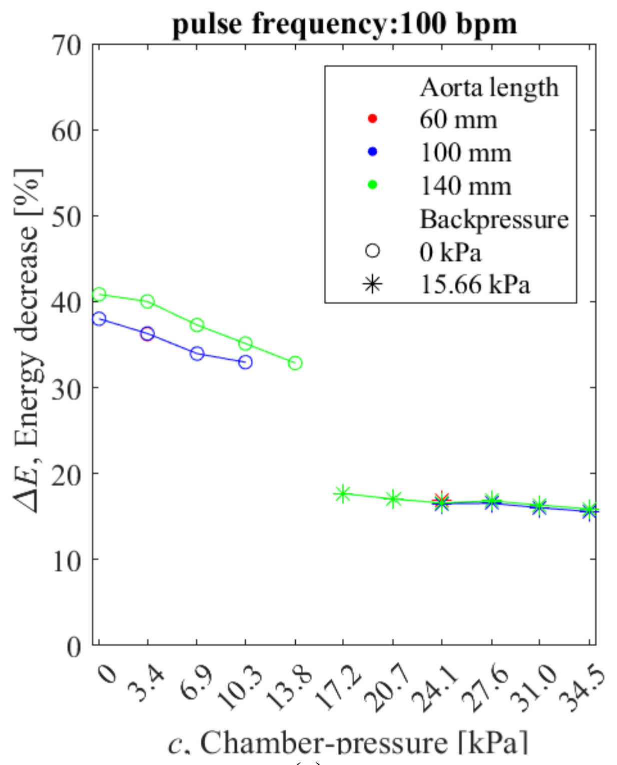
Figure 4.8 shows the plot of percent energy decrease, *ΔE*, at the aorta outlet from the aorta inlet vs chamber-pressure for all pulse frequency cases. Two sets of experimental data at backpressures of 0 and 15.66 kPa are compared. The aorta material is Dragon Skin. A downward trend in *ΔE* can be seen with increasing chamber pressure for the 0 kPa backpressure case. This downward trend is less prominent with increasing pulse frequency. The *ΔE* appears to either decrease or remain constant for the 15.66 kPa back pressure case. This agrees with the trend found in Figure 4.6 and Figure 4.7, where the *ΔE* is less across higher backpressure conditions. When comparing each aorta length plot at 0 kPa, as aorta length increase, the energy decrease at the aorta outlet increases. At 15.66 kPa, the energy change is similar across aorta lengths. This is an interesting contrast to Figure 4.5, which shows that an increase in aorta length resulted in a decrease in energy per cycle.



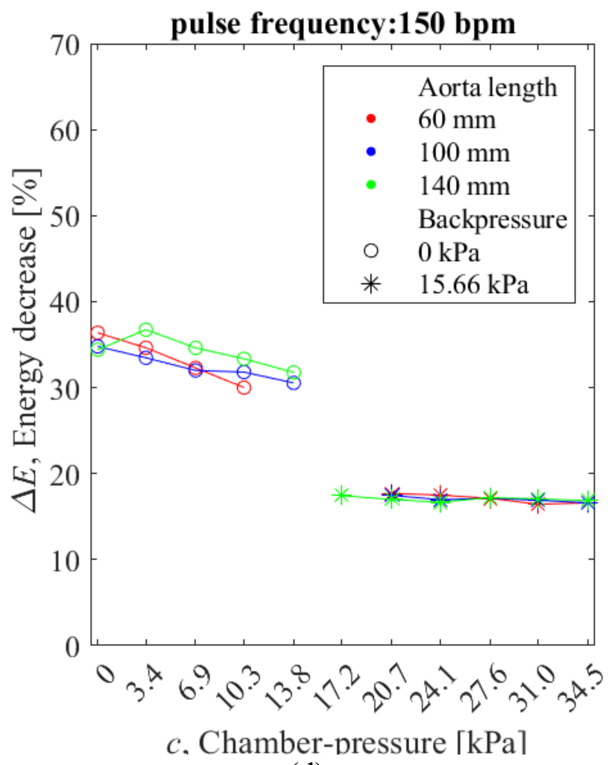
(a)



(b)



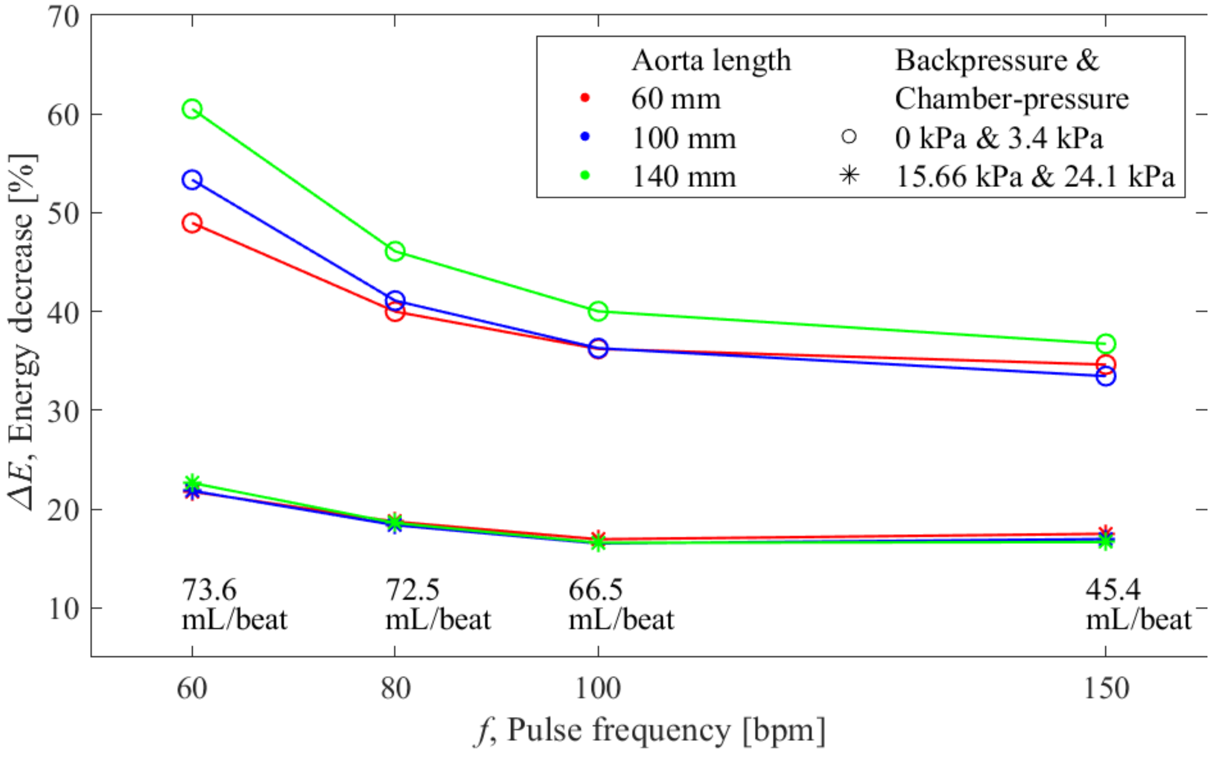
(c)



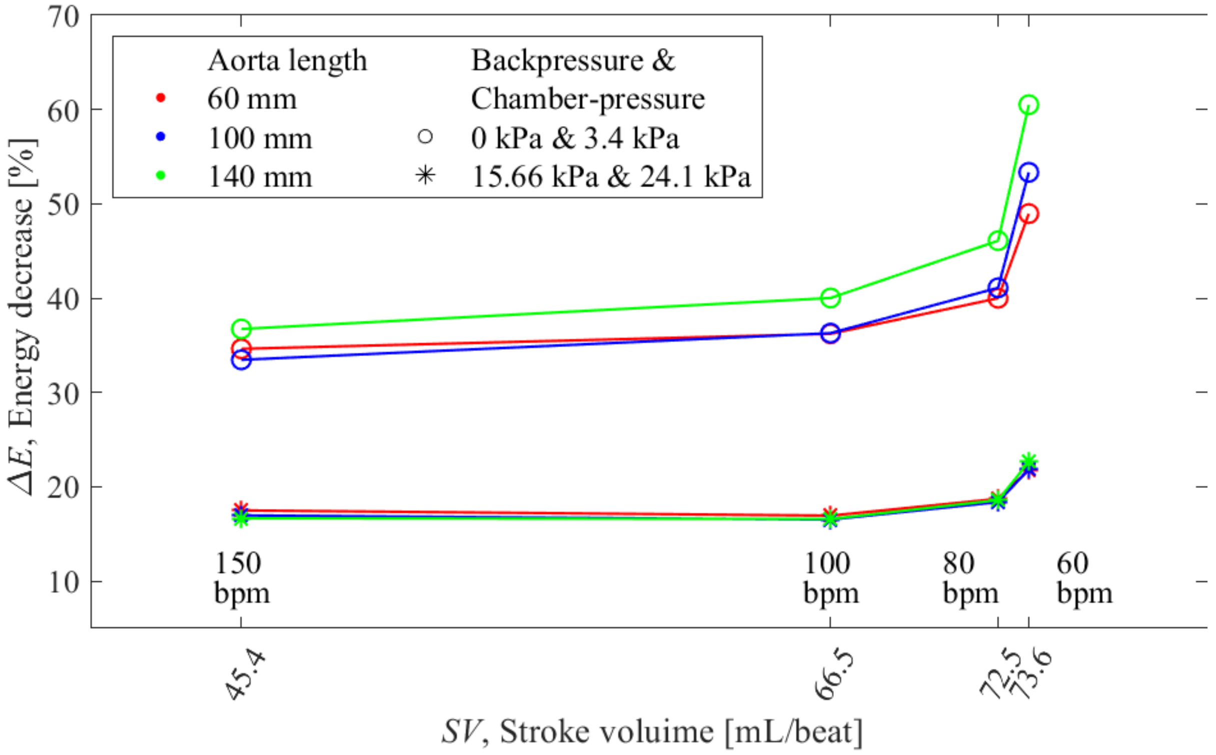
(d)

Figure 4.8: Plot of *ΔE* percent energy decrease at the aorta outlet from the aorta inlet vs chamber-pressure, separated by aorta length, for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Material: Dragon Skin.

ΔE is plotted against pulse frequency (Figure 4.9 (a)) and stroke volume per cycle (Figure 4.9 (b)) for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. The aorta material is Dragon Skin. The corresponding stroke volume per cycle for each pulse frequency case is labeled. Within each aorta length, the ΔE decreases with increasing pulse frequency from 60-100 bpm. Since an increase in pulse frequency corresponds to a decrease in stroke volume per cycle, the ΔE increases with increasing stroke volume. At 150 bpm, the ΔE appears to decrease slightly for the 0 kPa backpressure case, and increase slightly for the 15.66 kPa case, and vice versa for the corresponding stroke volume of 45.4 mL/cycle. There is a bigger range in ΔE between the aorta length at lower backpressures. This suggests aorta length and pulse frequency have more effect on the ΔE at lower backpressures. As with Figure 4.8, ΔE increases with increasing aorta length at 0 kPa backpressure and remains similar at 15.66 kPa backpressure. That is, a longer aorta length results in a larger decrease of energy at the aorta outlet from the aorta inlet at lower backpressures conditions.



(a)



(b)

Figure 4.9: Plot of percent energy decrease between aorta outlet and inlet vs (a) pulse frequency and (b) stroke volume per cycle, separated by aorta length, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure.

Figure 4.10, Figure 4.11, and Figure 4.12 compares percent energy change, *ΔE*, vs pulse frequency for experimental sets with varying backpressure, chamber-pressure, and material. Some experimental sets with too many removed data points (as discussed in Section 3.7) are not included. The full list of figures for *ΔE* vs pulse frequency is included in Appendix L. In general, for both aorta materials between the 60-100 bpm cases, a decrease in *ΔE* can be seen. For backpressure between 0-3.71 kPa, the 150 bpm cases follow the decrease. For the Dragon Skin case at backpressure of 1500-15.66 kPa, the *ΔE* appears to plateau, increase, or decrease depending on experimental parameters. However, there is not enough remaining cases to be conclusive. Ecoflex did not have enough experimental cases to compare. Section 4.5.3 discuss the limitations of the 150 bpm case.

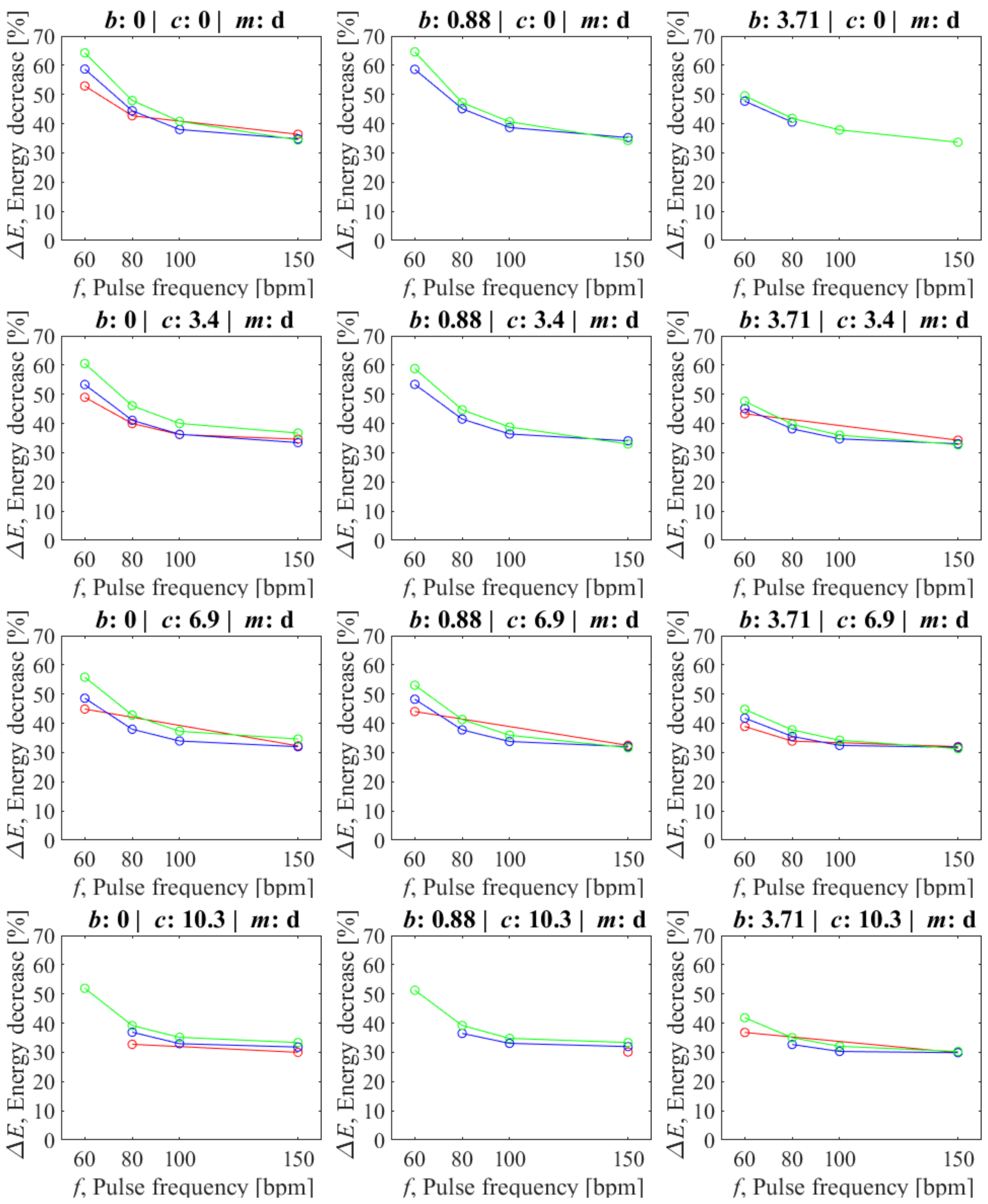


Figure 4.10: Plots of percent energy decrease between aorta outlet and inlet vs pulse frequency. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa] *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

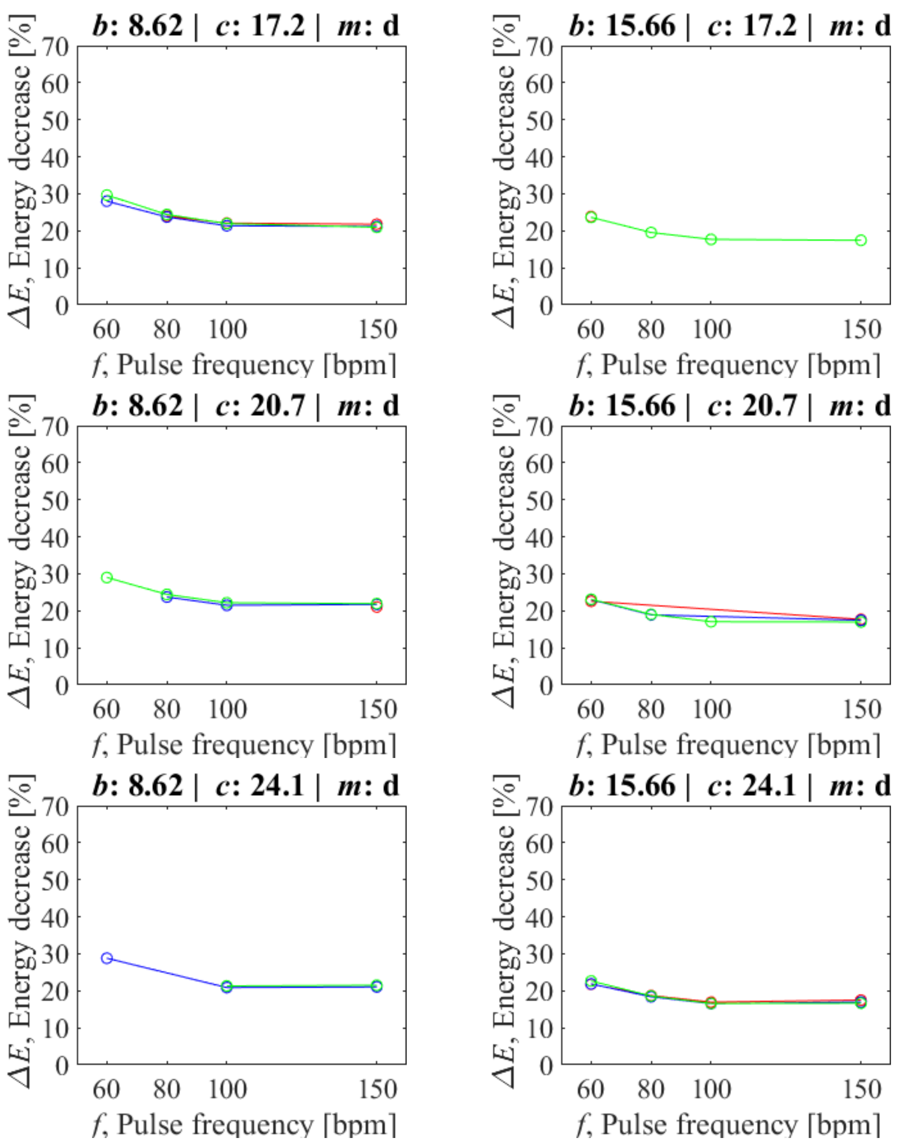


Figure 4.11: Plots of percent energy decrease between aorta outlet and inlet vs pulse frequency. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa] *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

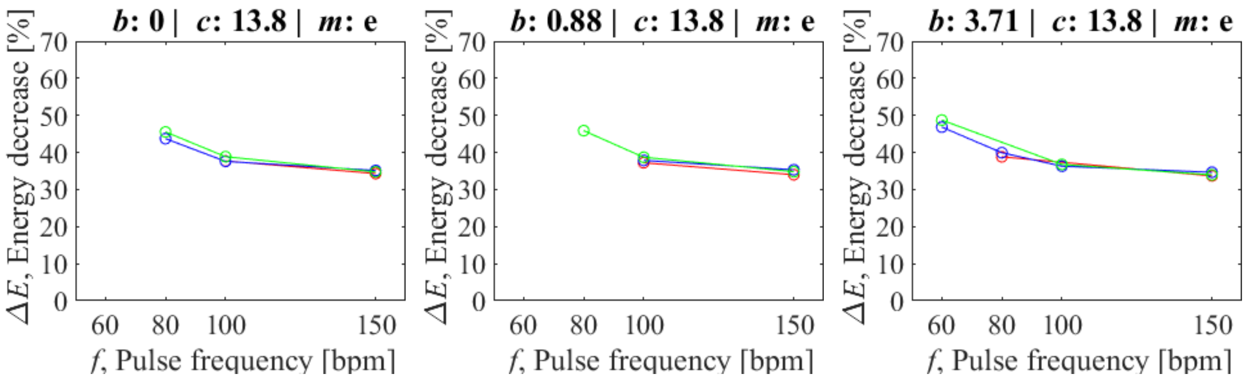


Figure 4.12: Plots of percent energy decrease between aorta outlet and inlet vs pulse frequency. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa] *m*: material; e: Ecoflex; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

Figure 4.13 shows plots of *ΔE* against aorta length for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure. Plots for the four pulse frequency/stroke volume cases are shown. Within the 0 kPa backpressure case, there is a much greater range in *ΔE* between the pulse frequency/stroke volume cases, especially when using the 140 mm aorta. The same trends are seen in the 15.66 kPa backpressure case, but with a much lower range in *ΔE*.

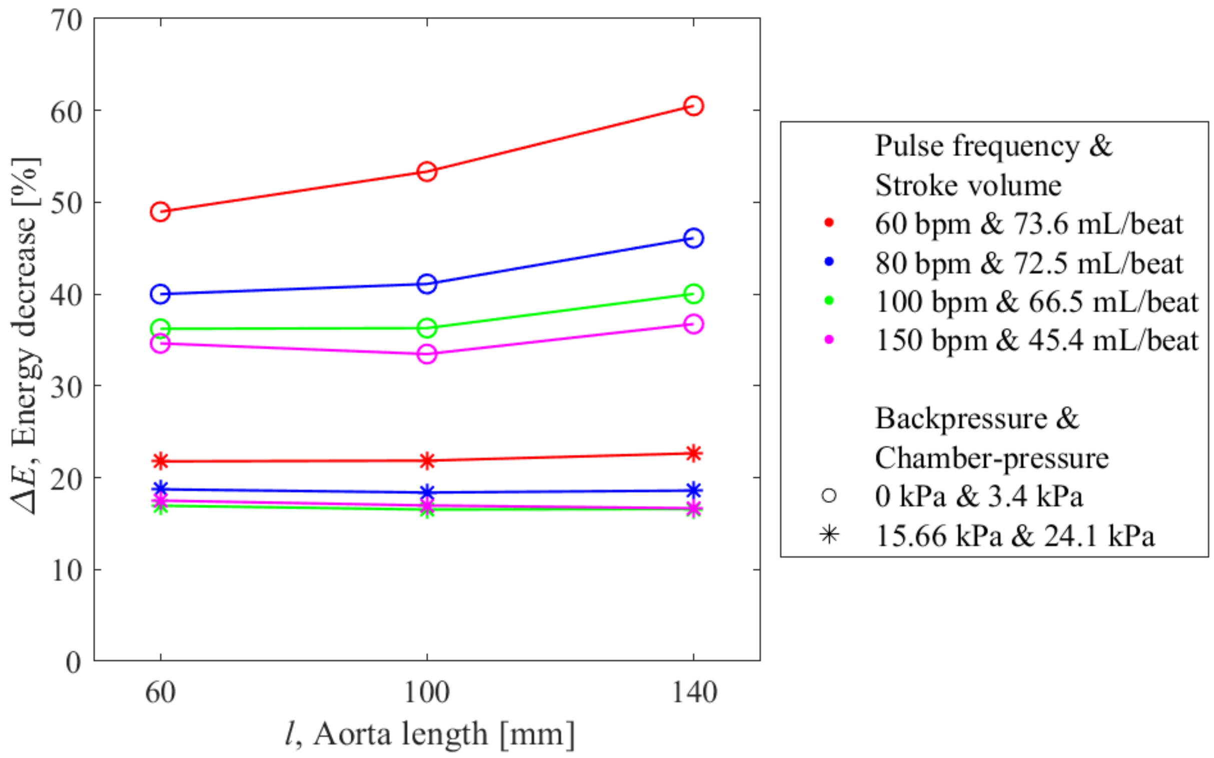


Figure 4.13: Plots of percent energy decrease between aorta outlet and inlet vs aorta length, separated by pulse frequency/stroke volume, for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure.

4.2.3. Discussion

For *E*, pump energy per cycle plots, there is a clear increase in backpressure between the highest backpressure cases at 15.66 kPa and the lowest backpressure cases at 0 kPa. When examining all experimental plots together in Figure 4.1, backpressure upwards of 3.71 kPa causes a sharp increase pump workload. Within the same backpressure, a longer aorta will lead to a lower pump energy for all pump pulse frequency/stroke volume conditions, as shown by Figure 4.3. The effects of aorta length on pump energy appears to be more prominent in lower pulse frequencies, as the energy value range is larger in 60 bpm, 73.6 mL/beat, than in 150 bpm, 45.4 mL/beat. However, as shown by Figure 4.4, the effect of aorta length on pump workload is minimal in comparison to the effect of backpressure. Figure 4.5 show an increase in pump energy with increasing chamber-pressure as well, with a significant jump from the 0 kPa backpressure to the 15.66 kPa backpressure conditions. The effect of aorta length across all chamber pressures show the longest aorta at 140 mm leads to the lowest pump workload, the effects of which are seen more prominently in lower pulse frequency conditions. When viewing all experimental parameters, backpressure is the dominant effect in pump workload. A lower backpressure will lead to the largest decrease in pump workload. The other experimental parameter effects are more prominent in the lower backpressure conditions at a lower heart pulse frequency. Overall, a system with a low backpressure and a longer aorta under the lowest chamber-pressure will provide the lowest pump workload for all pulse frequency/stroke volume pump conditions.

There is a greater range of ΔE at 0 kPa backpressure conditions than 15.66 kPa. There is a trend of decreasing ΔE with decreasing backpressure, as shown by Figure 4.6 and Figure 4.7. There is a much greater decrease in ΔE with respect to chamber-pressure at 0 kPa backpressure conditions than at 15.66 kPa. Figure 4.9 shows that for the same aorta length, lower pulse frequency/higher stroke volumes have the greatest range in ΔE. For the same pulse frequency/stroke volume, the longest aorta,140 mm, presented the greatest decrease in energy between the aorta outlet and aorta inlet, ΔE, as shown by Figure 4.13. Overall, the highest decrease in pump energy at the aorta outlet from the aorta inlet occurs with low backpressure, low chamber-pressure, and a longer aorta, for all pulse frequency/stroke volume pump conditions.

4.3. Impact of Aorta Properties on Aorta Elastic Response

The average maximum percent diameter distension is found at the center of the tube for a length of 200 pixels (Figure 4.14) and averaged over 8 pump cycles. The MATLAB code to calculate maximum percent diameter distension is included in Appendix F.

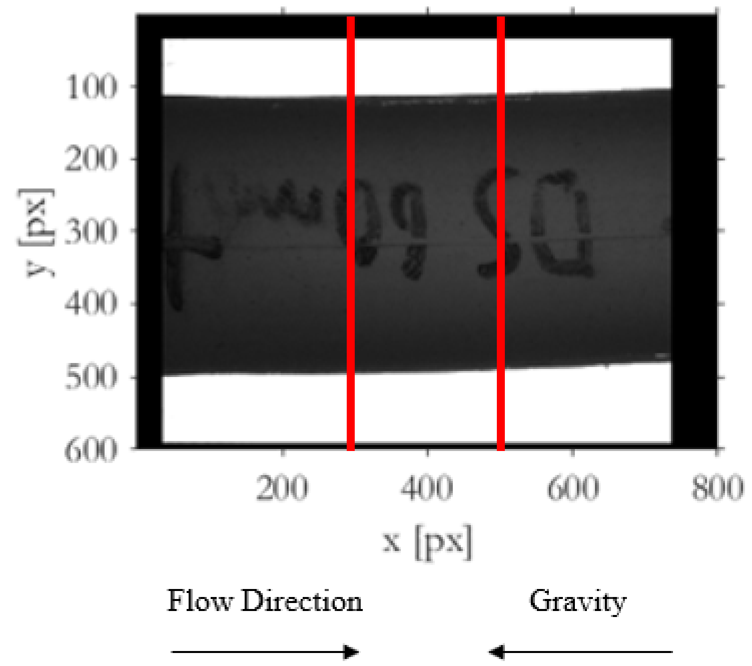


Figure 4.14: Cropped image used in image processing. Maximum distension from region *x* = 300-*x* = 500, highlighted in red, is averaged.

Figure 4.15 and Figure 4.16 show plots of maximum percent diameter distension, $\frac{\Delta D}{D}$, versus system backpressure, for all aorta lengths. All cases of $\frac{\Delta D}{D}$ vs backpressure with sufficient data points to be plotted can be found in Appendix O. As discussed in Section 3.7, for each backpressure case, there are data points for only the specific range of chamber-pressures the aorta can distend and retract without wall-contact or collapse. The 100 and 140 mm aorta cases show a constant $\frac{\Delta D}{D}$ from 0-3.71 kPa backpressure, and the 140 mm aorta at 6.9 kPa chamber pressure show a decrease in distension at -8.62 kPa backpressure for 60 and 80 bpm. For 60 mm aorta cases with more than one data point, $\frac{\Delta D}{D}$ trends fluctuate. Increasing, decreasing, and constant trends are all observed. More data points are required to make a concrete conclusion regarding the relationship between backpressure and $\frac{\Delta D}{D}$. Future investigations utilizing a larger pressure-chamber will help alleviate the issue of experimental case removal.

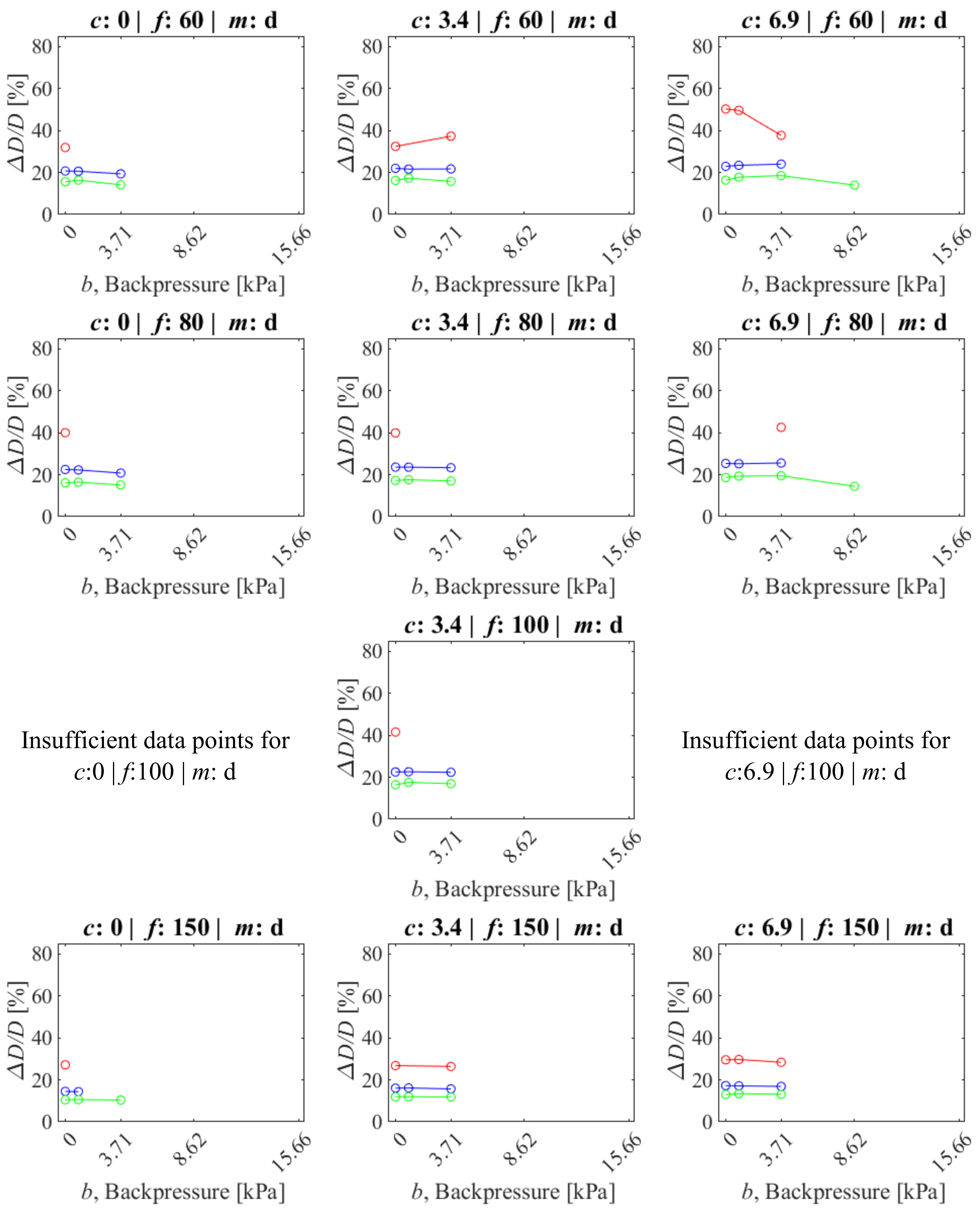


Figure 4.15: Maximum percent diameter distension, $\frac{\Delta D}{D}$, vs backpressure. Chamber-0-6.9 kPa. *c*: chamber-pressure [kPa]; *f*: pulse frequency [bpm]; *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

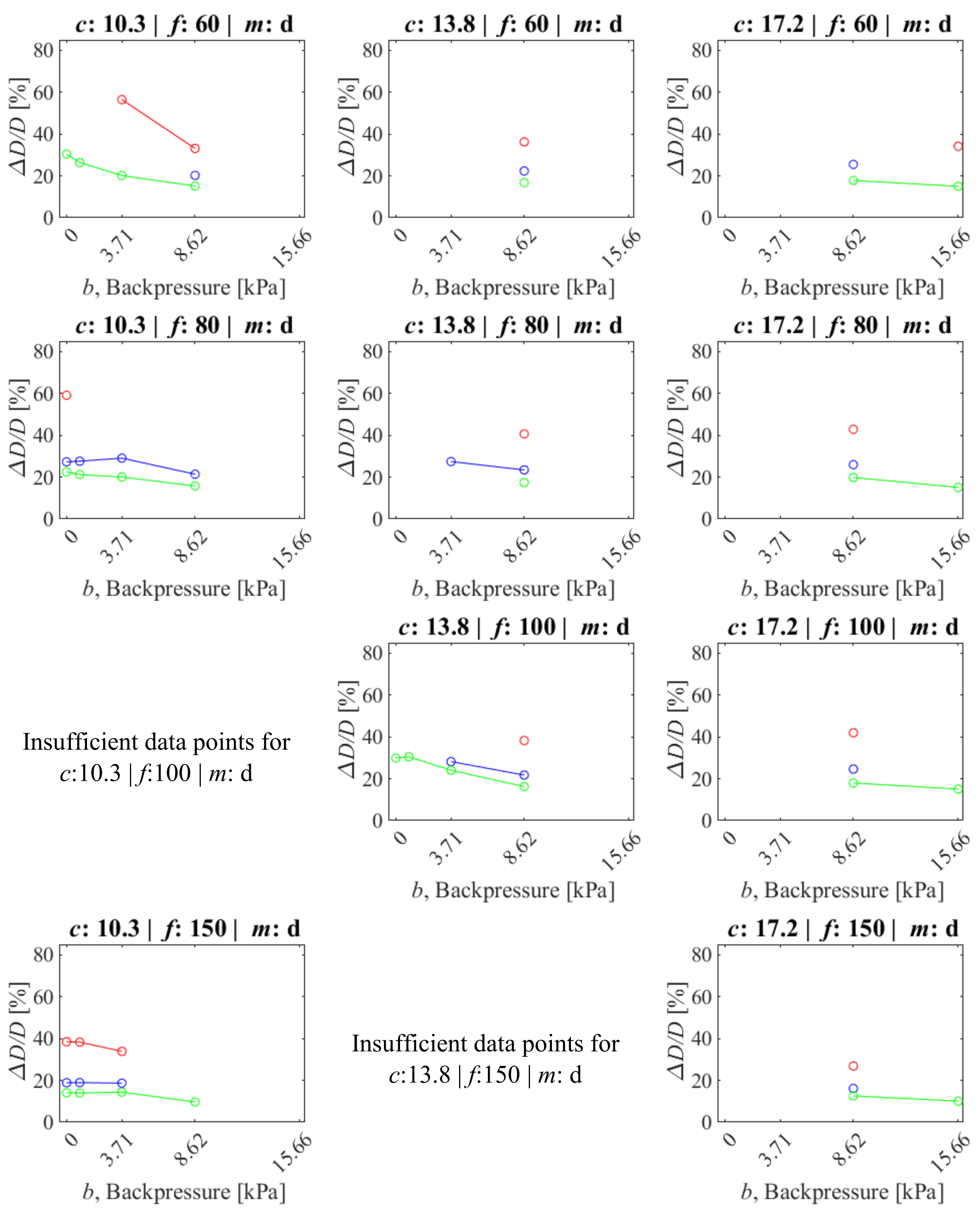
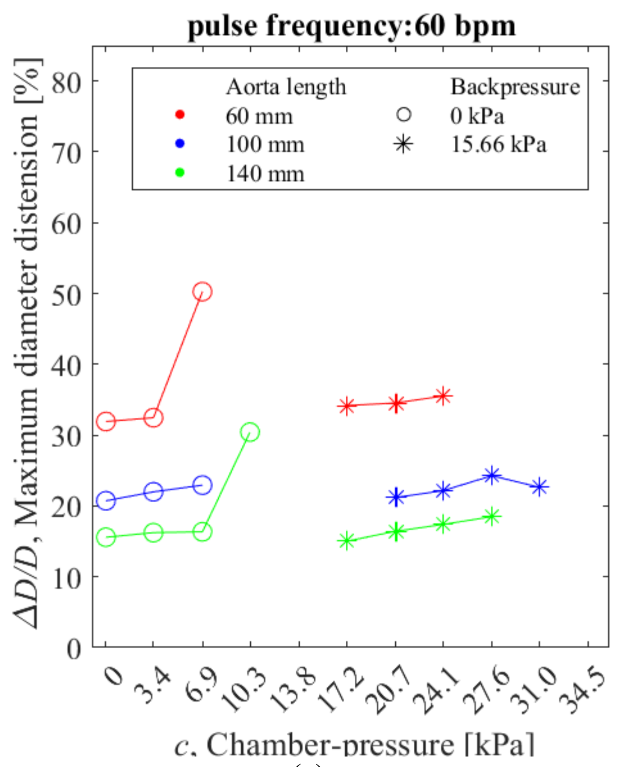
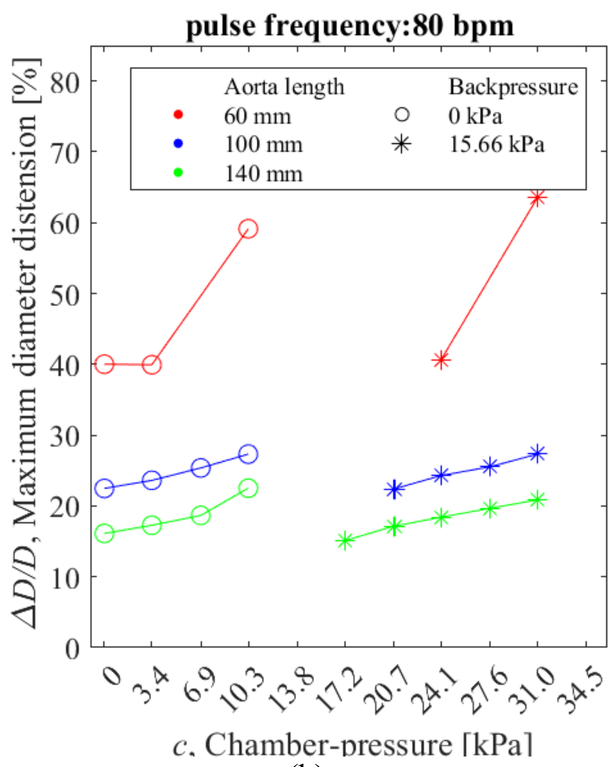


Figure 4.16: Maximum percent diameter distension, $\frac{\Delta D}{D}$, vs backpressure. Chamber-pressure 10.3-17.2 kPa.
c: chamber-pressure [kPa]; *f*: pulse frequency [bpm]; *m*: material; d: Dragon Skin; Aorta lengths [mm] of red: 60; blue: 100; green: 140.

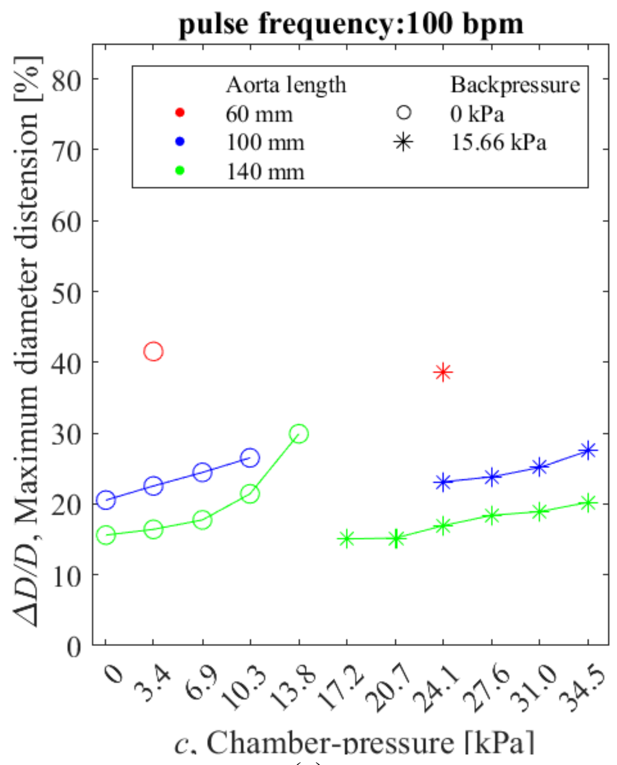
Figure 4.17 shows plots of maximum percent diameter distension, $\frac{\Delta D}{D}$, against chamber-pressure, separated by aorta lengths, for heart pump pulse frequencies of 60, 80, 100, and 150 kPa, at backpressures 0 and 15.66 kPa. There is a trend of increasing percent distension with increasing chamber-pressure. This trend is due to a decrease in the aorta minimum diameter with increasing chamber-pressure, as shown in Figure 4.18. Figure 4.18 shows the plot of both the maximum and minimum diameters of the aorta changing with chamber-pressure. As discussed in Section 3.2.1, the maximum and minimum aorta diameter is calculated by averaging the maximum and minimum diameter found over 8 pump cycles. This indicates that although a higher chamber-pressure causes a smaller diameter aorta, the maximum diameter, while still decreasing with increasing chamber-pressure, is less affected. Figure 4.18 also shows a similar minimum diameter between the different aorta lengths. However, the maximum diameters have a greater difference between the different lengths. An increase in aorta length resulted in a decrease in aorta maximum distension. This agrees with the trend seen in Figure 4.17, where an increase in aorta length led to a decrease in $\frac{\Delta D}{D}$.



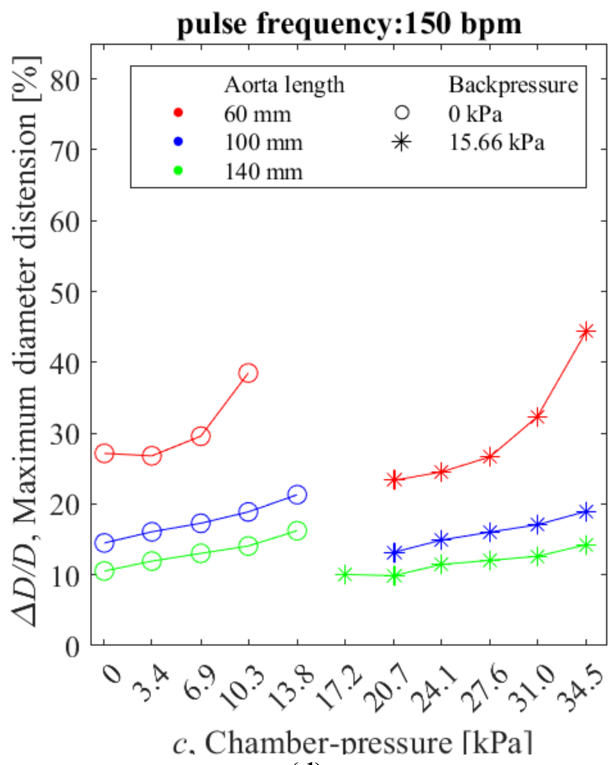
(a)



(b)



(c)



(d)

Figure 4.17: Plot of percent diameter distension vs chamber-pressure for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Material: Dragon Skin.

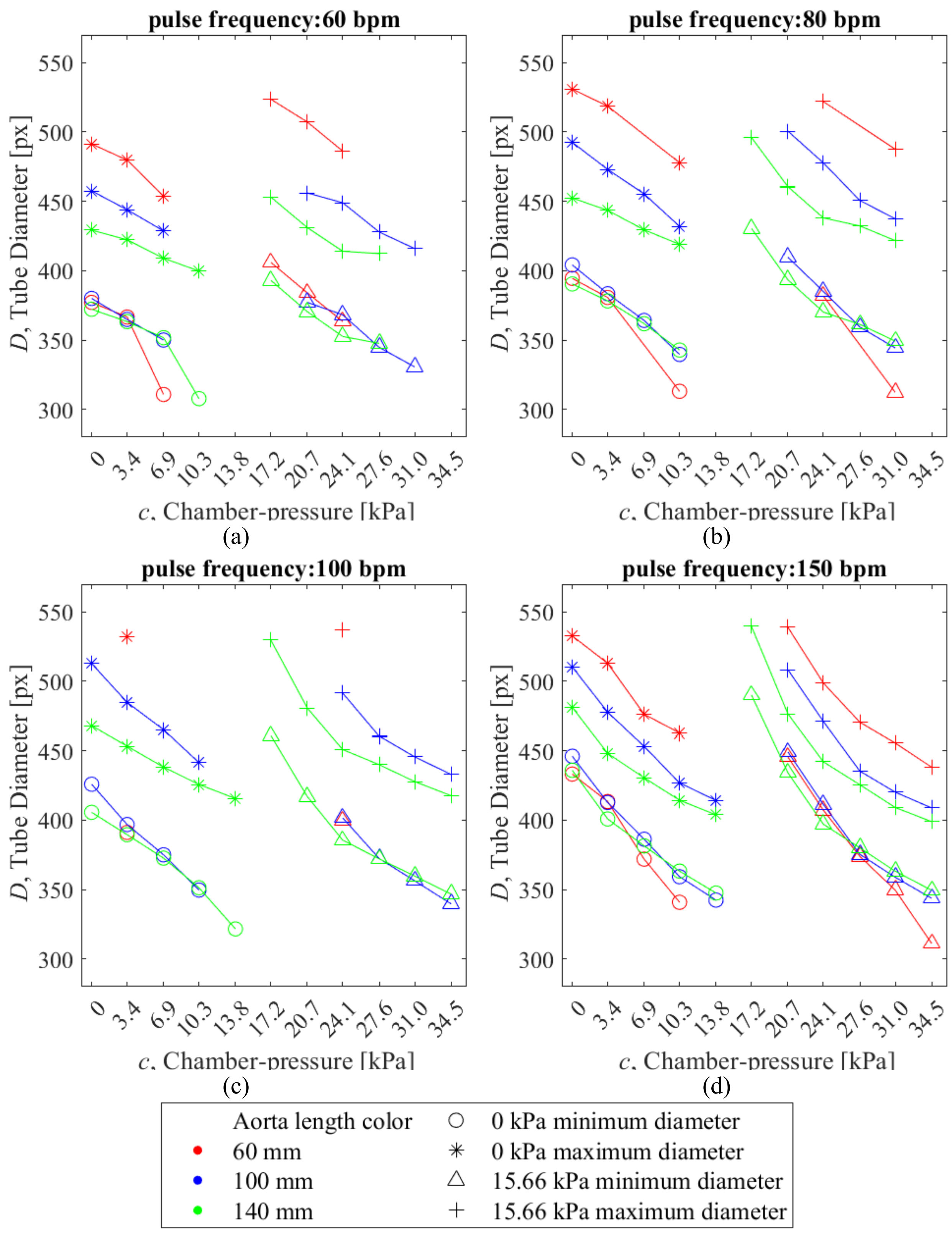


Figure 4.18: Plot of max and min tube diameter vs chamber-pressure for two sets of backpressure and chamber-pressure cases at (a) 60, (b) 80, (c) 100, and (d) 150 bpm. Material: Dragon Skin.

Figure 4.19 shows plots of maximum percent diameter distension vs aorta length, separated by stroke volume, for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure data sets. Within the same stroke volume per cycle, an increase in aorta length led to a decrease in the percent maximum distension of the aorta. There is an inverse relationship between compliant tube length and amount of fluid “stored” during its distension. A shorter tube length may store more fluid than a longer tube within the same material length. This agrees with the trends seen in Figure 4.17 and, Figure 4.18,

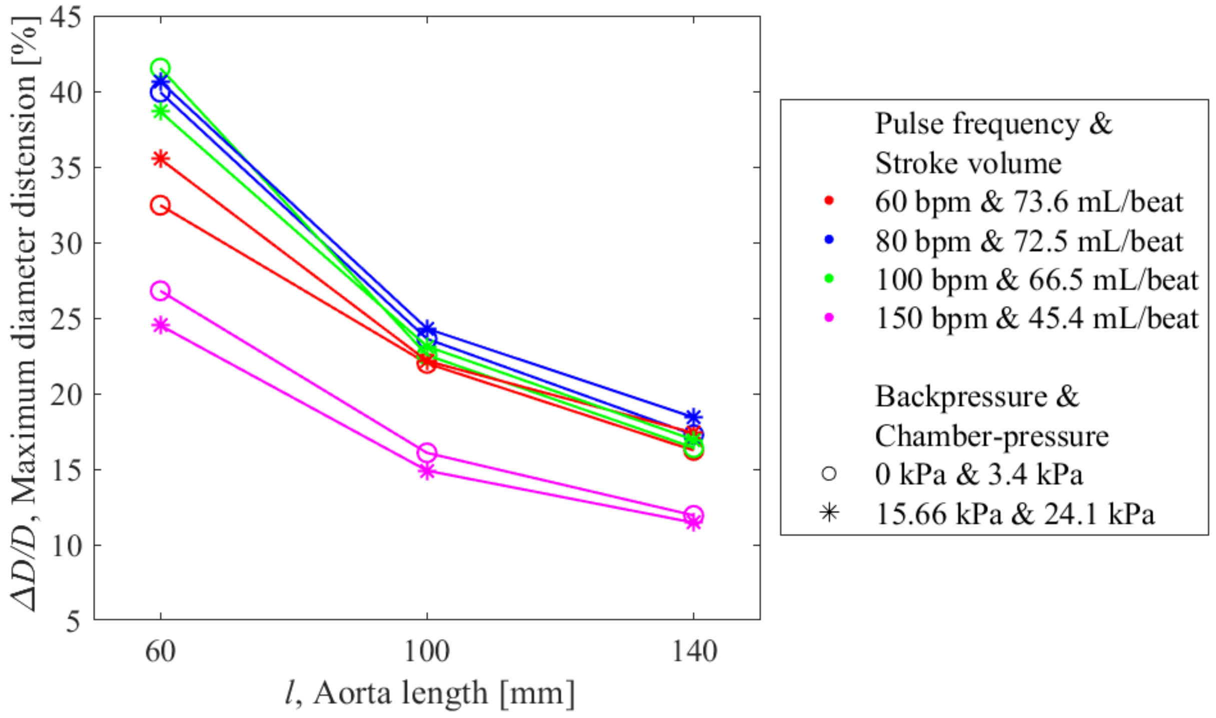


Figure 4.19: Plot of maximum percent diameter distension vs aorta length separated by stroke volume, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure.

Figure 4.20 and Figure 4.21 maximum percent diameter distension against aorta length for experimental sets with varying backpressure, chamber-pressure, and material. Some experimental sets with too many removed data points (as discussed in Section 3.7) are not included. The full list of figures for percent diameter distension vs aorta length is included in Appendix O. A decrease in maximum distension with increasing aorta length is seen across all data. Excluding the 60 mm aorta length case, the general trend of increasing stroke volume corresponding to an increase in maximum percent diameter distension is seen across all data sets. Limitations of the 60 bpm case are discussed in Sections 4.5.2 and 4.5.3.

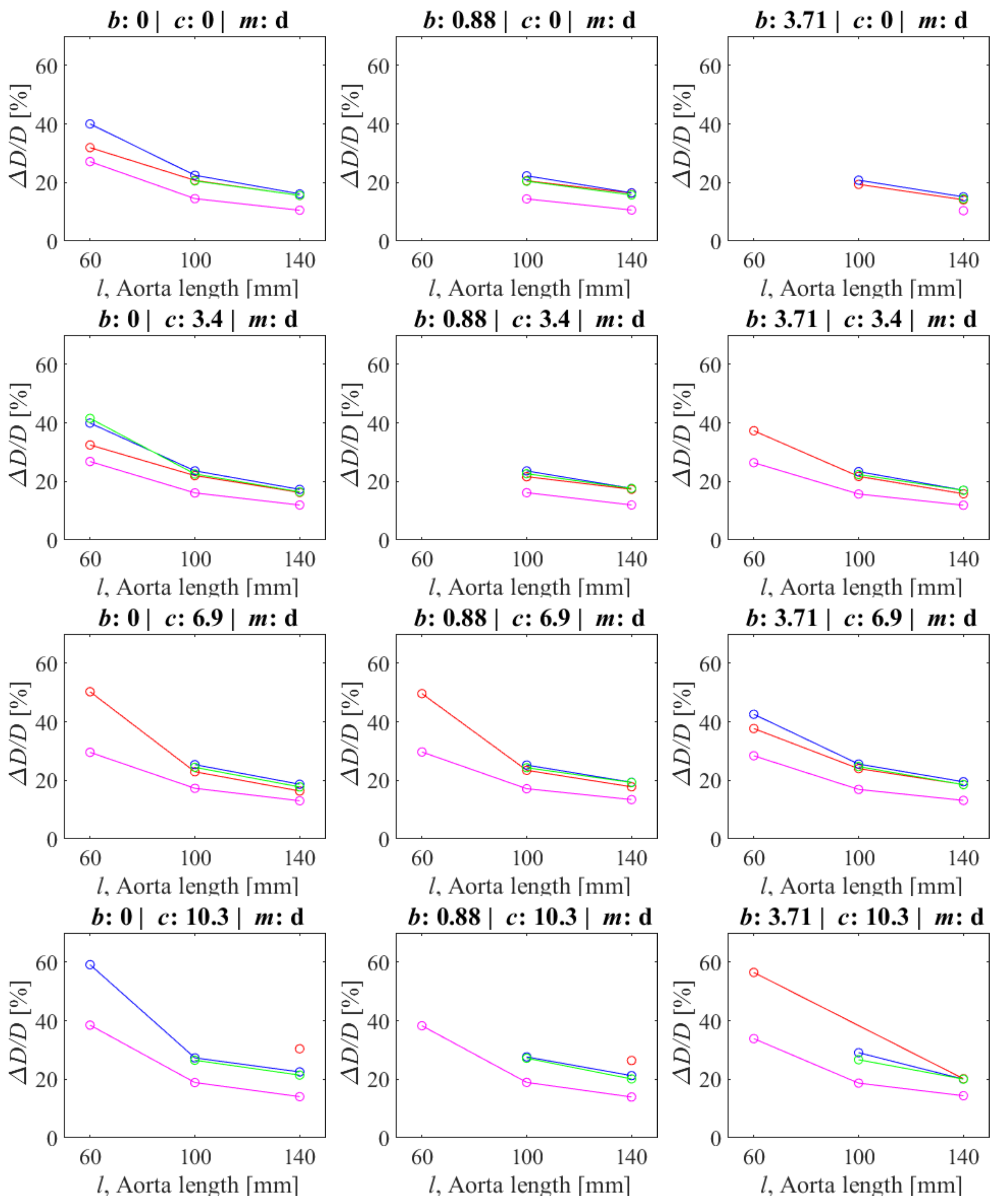


Figure 4.20: Plot of maximum percent diameter distension vs aorta length, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; d: Dragon Skin; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4.

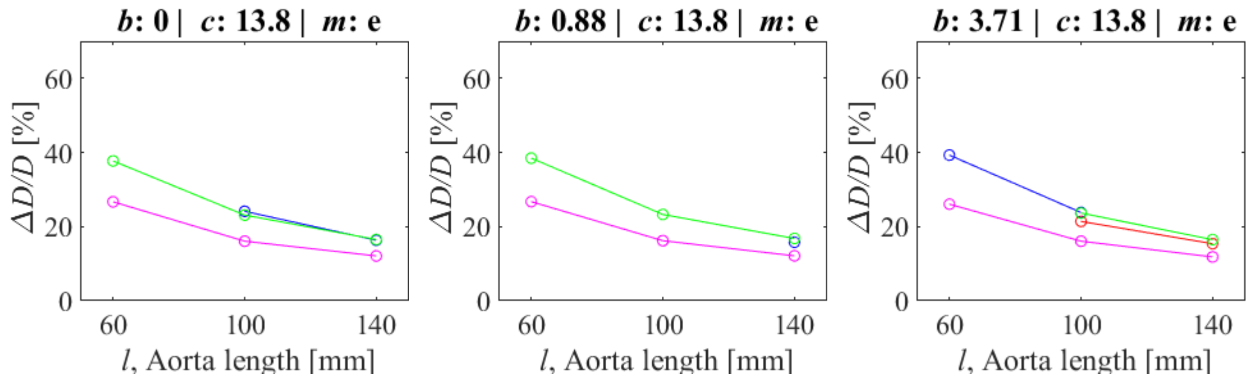
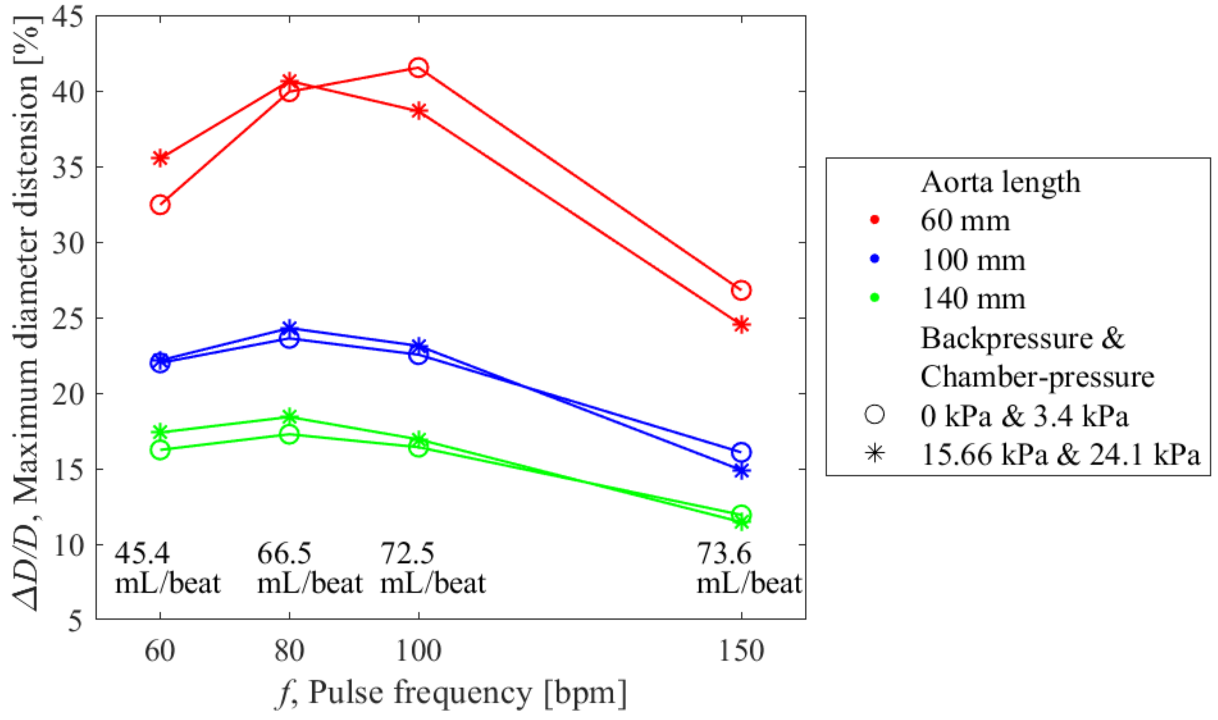
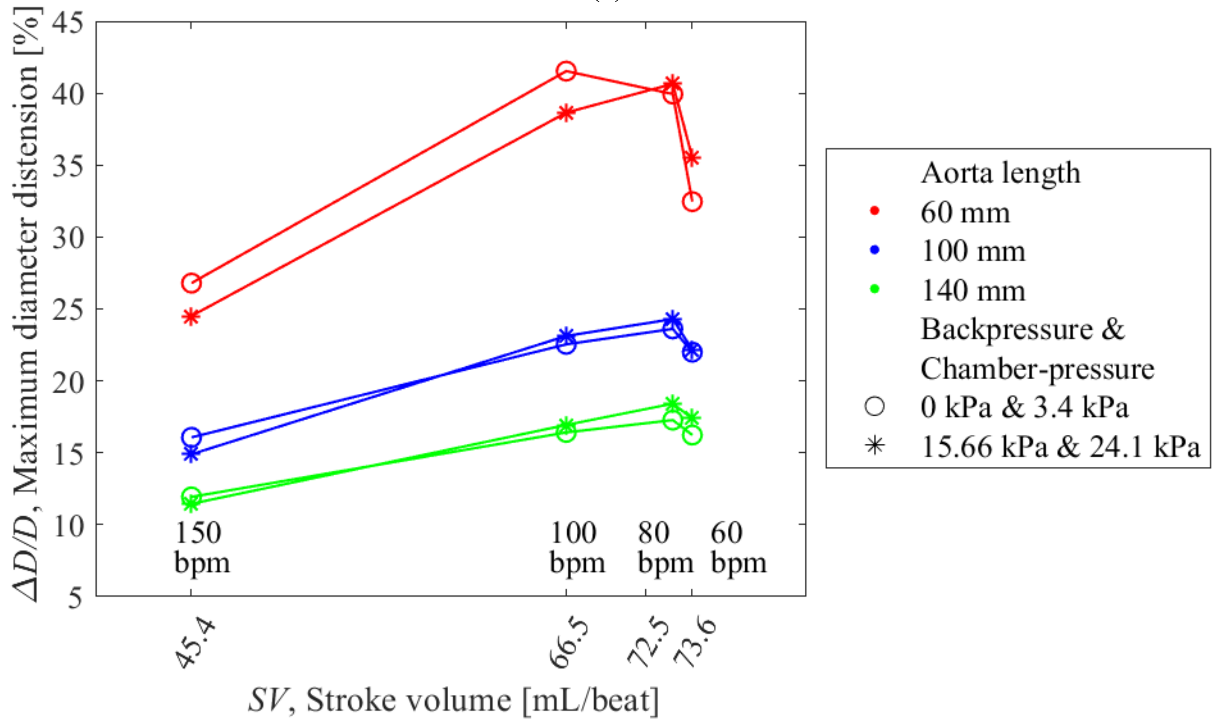


Figure 4.21: Plot of maximum percent diameter distension vs aorta length, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; e: Ecoflex; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4.

Figure 4.22 shows plots of maximum percent diameter distension vs (a) stroke volume and (b) pulse frequency, separated by aorta length, for the same experimental conditions. Between the stroke volume cases. excluding the 60 mm aorta length data points at 0 kPa backpressure and 3.4 kPa chamber-pressure, there is an increase in maximum percent diameter distension with increasing stroke volume up to 72.5 mL/beat, before decreasing at 73.6 mL/beat. The 60 mm aorta at 0 kPa backpressure and 3.4 kPa chamber-pressure experience an increase in maximum distension up to 66.5 mL/beat, before decreasing with increasing stroke volume. The opposite trend exists when examining the change in maximum distension with respect to pulse frequency, since a decrease in stroke volume corresponds to an increase in pulse frequency in the experimental set up. Limitations of the 60 mm aorta are discussed in Section 4.5.4.



(a)



(b)

Figure 4.22: Plot of maximum percent diameter distension vs (a) stroke volume and (b) pulse frequency, separated by aorta length, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure.

4.3.1. Discussion

For each backpressure, there is a working range of chamber-pressures within which the aorta will not bulge to the point of wall contact with the pressure-chamber nor collapse under chamber-pressure conditions that are too high. More data is required to make a concrete conclusion regarding the impact of backpressure on maximum percent diameter distension, $\frac{\Delta D}{D}$. A tube under higher backpressure and chamber-pressure will have similar distension values to a tube under lower backpressure and chamber pressure conditions, this is evident in Figure 4.17, where the $\frac{\Delta D}{D}$ value ranges less between the two backpressure conditions than between chamber-pressures or aorta lengths. Both the minimum and maximum aortic diameters decrease with increasing chamber-pressure. Conversely. increasing chamber-pressure increases $\frac{\Delta D}{D}$. This is due to the chamber-pressure having a greater impact on the minimum tube distension than the maximum tube distension. Minimum tube diameter remains fairly constant but varies greatly for the maximum tube diameter across aorta lengths. Overall, decreasing aorta length will increase $\frac{\Delta D}{D}$ across all chamber-pressure, backpressure, and pulse frequency/stroke volume. Figure 4.19-Figure 4.22 show aorta length to have the greatest impact on $\frac{\Delta D}{D}$, with the largest distension difference between different lengths than other experimental parameters.

4.4. Impact of Aorta Elastic Response on Pump Energy

4.4.1. Pump Energy at Aorta Inlet and Aorta Outlet, *E*

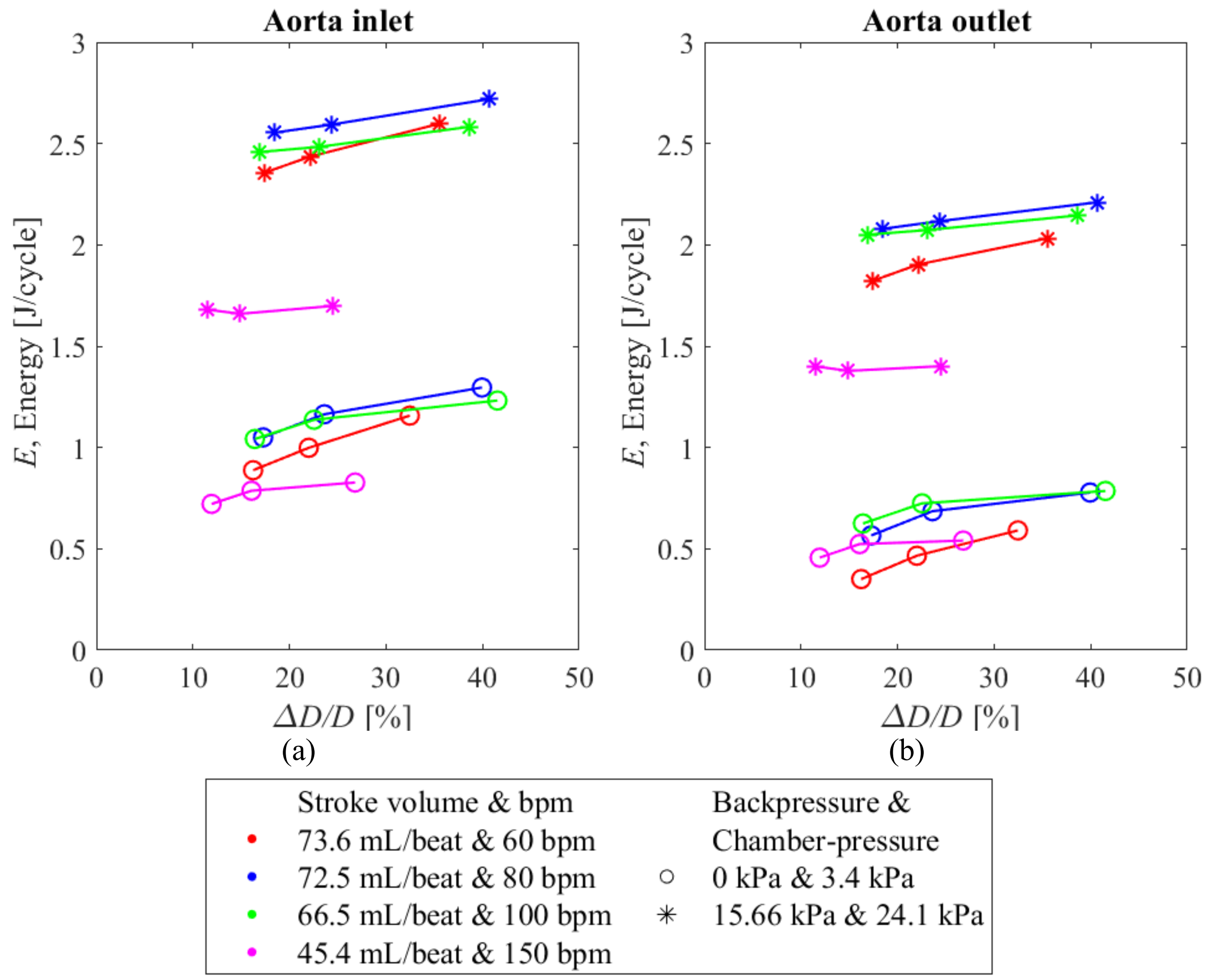


Figure 4.23: Plot of pump energy per cycle vs maximum percent diameter distension at (a) aorta inlet and (b) aorta outlet, separated by stroke volume and its corresponding pulse frequency.

Figure 4.23 shows plots of pump energy per cycle, E, versus aorta percent maximum distension, $\frac{\Delta D}{D}$. The plots are separated by stroke volume, with each corresponding stroke volume's pulse frequency listed as well. Experimental parameters of 0 kPa backpressure and 3.4 kPa chamber pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure are shown. There is a general increase in E with increasing $\frac{\Delta D}{D}$ among the same stroke volume/pulse frequency conditions. Higher backpressure and chamber pressure conditions will result in a higher E per $\frac{\Delta D}{D}$. Interestingly, at 15.66 kPa backpressure and 24.1 kPa chamber-pressure, the 45.4 mL/beat and 150 bpm case has much lower E values per $\frac{\Delta D}{D}$ than the rest of the experimental parameters, but at 0 kPa backpressure and 3.4 kPa chamber-pressure, the E values are closer in range to the rest. There appears to be much overlap in E among the pulse frequency/stroke volume cases otherwise.

4.4.2. Percent Energy Decrease from Aorta Inlet to Aorta Outlet, *ΔE*

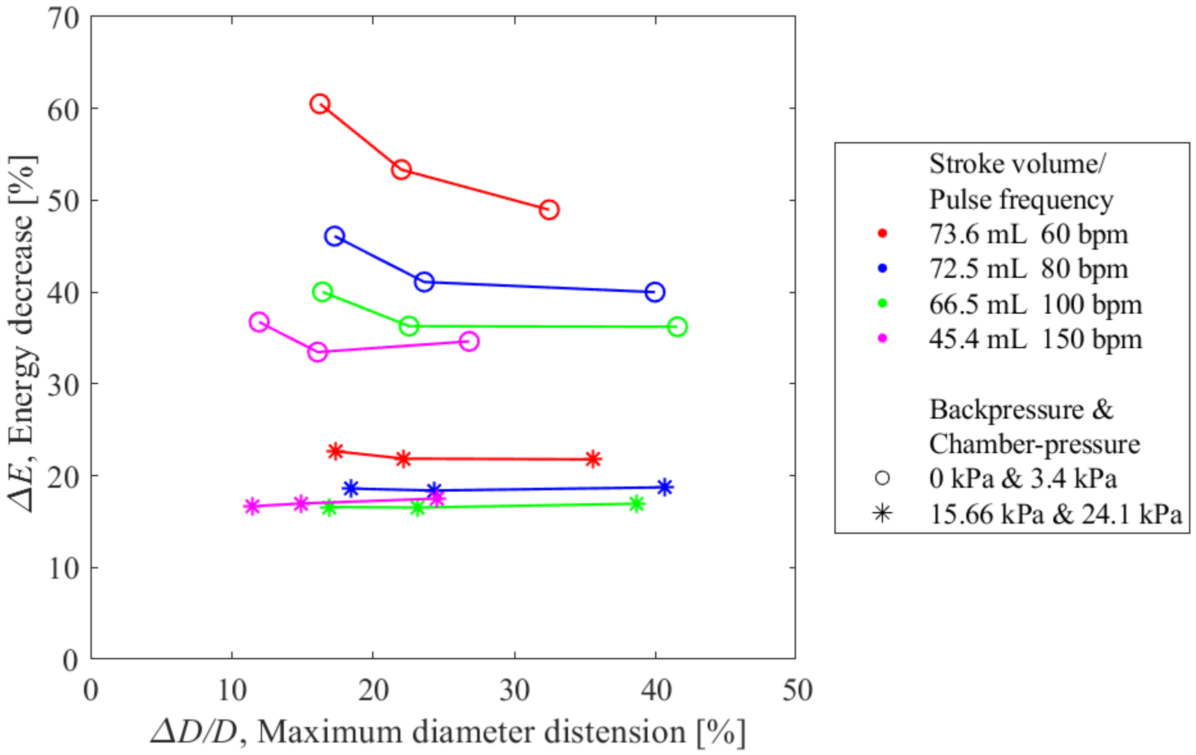


Figure 4.24: Plot percent energy change decrease between aorta outlet and inlet vs maximum percent diameter distension, separated by stroke volume and its corresponding pulse frequency, for 0 kPa backpressure and 3.4 kPa chamber-pressure and 15.66 kPa backpressure and 24.1 kPa chamber-pressure.

Figure 4.24 shows plots of percent energy decrease at the aorta outlet from the aorta inlet, *ΔE*, vs aorta percent maximum distension, $\Delta D/D$, separated by stroke volume per cycle with its corresponding pulse frequency (Figure 4.24), for 0 kPa backpressure and 3.4 kPa chamber-pressure, and 15.66 kPa backpressure and 24.1 kPa chamber-pressure data sets. Within the same stroke volume per cycle, an increase in maximum distension led to a decrease in *ΔE*. Between the stroke volume cases, an increase in stroke volume per cycle led to a decrease in *ΔE*.

The same trends can be seen across experimental sets with enough data points to allow comparison, as shown in Figure 4.25 for Dragon Skin and Figure 4.26 for Ecoflex aortas. The full range of figures can be found in Appendix Q. This indicates that higher distension results in higher losses through the aorta, as more fluid is stored within the distension in the pump out-stroke. Lower backpressure and chamber-pressure conditions result in higher ΔE with respect to $\frac{\Delta D}{D}$. There is also a larger range of ΔE between the stroke volume plots within the lower backpressure and chamber-pressure conditions.

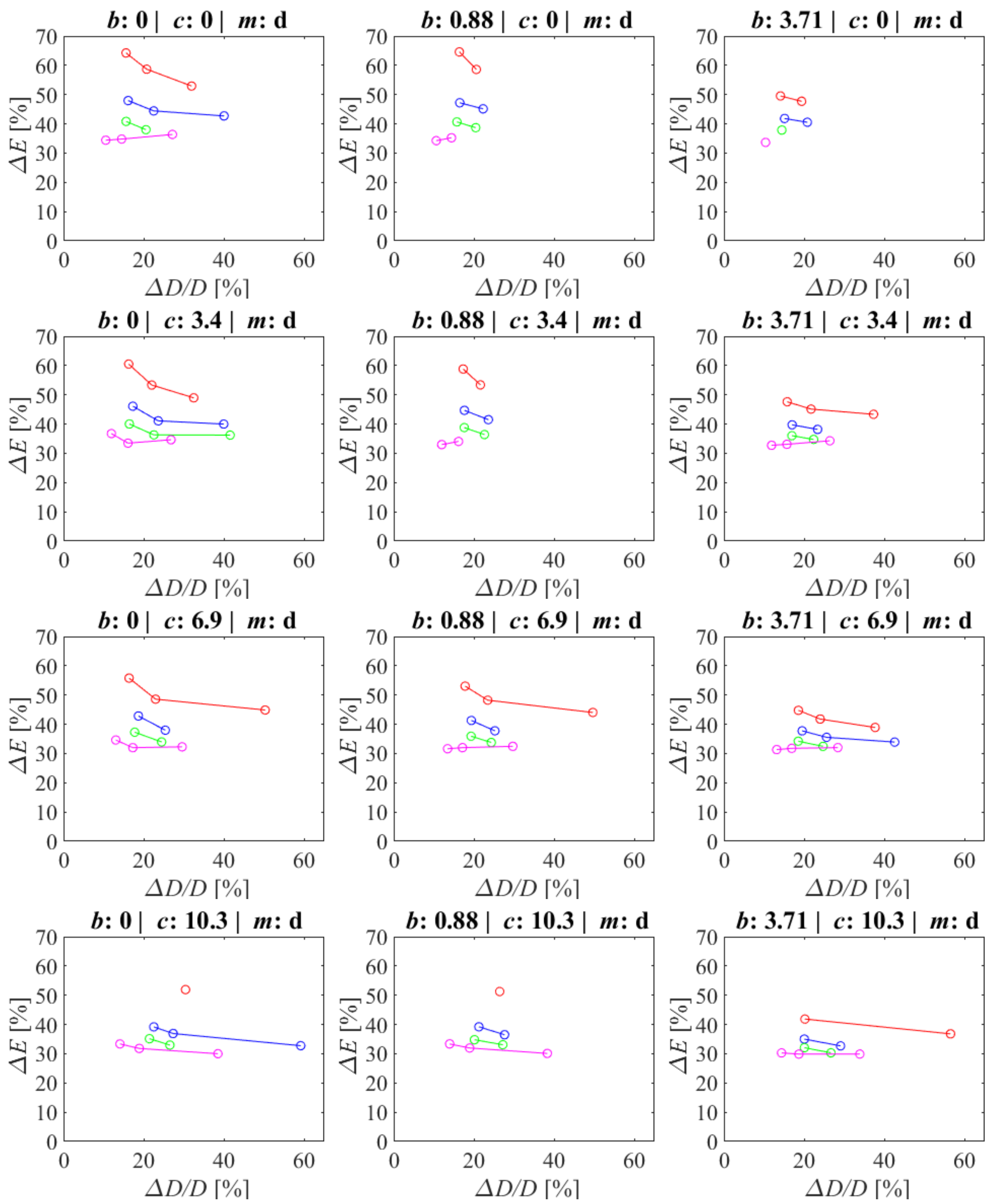


Figure 4.25: Plot percent energy change decrease between aorta outlet and inlet vs maximum percent diameter distension, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; d: Dragon Skin; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4.

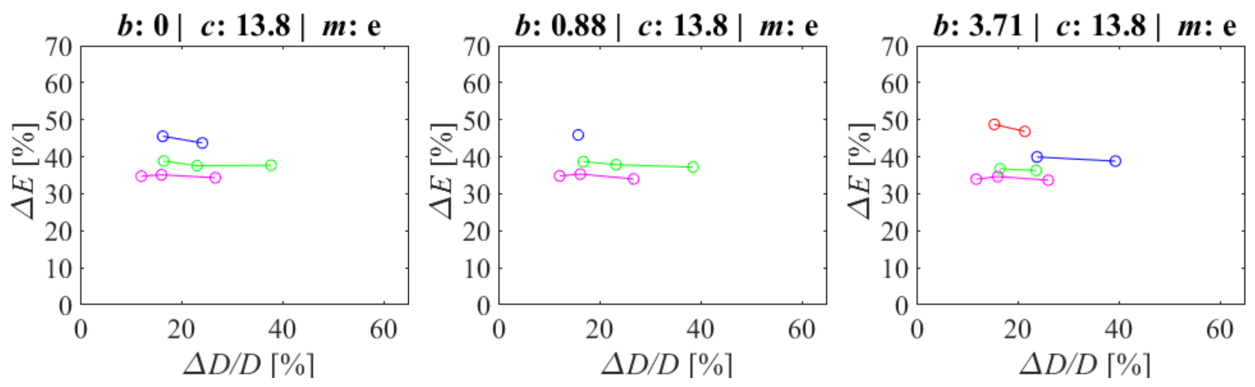


Figure 4.26: Plot percent energy change decrease between aorta outlet and inlet vs maximum percent diameter distension, separated by stroke volume. *b*: back-pressure [kPa]; *c*: chamber-pressure [kPa]; *m*: material; e: Ecoflex; Stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4.

4.4.3. Discussion

E generally increased with increasing $\frac{\Delta D}{D}$ within the same stroke volume and its corresponding pulse frequency, backpressure, and chamber-pressure conditions. The 15.66 kPa backpressure and 24.1 kPa chamber pressure cases have higher E values than the 0 kPa backpressure and 3.4 kPa chamber-pressure cases. ΔE decreased with increasing $\frac{\Delta D}{D}$ within the same stroke volume, backpressure, and chamber-pressure conditions. This indicates that higher distension result in lower energy decrease through the aorta, and higher pump workload.

ΔE decreased with increasing $\frac{\Delta D}{D}$ for the same stroke volume/pulse frequency, backpressure, and chamber-pressure conditions. This indicates a higher tube distension will result in lower energy decrease at the aorta outlet from the aorta inlet. ΔE decreased with decreasing stroke volume/increasing bpm for similar $\frac{\Delta D}{D}$. Higher values of ΔE, as well as a larger distribution of ΔE are seen in the 0 kPa backpressure and 3.4 kPa chamber-pressure case. This suggests the losses through the aorta are more affected by tube distension at lower backpressure/chamber-pressures.

4.5. Limitations

4.5.1. Impact of Discarding Specific Experimental Cases

As discussed in 3.7, due to the limiting size of the pressure-chamber, cases where the aorta distension causes contact with the pressure-chamber walls are removed from the experimental results during image processing. Due to the distensibility of the silicone material, cases where the aorta collapses with high chamber pressure are removed as well. This results in incomplete data sets for comparison between cases. Experimental cases using an Ecoflex aorta are heavily affected by case removal. The Ecoflex silicone is softer, with a lower Young's modulus than the Dragon Skin silicone.

Only the two data sets with 0 kPa backpressure and 3.4 kPa chamber pressure, and 15.66 kPa backpressure and 24.1 kPa chamber pressure have complete data points for comparison. From the figures displaying all experimental cases, many are reduced to one or two data points for each specific set. For example, in Figure 4.25 for the case of 0.88 kPa backpressure and 3.4 kPa chamber-pressure has only two data sets for each stroke volume per cycle. The 3.71 kPa backpressure and 0 kPa chamber-pressure case has only one data point for the 45.4 and 66.5 mL/cycle cases. However, when there are more than three data points for each comparison case, the same common trends can be seen.

4.5.2. Tube Rest State

A small drift in the aorta minimum diameter could be seen during data capture, where the minimum diameter of the active aorta under pump operation is higher than its minimum diameter when the pump is at rest. At higher chamber-pressures, the aorta rest state may be collapsed, but regains roundness once the pump initiates. To account for this, the percentage distension is calculated using an average of the aorta during its active state, at its lowest diameter during each cycle. The 60 bpm cases are heavily affected by both wall contact with the pressure-chamber, and aorta collapse. This is due to the higher stroke volume pushed through the aorta for the 60 bpm cases. Thus, many experimental cases involving using 60 bpm were removed.

4.5.3. Pulse Frequency and Stroke Volume

With the experimental set-up used, increasing the pulse frequency of the pump requires lowering its stroke volume per cycle output. This decrease in stroke volume is less apparent in the 60, 80, and 100 bpm cases, which have 73.6, 72.5, and 66.5 mL/cycle respectively. However, in the 150 bpm case, the stroke volume decreased to almost 60% of the original 60 bpm case, at 45.4 mL. Therefore, all pulse frequency case comparison must also take into consideration its corresponding stroke volume. There is no direct comparison of the impact on pump performance for the same stroke volume but varying pulse frequency, and vice versa.

4.5.4. 60 mm Tube Experiments

Experiments involving the 60 mm tube are the source of the most outliers in this investigation. This is most apparent in Section 4.3. A possible explanation for this is the short length of tube. The aorta is clamped at both ends to rigid tubing within the pressure-chamber. With the shortest distance between the aorta outlet and aorta inlet, the 60 mm aorta is most heavily impacted by entrance effects. Longer aorta lengths allow these effects to dissipate over larger distances. Due to its short length, the 60 mm aorta may resemble most closely to a rigid tube. A future investigation is needed to compare rigid tubing to compliant tubing for different aorta lengths. The 60 mm length aorta also appears to be most affected by chamber-pressure and backpressure conditions. It is typically the first tube to collapse with increasing chamber-pressure, as well as the first to bulge to the point of pressure-chamber wall-contact with increasing backpressure. This leads to the 60 mm aorta having the most discarded cases, as discussed earlier in Section 4.5.1.

4.6. Summary

This chapter discussed the effect of experimental parameters on pump energy and aorta elastic response, as well as the effect of aorta elastic response on pump energy. Table 4.3 compares the full results.

Table 4.3: Impact of experimental parameters on energy per cycle and percent energy decrease at aorta outlet from aorta inlet

Experimental Parameter	***E*, Energy per cycle, Pump workload**	***ΔE* ,Percent energy decrease at aorta outlet from aorta inlet**	**$\frac{\Delta D}{D}$, Maximum aorta percent distension**
Backpressure increase	Increase	Decrease	Inconclusive
Chamber-pressure increase	Increase	Decrease	Increase
Aorta length increase	Decrease	Increase	Decrease
Pulse frequency increase/ stroke volume decrease	Increase, then decrease	Decrease[2]	Increase, then decrease

[2] An exception may be the 150 bpm case. The percent energy decrease at the aorta outlet increased from 100 bpm to 150 bpm at 0 kPa backpressure but appears to plateau or decrease slightly for 15.66 kPa backpressure.

4.6.1. Impact of Experimental Parameters on Pump Energy per Cycle, E and *ΔE*

Across all experimental cycles, there is a trend of higher pump cycle energy at the aorta inlet than the aorta outlet. This is to be expected due to the losses in the aorta. Energy per cycle, *E*, and the percent energy decrease between the inlet and the outlet, *ΔE,* was calculated and compared to backpressure, aorta length, bpm, and stroke volume. Increasing backpressure corresponds to a tentative trend of increase in *E* and a decrease in *ΔE*. Increasing chamber-pressure corresponds to an increase in *E* and a decrease in *ΔE*. Increasing aorta length corresponds to a decrease in *E* and an increase in *ΔE*. Increasing pulse frequency, with its corresponding decrease in stroke volume, first leads to an increase in *E* from approximately 60-80 or 60-100 bpm, depending on aorta length, and then decrease with higher bpm/lower stroke volumes. *ΔE* decreases with increasing pulse frequency/decreasing stroke volume.

4.6.2. Impact of Aorta Properties on Elastic Response

As aorta length increases, the percent maximum distension from the tube rest state decrease across all experimental cases. The tube minimum distension, averaged over 8 pump cycles, remains similar across all aorta length. The tube maximum distension, also averaged over 8 pump cycles, clearly decrease with increasing aorta length. This indicates an inverse relationship between the amount of fluid stored during distension in the pump cycle per material length of the compliant section in the system.

The aorta maximum and minimum diameter, as well as the maximum distension, decrease with increasing chamber-pressure. This indicates that although the chamber-pressure plays an effect on the aorta diameter, the aorta distension is less affected.

An increase in stroke volume per cycle, and a corresponding decrease to pulse frequency, results in an increase in maximum percent diameter distension, $\frac{\Delta D}{D}$. As the pump pushes more fluid through the cycle, the compliant aorta distends more to accommodate. An exception to this is the 73.6 mL/cycle case, corresponding to a stroke volume of 60 bpm. When stroke volume increases from 72.5 to 73.6 mL/cycle, or decreases from 80 to 60 bpm respectively, $\frac{\Delta D}{D}$ decreases. A further exception to this is the 0 kPa backpressure and 3.4 kPa chamber-pressure experimental case, where instead the decreasing $\frac{\Delta D}{D}$ starts from 66.5 mL/cycle up to 73.6 mL/cycle, or from 100 bpm down to 60 bpm respectively.

4.6.3. Impact of Aorta Elastic Response on Pump Energy

Higher distension results in lower energy decrease, ΔE, through the aorta. Within the same stroke volume and its corresponding pulse frequency, backpressure, and chamber-pressure conditions, E generally increased with increasing $\frac{\Delta D}{D}$. Tube distension result in higher pump workload when under lower backpressure/chamber-pressure conditions. The 15.66 kPa backpressure and 24.1 kPa chamber pressure cases have significantly higher E values than the 0 kPa backpressure and 3.4 kPa chamber-pressure cases.

Higher distension results in lower energy decrease at the aorta outlet. Within the same stroke volume/pulse frequency, backpressure, and chamber-pressure conditions, ΔE decreased with increasing $\frac{\Delta D}{D}$. ΔE decreased with decreasing stroke volume for similar $\frac{\Delta D}{D}$. Since pulse frequency is inversely correlated to stroke volume, ΔE decreased with increasing pulse frequency for similar $\frac{\Delta D}{D}$. Tube distension result in higher losses through the aorta when under lower backpressure/chamber-pressure conditions. There was a larger ΔE and a larger distribution of ΔE

among the 0 kPa backpressure and 3.4 kPa chamber-pressure case than the 15.66 kPa backpressure and 24.1 kPa chamber-pressure case.

4.6.4. Overall Effect on Pump Performance

Taking into consideration the effect of all experimental parameters as a whole, for all pulse frequency/stroke volume conditions, a low backpressure with low aorta distension due to low chamber-pressure and long aorta length will have a positive impact on pump performance. The worst conditions for pump performance would be a high backpressure with high aorta distension due to high chamber-pressure and low aorta length.

Chapter 5. Conclusion and Future Work

5.1. Conclusion

The purpose of this work was to study the fluid mechanics of an EVHP system on cardiac performance. Previous work has shown that including a compliant mock aorta at the pulsatile pump exit can reduce overall pump workload. This was further investigated in this study. The EVHP system was assumed analogous to a mechanical reciprocating pump loop. An experimental test loop was developed to model the systemic portion of the medical EVHP system. The modular design of the developed mechanical test loop allows for modifications to current components and expansions to include more components as required by testing needs in the future. To test the effect of compliance, a design strategy was developed to cast compliant components such as the aorta and heart valves. This design strategy can be applied to other compliant components as EVHP testing requires in the future.

Backpressure, chamber-pressure, pulse frequency/stroke volume, aorta material, and aorta length were the system parameters varied. Fluid pressure and aorta tube distension were the results collected. Fluid pressure was measured at the pump inlet, pump outlet, aorta inlet, and aorta outlet. Aorta response was captured using a high-speed camera. These measurements were processed and used to calculated pump energy, pump energy decrease at the aorta outlet from the aorta inlet, and maximum aorta distension. The results are plotted against the varied system parameters and trends are examined.

In general, for the Dragon Skin aorta, increasing backpressure and chamber-pressure led to an increase in pump workload. Increasing aorta length led to reduced pump workload. Pulse frequency and stroke volume were connected to pump operation. Increasing pulse frequency required a decrease in stroke volume. Increasing pulse frequency/decreasing stroke volume at first led to an increase in pump workload up to approximately 80-100 bpm/down to approximately 66.5-72.5 mL/beat. From 100 bpm upwards, or 66.5 mL/beat downwards, pump workload decreased. These results are mirrored in $\frac{\Delta D}{D}$, the maximum aorta percent distension results. Backpressure effects on $\frac{\Delta D}{D}$ are inconclusive due to insufficient and conflicting data. There appears to be an inverse relationship between E and ΔE. The parameters which increase pump workload result in a decrease in pump energy at the aorta outlet from the aorta inlet. That is, parameters which increase E will decrease ΔE. This agrees with the relationships found for $\frac{\Delta D}{D}$ with both E and ΔE. Increasing tube distension will increase pump workload and decrease ΔE. The effects of tube distension on system energy are more pronounced at lower backpressure and chamber-pressure, and lower pulse frequency/higher stroke volume conditions. The Ecoflex aorta had insufficient data points for a full analysis due to a high number of discarded cases.

High aorta length and low chamber pressure are aortic parameters which lead to low aortic distension. A low distension results in both a reduced pump workload and a greater decrease in pump energy downstream of the aorta compliance section. These positive effects are greater when pump operating conditions are at the lower end of the pulse frequency, and correspondingly the higher end of stroke volume. Lowering the backpressure conditions acting on the pump result in an increase of positive effects of compliance parameters. Overall, a low system backpressure with low aorta distension due to low chamber-pressure and long aorta length will have a positive impact on pump performance. The worst conditions for pump performance would be a high backpressure with high aorta distension due to high chamber-pressure and low aorta length. This applies for all pulse frequency/stroke volume conditions.

5.2. Limitations and Future Work

This investigation created the foundations of the modular experimental EVHP flow loop and allowed testing of several parameters. Improvements can be made from this point forward. The modular design of the experimental system allows for the ease of modifying or adding major components. Table 5.1 lists the limitations and potential solutions and suggestions for future work. Many limitations can be solved with one proposed solution.

Table 5.1: Limitations and future work

Limitation	**Potential solution/future work**
System vibrations	Develop new iteration of pulsatile pump with smaller size and less impactful shaft in-stroke and out-stroke, with pressure waveform closer to human heart ranges, pulse frequency and stroke volume can be separately controlled.
Pulsatile pump pressure waveform not an exact match to human heart pressure waveform	
Experimental pressure outside pressure transducer measurement range	
Pulse frequency and stroke volume cannot be separately controlled	
Discarding experimental cases due to tube wall-contact and collapse	Develop larger pressure chamber to facilitate higher aorta distension diameters or develop aorta with self-retracting properties to remove need for pressure-chamber.
Ecoflex aorta lacking sufficient data points	
Chamber-pressure fluctuation	
Newtonian fluid (water) as system fluid	Test pulsatile system using blood analog fluid for non-Newtonian effects.
Unknown profile of velocity	Conduct PIV at aorta inlet and aorta outlet.

References

[1] S. L. Longnus, V. Mathys, M. Dornbierer, F. Dick, T. P. Carrel, and H. T. Tevaearai, “Heart transplantation with donation after circulatory determination of death,” *Nat. Rev. Cardiol.*, vol. 11, no. 6, pp. 354–363, 2014, doi: 10.1038/nrcardio.2014.45.

[2] M. Guglin, “How to increase the utilization of donor hearts ?,” no. May 2014, pp. 95–105, 2015, doi: 10.1007/s10741-014-9434-y.

[3] J. E. E. Tuttle-Newhall, S. M. M. Krishnan, M. F. F. Levy, V. McBride, J. P. P. Orlowski, and R. S. S. Sung, “Organ donation and utilization in the United States: 1998-2007,” *Am. J. Transplant.*, vol. 9, no. 4 PART 2, pp. 879–893, 2009, doi: 10.1111/j.1600-6143.2009.02565.x.

[4] K. Hornby, H. Ross, S. Keshavjee, V. Rao, and S. D. Shemie, “Non-utilization of hearts and lungs after consent for donation: a Canadian multicentre study.,” *Can. J. Anaesth.*, vol. 53, no. 8, pp. 831–7, 2006, doi: 10.1007/BF03022801.

[5] S. M. Minasian, M. M. Galagudza, Y. V. Dmitriev, A. A. Karpov, and T. D. Vlasov, “Preservation of the donor heart: from basic science to clinical studies,” *Interact. Cardiovasc. Thorac. Surg.*, vol. 20, no. 4, pp. 510–519, 2015, doi: 10.1093/icvts/ivu432.

[6] S. J. Messer, R. G. Axell, S. Colah, P. A. White, M. Ryan, A. A. Page, B. Parizkova, K. Valchanov, W. White, D. H. Freed, E. Ashley, M. Goddard, J. Parameshwar, C. J. Watson, T. Krieg, A. Ali, S. Tsui, S. R. Large, M. Ryan, A. A. Page, B. Parizkova, K. Valchanov, C. W. White, D. H. Freed, E. Ashley, J. Dunning, J. Parameshwar, C. J. Watson, T. Krieg, and A. Ali, “Functional Assessment and Transplantation of the Donor Heart Following

Circulatory Death," *J. Hear. Lung Transplant.*, 2016, doi: 10.1016/j.healun.2016.07.004.

[7] C. W. C. W. White, E. Ambrose, A. Müller, Y. Li, H. Le, B. Hiebert, R. Arora, T. W. T. W. Lee, I. Dixon, G. Tian, J. Nagendran, L. Hryshko, and D. D. H. Freed, "Assessment of donor heart viability during ex vivo heart perfusion," *Can. J. Physiol. Pharmacol.*, vol. 901, no. May, pp. 893–901, 2015, doi: 10.1139/cjpp-2014-0474.

[8] J. F. McAnulty, "Hypothermic organ preservation by static storage methods: Current status and a view to the future," *Cryobiology*, vol. 60, no. 3 SUPPL., pp. S13–S19, 2010, doi: 10.1016/j.cryobiol.2009.06.004.

[9] P. Biberthaler, B. Luchting, S. Massberg, D. Teupser, S. Langer, R. Leiderer, K. Messmer, and F. Krombach, "The influence of organ temperature on hepatic ischemia-reperfusion injury: A systematic analysis," *Transplantation*, vol. 72, no. 9, pp. 1486–1490, 2001, doi: 10.1097/00007890-200111150-00003.

[10] A. R. Manara, P. G. Murphy, and G. Ocallaghan, "Donation after circulatory death," *British Journal of Anaesthesia*, vol. 108, no. SUPPL. 1. 2012, doi: 10.1093/bja/aer357.

[11] C. Moers, H. G. D. Leuvenink, and R. J. Ploeg, "Donation after cardiac death: Evaluation of revisiting an important donor source," *Nephrol. Dial. Transplant.*, vol. 25, no. 3, pp. 666–673, 2010, doi: 10.1093/ndt/gfp717.

[12] C. W. White, A. Ali, D. Hasanally, B. Xiang, Y. Li, P. Mundt, M. Lytwyn, S. Colah, J. Klein, A. Ravandi, R. C. Arora, T. W. Lee, L. Hryshko, S. Large, G. Tian, and D. H. Freed, "A cardioprotective preservation strategy employing ex vivo heart perfusion facilitates successful transplant of donor hearts after cardiocirculatory death," *J. Hear. Lung Transplant.*, vol. 32, no. 7, pp. 734–743, 2013, doi: 10.1016/j.healun.2013.04.016.

[13] C. W. White, D. Hasanally, P. Mundt, Y. Li, B. Xiang, J. Klein, A. Müller, E. Ambrose, A. Ravandi, R. C. Arora, T. W. Lee, L. V. Hryshko, S. Large, G. Tian, and D. H. Freed, "A whole blood-based perfusate provides superior preservation of myocardial function during ex vivo heart perfusion," *J. Hear. Lung Transplant.*, vol. 34, no. 1, pp. 113–121, 2015, doi: 10.1016/j.healun.2014.09.021.

[14] G. G. G. Belz, "Elastic Properties and Windkessel Function of the Human Aorta," *Cardiovasc. Drugs Ther.*, vol. 9, no. 1, pp. 73–83, 1995, doi: 10.1007/BF00877747.

[15] K. G. Cameron, "Investigation into the effect of compliant response on fluid flow in an ex vivo heart perfusion test system," 2016.

[16] M. Büsen, C. Arenz, M. Neidlin, S. Liao, T. Schmitz-Rode, U. Steinseifer, and S. J. Sonntag, "Development of an In Vitro PIV Setup for Preliminary Investigation of the Effects of Aortic Compliance on Flow Patterns and Hemodynamics," *Cardiovasc. Eng. Technol.*, vol. 8, no. 3, pp. 368–377, Sep. 2017, doi: 10.1007/s13239-017-0309-y.

[17] P. H. Geoghegan, N. A. Buchmann, C. J. T. Spence, S. Moore, and M. Jermy, "Fabrication of rigid and flexible refractive-index-matched flow phantoms for flow visualisation and optical flow measurements," *Exp. Fluids*, vol. 52, no. 5, pp. 1331–1347, 2012, doi: 10.1007/s00348-011-1258-0.

[18] M. Stoiber, T. Schlöglhofer, P. Aigner, C. Grasl, and H. Schima, "An alternative method to create highly transparent hollow models for flow visualization," *Int. J. Artif. Organs*, vol. 36, no. 2, pp. 131–134, 2013, doi: 10.5301/ijao.5000171.

[19] U. Gülan, B. Lüthi, M. Holzner, A. Liberzon, A. Tsinober, and W. Kinzelbach, "Experimental study of aortic flow in the ascending aorta via Particle Tracking

Velocimetry," *Exp. Fluids*, vol. 53, no. 5, pp. 1469–1485, 2012, doi: 10.1007/s00348-012-1371-8.

[20] R. Yip, R. Mongrain, A. Ranga, J. Brunette, and R. Cartier, "Development of Anatomically Correct Mock-Ups of the Aorta for Piv Investigations," *Proc. Can. Eng. Educ. Assoc.*, no. 1, pp. 1–10, 2011, doi: 10.24908/pceea.v0i0.3984.

[21] K. Pielhop, C. Schmidt, S. Zholtovski, M. Klaas, and W. Schröder, "Experimental investigation of the flow field in an elastic 180 ° curved vessel," no. 1997, p. 2014, 2014.

[22] S. Burgmann, S. Groe, W. Schröder, J. Roggenkamp, S. Jansen, F. Gräf, and M. Büsen, "A refractive index-matched facility for fluid-structure interaction studies of pulsatile and oscillating flow in elastic vessels of adjustable compliance," *Exp. Fluids*, vol. 47, no. 4–5, pp. 865–881, 2009, doi: 10.1007/s00348-009-0681-y.

[23] A. Benbrahim, G. J. L'Italien, B. B. Milinazzo, D. F. Warnock, S. Dhara, J. P. Gertler, R. W. Orkin, and W. M. Abbott, "A compliant tubular device to study the influences of wall strain and fluid shear stress on cells of the vascular wall," *J. Vasc. Surg.*, vol. 20, no. 2, pp. 184–94, 1994, doi: 10.1016/0741-5214(94)90005-1.

[24] R. M. Berne, B. M. Koeppen, and B. A. Stanton, *Berne & Levy Physiology*. Philadelphia: Mosby/Elsevier, 2003.

[25] W. W. Nichols, "Clinical measurement of arterial stiffness obtained from noninvasive pressure waveforms," *Am. J. Hypertens.*, vol. 18, no. 1 SUPPL., pp. 3–10, 2005, doi: 10.1016/j.amjhyper.2004.10.009.

[26] T. Weber, J. Auer, M. F. O'Rourke, C. Punzengruber, E. Kvas, and B. Eber, "Prolonged mechanical systole and increased arterial wave reflections in diastolic dysfunction," *Heart*,

vol. 92, no. 11, pp. 1616–1622, 2006, doi: 10.1136/hrt.2005.084145.

[27] J. P. Murgo, N. Westerhof, J. P. Giolma, and S. a Altobelli, “Aortic input impedance in normal man: relationship to pressure wave forms.,” *Circulation*, vol. 62, no. 1, pp. 105–116, 1980, doi: 10.1161/01.CIR.62.1.105.

[28] S. Emrani, T. S. Saponas, D. Morris, and H. Krim, “A Novel Framework for Pulse Pressure Wave Analysis Using Persistent Homology,” *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1879–1883, 2015, doi: 10.1109/LSP.2015.2441068.

[29] J. R. R. Womersley, “Method for the calculation of velocity, rate of flow and viscous drag in arteries when the pressure gradient is known,” *J. Physiol.*, vol. 127, no. 3, pp. 553–563, 1955, doi: 10.1113/jphysiol.1955.sp005276.

[30] Y. A. Cengel and J. M. Cimbala, *Fluid Mechanics Fundamentals and Applications, Second Edition*, 2nd ed. New York: McGraw-Hill, 2010.

[31] N. B. Wood, “Aspects of fluid dynamics applied to the larger arteries,” *J. Theor. Biol.*, vol. 199, no. 2, pp. 137–161, 1999, doi: 10.1006/jtbi.1999.0953.

[32] N. M. Pahlevan and M. Gharib, “In-vitro investigation of a potential wave pumping effect in human aorta,” *J. Biomech.*, vol. 46, no. 13, pp. 2122–2129, 2013, doi: 10.1016/j.jbiomech.2013.07.006.

[33] B. B. Lieber, V. Livescu, L. N. Hopkins, and A. K. Wakhloo, “Particle image velocimetry assessment of stent design influence on intra-aneurysmal flow,” *Ann. Biomed. Eng.*, vol. 30, no. 6, pp. 768–777, 2002, doi: 10.1114/1.1495867.

[34] J. P. Ku, C. J. Elkins, and C. A. Taylor, “Comparison of CFD and MRI flow and velocities

in an in vitro large artery bypass graft model," *Ann. Biomed. Eng.*, vol. 33, no. 3, pp. 257–269, 2005, doi: 10.1007/s10439-005-1729-7.

[35] P. R. Hoskins, "Simulation and Validation of Arterial Ultrasound Imaging and Blood Flow," *Ultrasound Med. Biol.*, vol. 34, no. 5, pp. 693–717, 2008, doi: 10.1016/j.ultrasmedbio.2007.10.017.

[36] M. H. Babiker, L. F. Gonzalez, J. Ryan, F. Albuquerque, D. Collins, A. Elvikis, and D. H. Frakes, "Influence of stent configuration on cerebral aneurysm fluid dynamics," *J. Biomech.*, vol. 45, no. 3, pp. 440–447, 2012, doi: 10.1016/j.jbiomech.2011.12.016.

[37] V. Labs, "SuperPump System User Manual," pp. 1–26.

[38] H. Studies, "Series 1400 Pulsatile Blood Pumps User ' s Manual," 2004.

[39] C. E. Taylor, Z. W. Dziczkowski, and G. E. Miller, "Automation of the Harvard Apparatus Pulsatile Blood Pump," *J. Med. Devices, Trans. ASME*, vol. 6, no. 4, pp. 1–10, 2012, doi: 10.1115/1.4007637.

[40] Shelly Medical Imaging Technologies, "Cardioflow 5000 Mr Computer-Controlled Flow Pump System," pp. 0–1, 2015.

[41] Shelly Medical Imaging Technologies, "CompuFlow 1000 Physiological Flow Pump System." .

[42] R. A. Chaudhury, V. Atlasman, G. Pathangey, N. Pracht, R. J. Adrian, and D. H. Frakes, "A High Performance Pulsatile Pump for Aortic Flow Experiments in 3-Dimensional Models," *Cardiovasc. Eng. Technol.*, vol. 7, no. 2, pp. 148–158, 2016, doi: 10.1007/s13239-016-0260-3.

[43] E. Asmar, G. Bejjani, R. Chamoun, J. Hachem, G. Oweis, and M. Liermann, “Experimental study on active pneumatic damping of pulsatile flow delivered from peristaltic pump,” *ASME/BATH 2017 Symp. Fluid Power Motion Control. FPMC 2017*, pp. 1–8, 2017, doi: 10.1115/FPMC2017-4330.

[44] M. Liermann, “Active pneumatic pulsation damper for peristaltic pump flow loops,” *BATH/ASME 2016 Symp. Fluid Power Motion Control. FPMC 2016*, pp. 1–9, 2016, doi: 10.1115/FPMC2016-1707.

[45] S. Yilmaz, O. Toker, N. Arslan, and H. Sedef, “Optimal in vitro realization of pulsatile coronary artery flow waveforms using closed-loop feedback algorithms with multiple flow control devices,” *Turkish J. Electr. Eng. Comput. Sci.*, vol. 20, no. 6, pp. 1006–1030, 2012, doi: 10.3906/elk-1101-1024.

[46] W. Tsai and Ö. Savaş, “Flow pumping system for physiological waveforms,” *Medical and Biological Engineering and Computing*, vol. 48, no. 2. pp. 197–201, 2010, doi: 10.1007/s11517-009-0573-6.

[47] A. Eriksson, H. W. Persson, and K. Lindström, “A computer-controlled arbitrary flow wave form generator for physiological studies,” *Rev. Sci. Instrum.*, vol. 71, no. 1, pp. 235–242, 2000, doi: 10.1063/1.1150189.

[48] F. Gräf, T. Finocchiaro, M. Laumen, I. Mager, and U. Steinseifer, “Mock Circulation Loop to Investigate Hemolysis in a Pulsatile Total Artificial Heart,” *Artif. Organs*, vol. 39, no. 5, pp. 416–422, 2015, doi: 10.1111/aor.12399.

[49] J. R. Crosby, K. J. Decook, P. L. Tran, R. G. Smith, D. F. Larson, Z. I. Khalpey, D. Burkhoff, and M. J. Slepian, “Physiological characterization of the SynCardia total artificial heart in a

mock circulation system," *ASAIO J.*, vol. 61, no. 3, pp. 274–281, 2015, doi: 10.1097/MAT.0000000000000192.

[50] F. M. Donovan, "Design of a Hydraulic Analog of the Circulatory System for Evaluating Artificial Hearts," *Biomater. Med. Devices. Artif. Organs*, vol. 3, no. 4, pp. 439–449, 1975, doi: 10.3109/10731197509118635.

[51] O. Dur, M. Yoshida, P. Manor, A. Mayfield, P. D. Wearden, V. O. Morell, and K. Pekkan, "In vitro evaluation of right ventricular outflow tract reconstruction with bicuspid valved polytetrafluoroethylene conduit," *Artif. Organs*, vol. 34, no. 11, pp. 1010–1016, 2010, doi: 10.1111/j.1525-1594.2010.01136.x.

[52] D. J. Farrar and J. D. Hill, "Univentricular and biventricular thoratec VAD support as a bridge to transplantation," *Ann. Thorac. Surg.*, vol. 55, no. 1, pp. 276–282, 1993, doi: 10.1016/0003-4975(93)90537-R.

[53] D. J. Farrar, P. G. Compton, J. H. Lawson, J. J. Hershon, and J. D. Hill, "Control Modes of a Clinical Ventricular Assist Device," *IEEE Eng Med Biol*, vol. 5(1), no. March, pp. 19–25, 1986, doi: 10.1109/MEMB.1986.5006251.

[54] Swanson and Clark, "Dimensions and Geometric Relationships of the Human Aortic Value as a Function of Pressure," pp. 871–883, 1974.

[55] A. P. Yoganathan, Z. He, and S. Casey Jones, "Fluid Mechanics of Heart Valves," *Annu. Rev. Biomed. Eng.*, vol. 6, no. August, pp. 331–362, 2004, doi: 10.1146/annurev.bioeng.6.040803.140111.

[56] K. B. Chandran, G. N. Cabell, B. Khalighi, and C. J. Chen, "Laser anemometry measurements of pulsatile flow past aortic valve prostheses," *J. Biomech.*, vol. 16, no. 10,

pp. 865–873, 1983, doi: 10.1016/0021-9290(83)90011-8.

[57] L. P. Dasi, H. A. Simon, P. Sucosky, and A. P. Yoganathan, “Fluid mechanics of artificial heart valves,” *Clin. Exp. Pharmacol. Physiol.*, vol. 36, no. 2, 2009, doi: 10.1111/j.1440-1681.2008.05099.x.

[58] S. Johnson, M. R. Stroud, J. M. Kratz, S. M. Bradley, F. A. Crawford, and J. S. Ikonomidis, “Thirty-year experience with a bileaflet mechanical valve prosthesis,” *J. Thorac. Cardiovasc. Surg.*, vol. 157, no. 1, pp. 213–222, 2019, doi: 10.1016/j.jtcvs.2018.09.002.

[59] J. C. Bokros, V. L. Gott, L. D. La Grange, A. M. Fadall, K. D. Vos, and M. D. Ramos, “Correlations between blood compatibility and heparin adsorptivity for an impermeable isotropic pyrolytic carbon,” *J. Biomed. Mater. Res.*, vol. 3, no. 3, pp. 497–528, 1969, doi: 10.1002/jbm.820030311.

[60] F. J. H. Gijsen, E. Allanic, F. N. Van De Vosse, and J. D. Janssen, “The influence of the non-Newtonian properties of blood on the flow in large arteries: steady flow in a carotid bifurcation model,” *J. Biomech.*, vol. 32, no. 7, pp. 705–713, 1999, doi: 10.1016/S0021-9290(99)00015-9.

[61] F. J. H. Gijsen, E. Allanic, F. N. van de Vosse, and J. D. Janssen, “The influence of the non-Newtonian properties of blood on the flow in large arteries: Unsteady flow in a 90°curved tube,” *J. Biomech.*, vol. 32, no. 7, pp. 705–713, 1999, doi: 10.1016/S0021-9290(99)00014-7.

[62] M. A. Elblbesy and A. T. Hereba, “Computation of the Coefficients of the Power law model for Whole Blood and Their Correlation with Blood Parameters,” *Appl. Phys. Res.*, vol. 8, no. 2, p. 1, 2016, doi: 10.5539/apr.v8n2p1.

[63] D. M. Eckmann, S. Bowers, M. Stecker, and a T. Cheung, “Hematocrit, volume expander, temperature, and shear rate effects on blood viscosity.,” *Anesth. Analg.*, vol. 91, no. 3, pp. 539–545, 2000, doi: 10.1213/00000539-200009000-00007.

[64] Y. I. Cho and K. R. Kensey, “Effects of the non-Newtonian viscosity of blood on flows in a diseased arterial vessel. Part 1: Steady flows,” *Biorheology*, vol. 28, no. 3–4, pp. 241–262, 1991, doi: 10.3233/BIR-1991-283-415.

[65] B. Liu, T. Dalin, and D. Tang, “Influence of non-Newtonian properties of blood on the wall shear stress in human atherosclerotic right coronary arteries,” *MCB Mol. Cell. Biomech.*, vol. 8, no. 1, pp. 73–90, 2011, doi: 10.3970/mcb.2011.008.073.

[66] ARO Fluild Management, “PD05P-AXS-XXX 1/2’’ DIAPHRAGM PUMP,” no. mm, pp. 1–2, 2000.

[67] Smooth-on, “Ecoflex® 00-50,” *Smooth-on*, 2017. https://www.smooth-on.com/products/ecoflex-00-50/.

[68] Smooth-On Inc., “Dragon Skin™ 10 Slow.” https://www.smooth-on.com/products/dragon-skin-10-medium/.

[69] M. J. Roman, R. B. Devereux, R. Kramer-Fox, and J. O’Loughlin, “Two-dimensional echocardiographic aortic root dimensions in normal children and adults,” *Am. J. Cardiol.*, vol. 64, no. 8, pp. 507–512, Sep. 1989, doi: 10.1016/0002-9149(89)90430-X.

[70] L. Campens, L. Demulier, K. De Groote, K. Vandekerckhove, D. De Wolf, M. J. Roman, R. B. Devereux, A. De Paepe, and J. De Backer, “Reference Values for Echocardiographic Assessment of the Diameter of the Aortic Root and Ascending Aorta Spanning All Age Categories,” *Am. J. Cardiol.*, vol. 114, no. 6, pp. 914–920, Sep. 2014, doi:

10.1016/J.AMJCARD.2014.06.024.

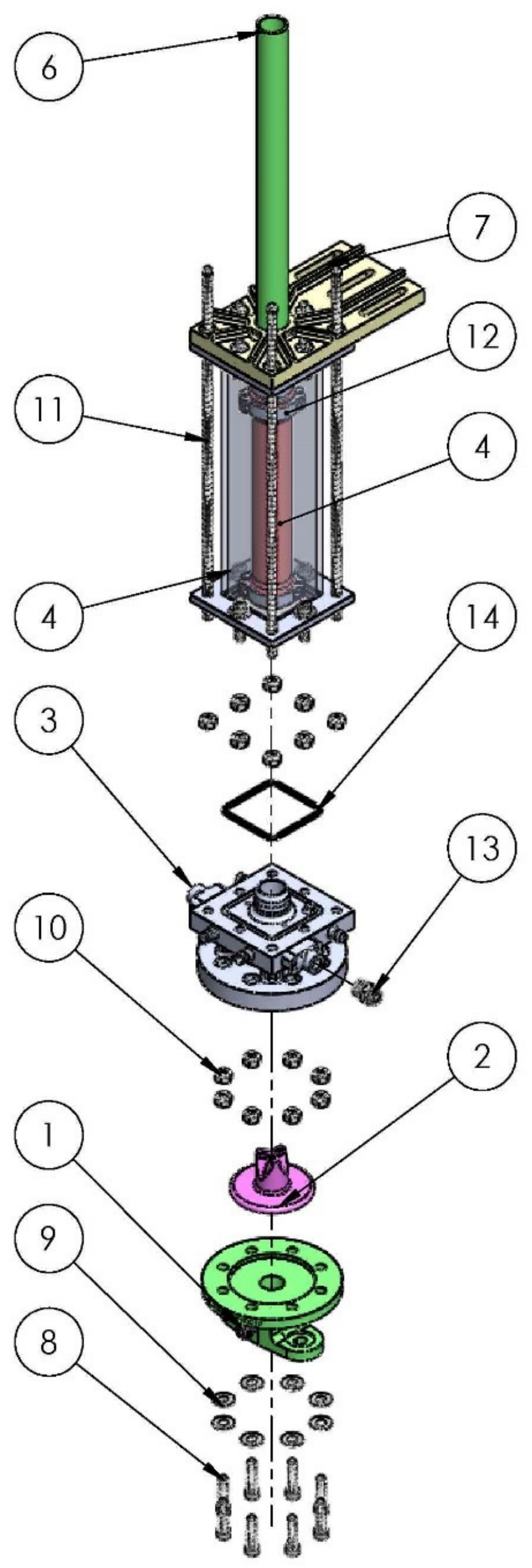
[71] I. Voges, M. Jerosch-Herold, J. Hedderich, E. Pardun, C. Hart, D. D. Gabbert, J. H. Hansen, C. Petko, H.-H. Kramer, and C. Rickers, "Normal values of aortic dimensions, distensibility, and pulse wave velocity in children and young adults: a cross-sectional study," *J. Cardiovasc. Magn. Reson.*, vol. 14, no. 1, p. 77, 2012, doi: 10.1186/1532-429X-14-77.

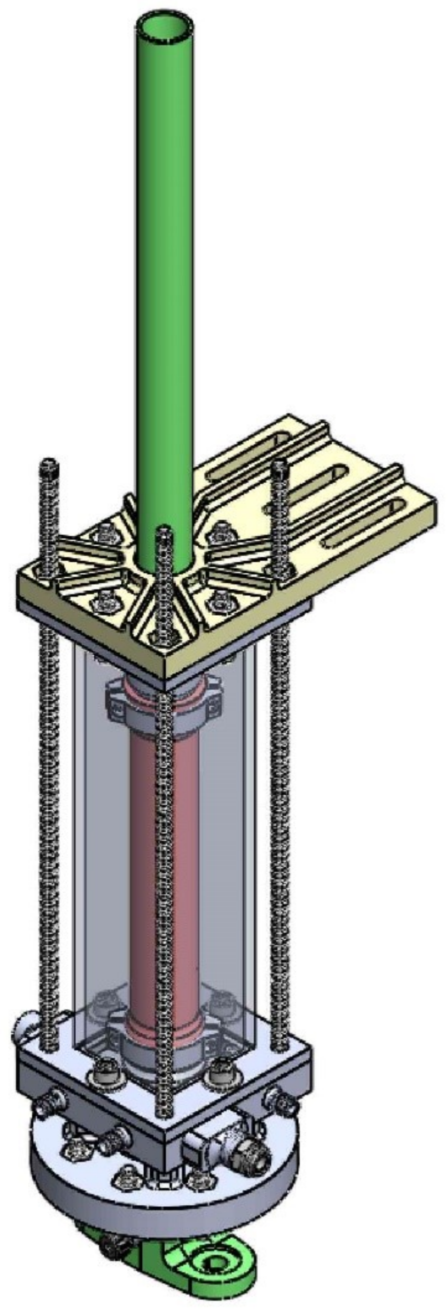
[72] Y. Sahasakul, W. D. Edwards, J. M. Naessens, and A. J. Tajik, "Age-related changes in aortic and mitral valve thickness: Implications for two-dimensional echocardiography based on an autopsy study of 200 normal human hearts," *Am. J. Cardiol.*, vol. 62, no. 7, pp. 424–430, 1988, doi: 10.1016/0002-9149(88)90971-X.

[73] S. Westaby, R. B. Karp, E. H. Blackstone, and S. P. Bishop, "Adult human valve dimensions and their surgical significance," *Am. J. Cardiol.*, vol. 53, no. 4, pp. 552–556, 1984, doi: 10.1016/0002-9149(84)90029-8.

Appendices

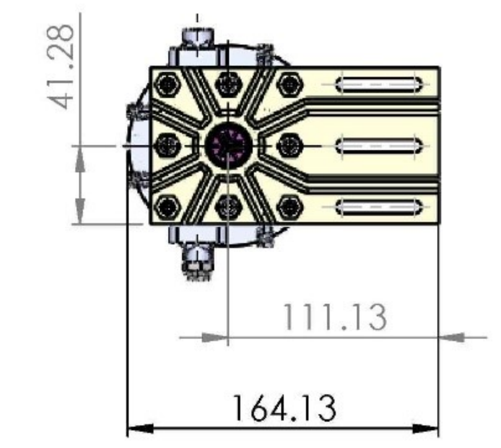
Appendix A. Experimental Setup Drawing Package

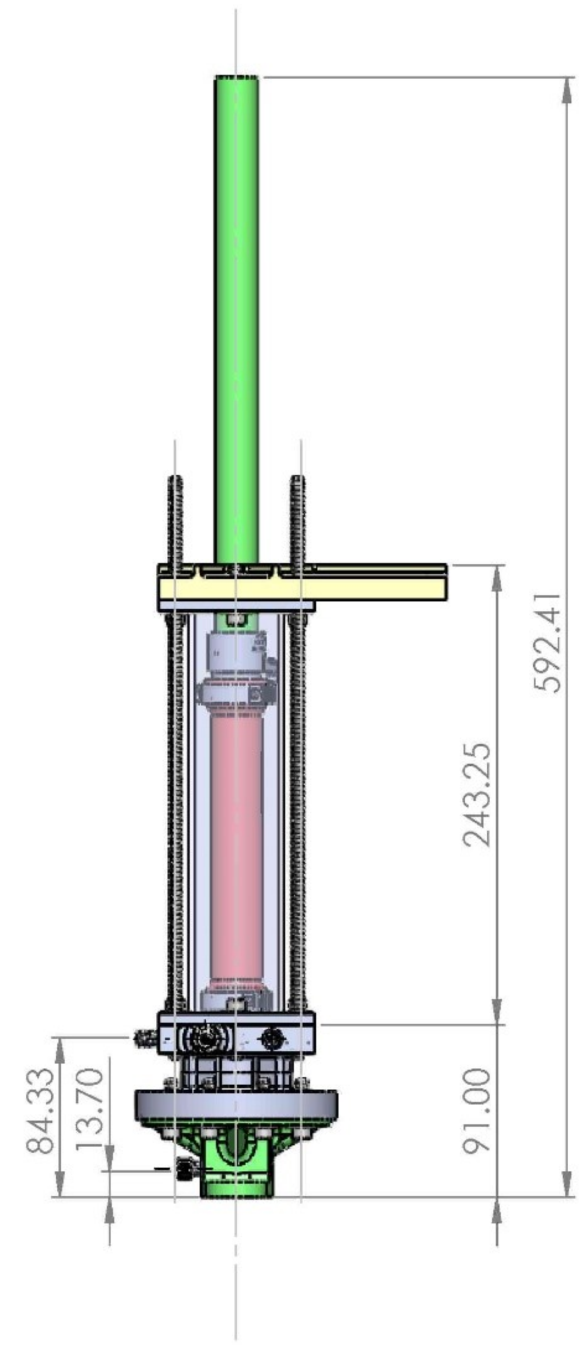


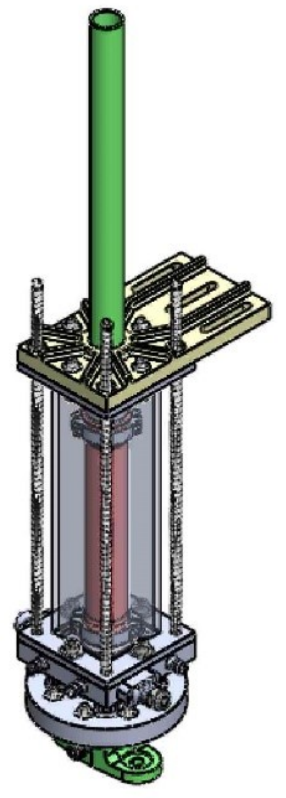


ITEM NO.	QTY.	Part Name
1	1	Pump Connection Plate
2	1	Trileaflet Valve
3	1	Valve Casing
4	1	Assembly - Acrylic Chamber
5	1	Silicone Aorta
6	1	Delring tube
7	1	Square Flange Cap
8	16	0.25" Socket Head Bolt Short 1" long
9	16	Washer for 0.25" Bolt
10	20	0.25" Hex Nut
11	4	Threaded Rod - 0.25"
12	4	C-Clamp
13	1	5779K109 Push-to-connect tube fitting for air
14	1	Valve Casing O-ring

Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH µm 0.6	Lab Day	TITLE: Pressure Chamber Assembly Isometric and Exploded View
	DO NOT SCALE DRAWING	SM By **Isobel Tetreau**	SIZE **B** / Assignment Number / REV
MATERIAL:		TA Initials / Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:5 / Mass: / SHEET 1 OF 17

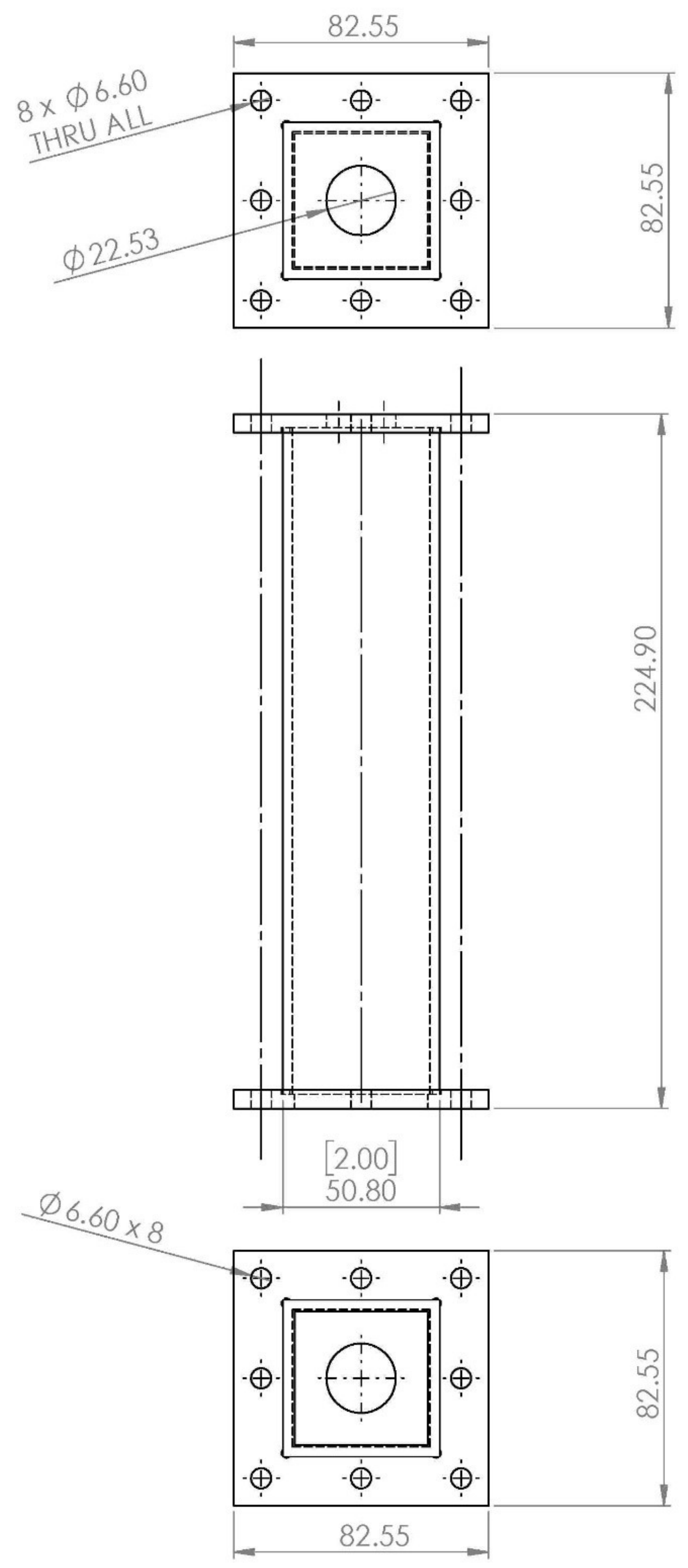


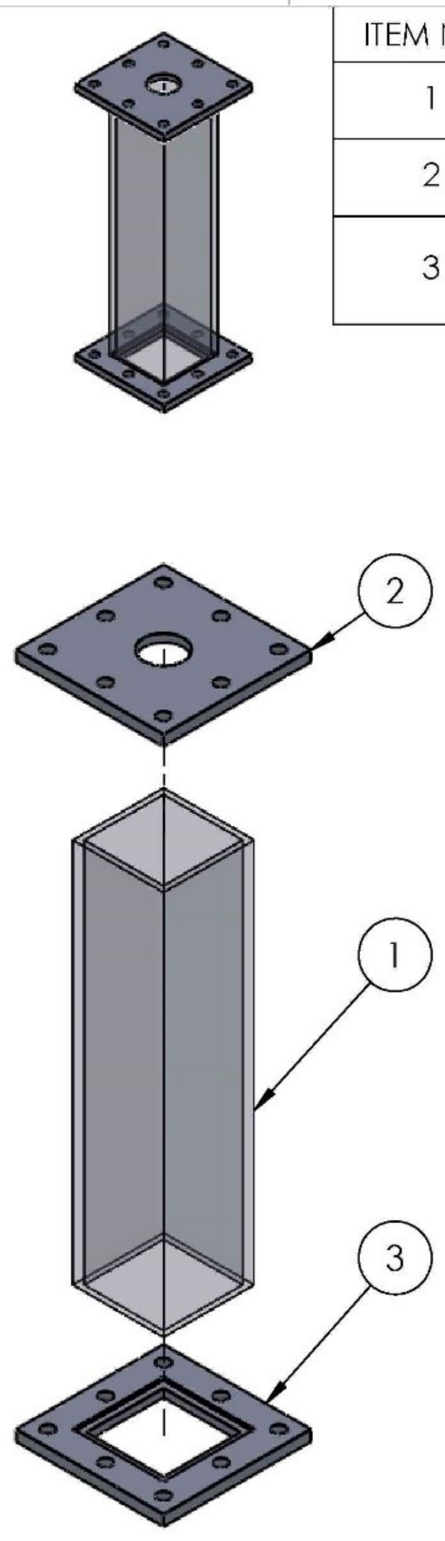




Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH µm 0.6	Lab Day	TITLE: Pressure Chamber Assembly Orthogonal View
	DO NOT SCALE DRAWING	SM By **Isobel Tetreau**	SIZE **B**
MATERIAL:		TA Initials	Assignment Number
		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	REV
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:5 Mass: SHEET 2 OF 17

SOLIDWORKS Educational Product. For Instructional Use Only.

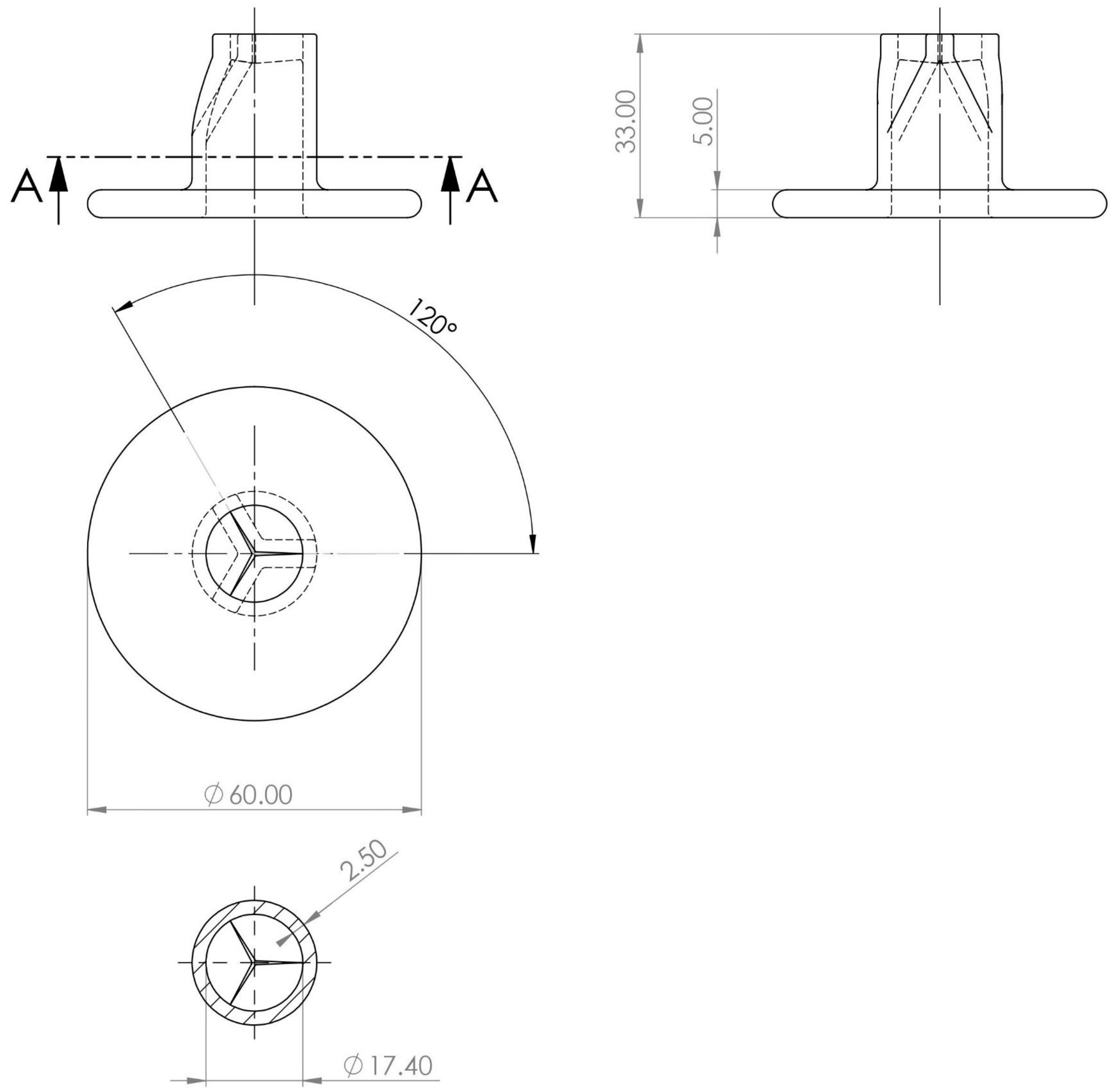




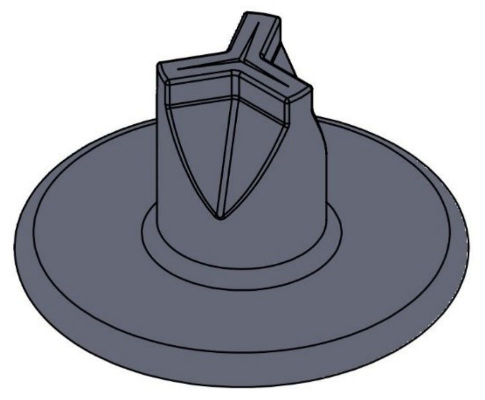
ITEM NO.	PART NUMBER	QTY.
1	Acrylic Chamber Rev N	1
2	Square Flange for PVC Pipe	1
3	Square Flange for Valve Connection Rev N Milled Corners	1

Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Acrylic Chamber
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** Assignment Number REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:2 Mass: SHEET 3 OF 17

SOLIDWORKS Educational Product. For Instructional Use Only.

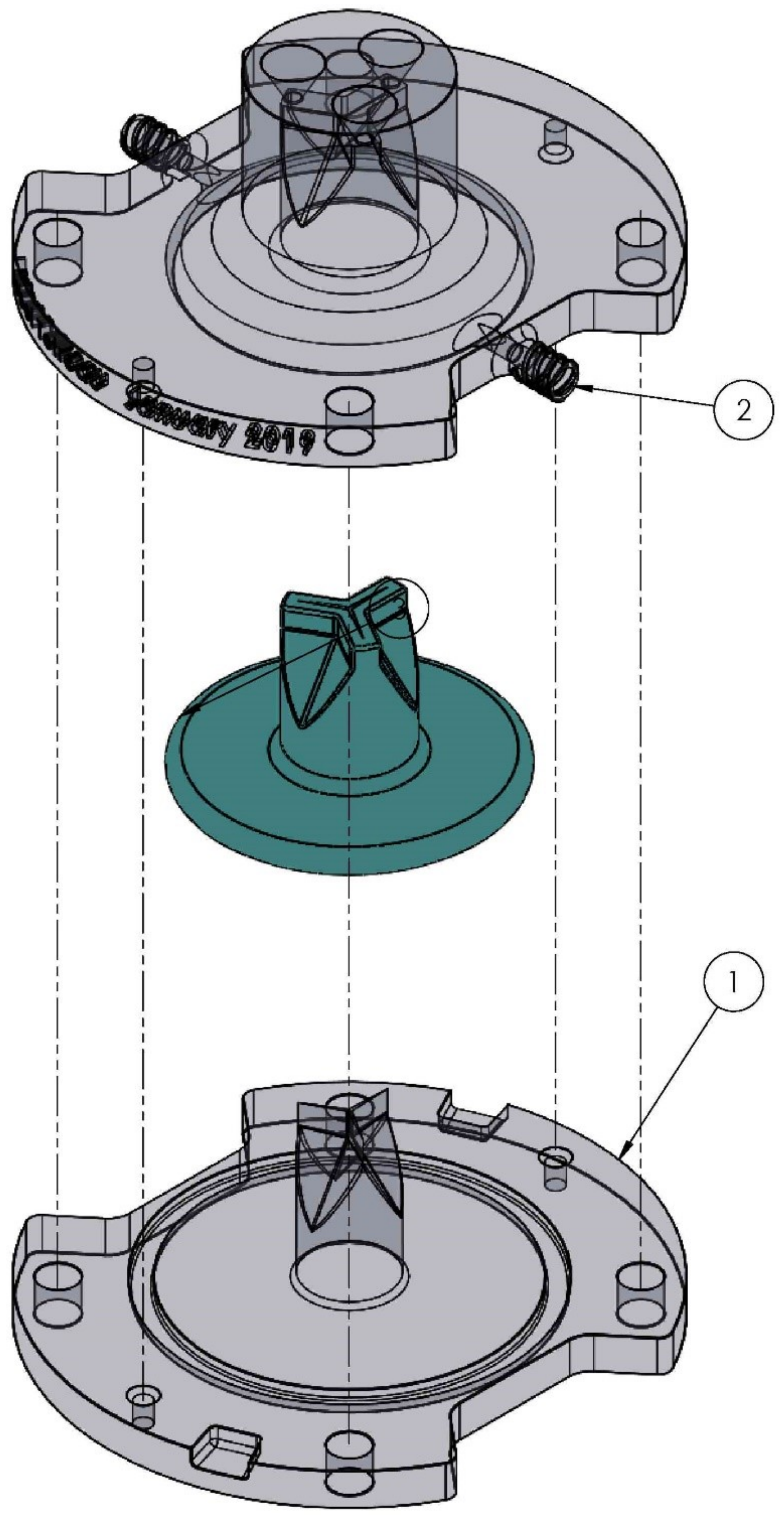


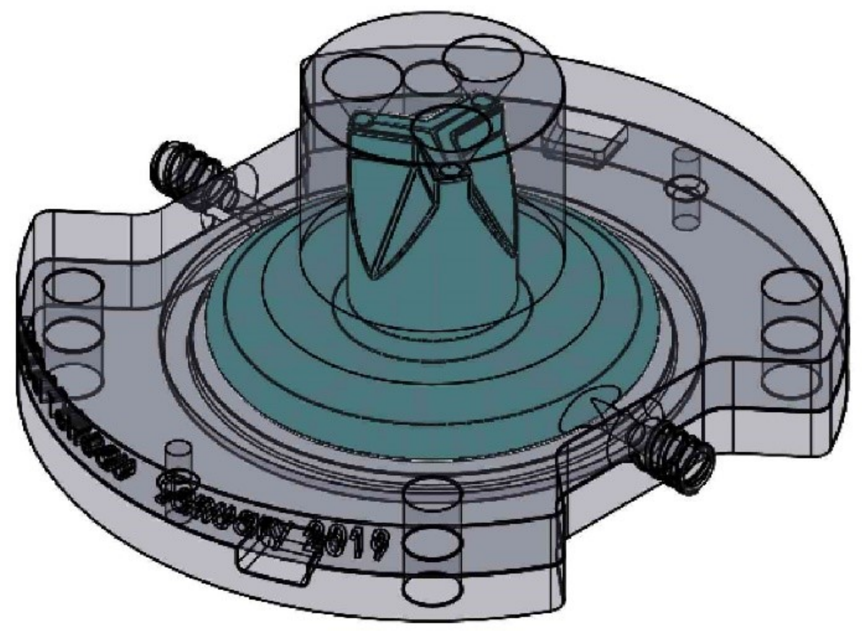
SECTION A-A
SCALE 1 : 1



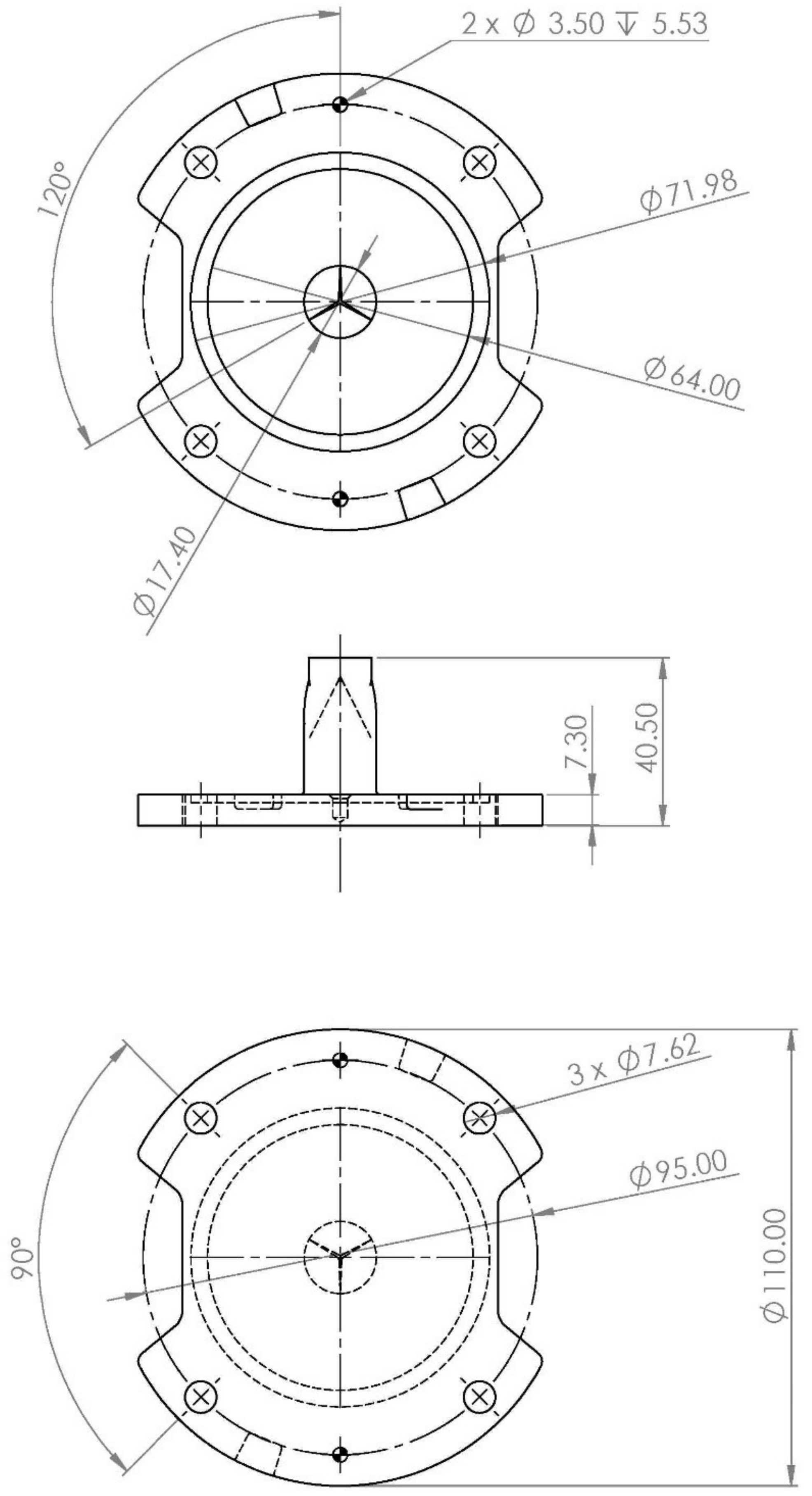
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Trileaflet Valve
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** — Assignment Number — REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:2 — Mass: — SHEET 4 OF 17

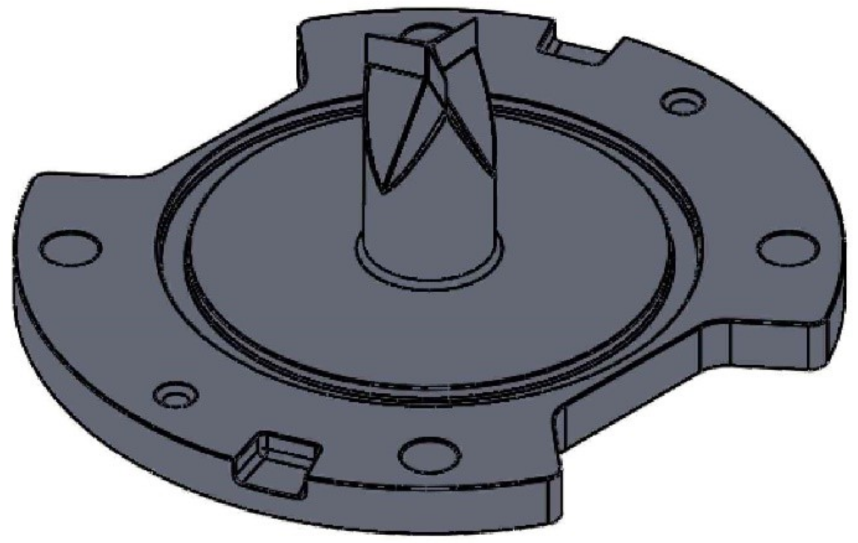
ITEM NO.	PART NAME	QTY.
1	Valve Positive Mold (Bottom)	1
2	Valve Negative Mold (Top)	1
3	Silicone Trileaflet Valve	1



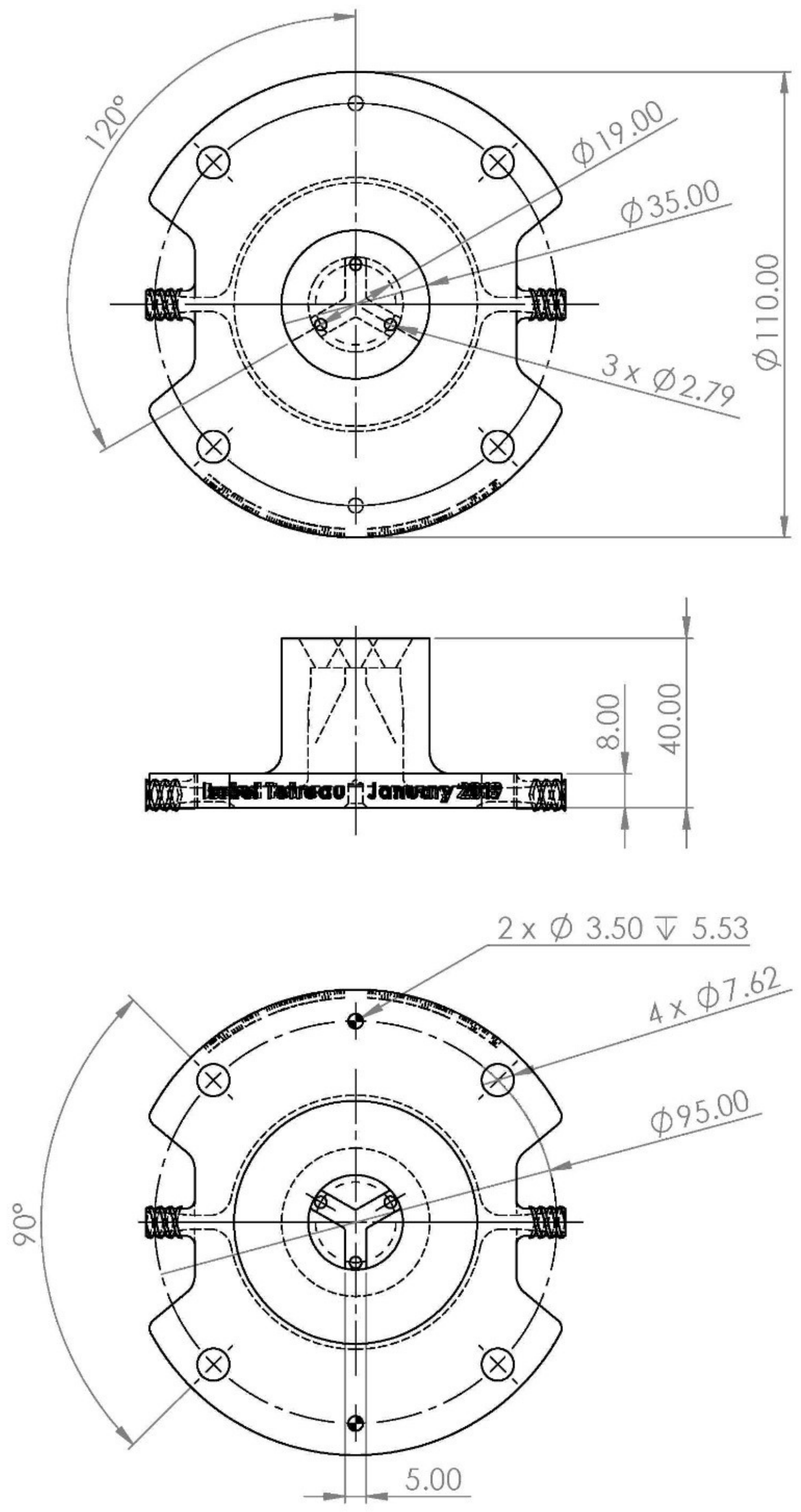


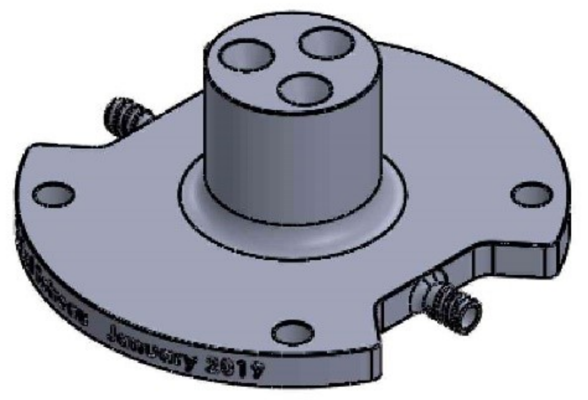
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Valve Mold Exploded
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** / Assignment Number / REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:2 / Mass: / SHEET 5 OF 17



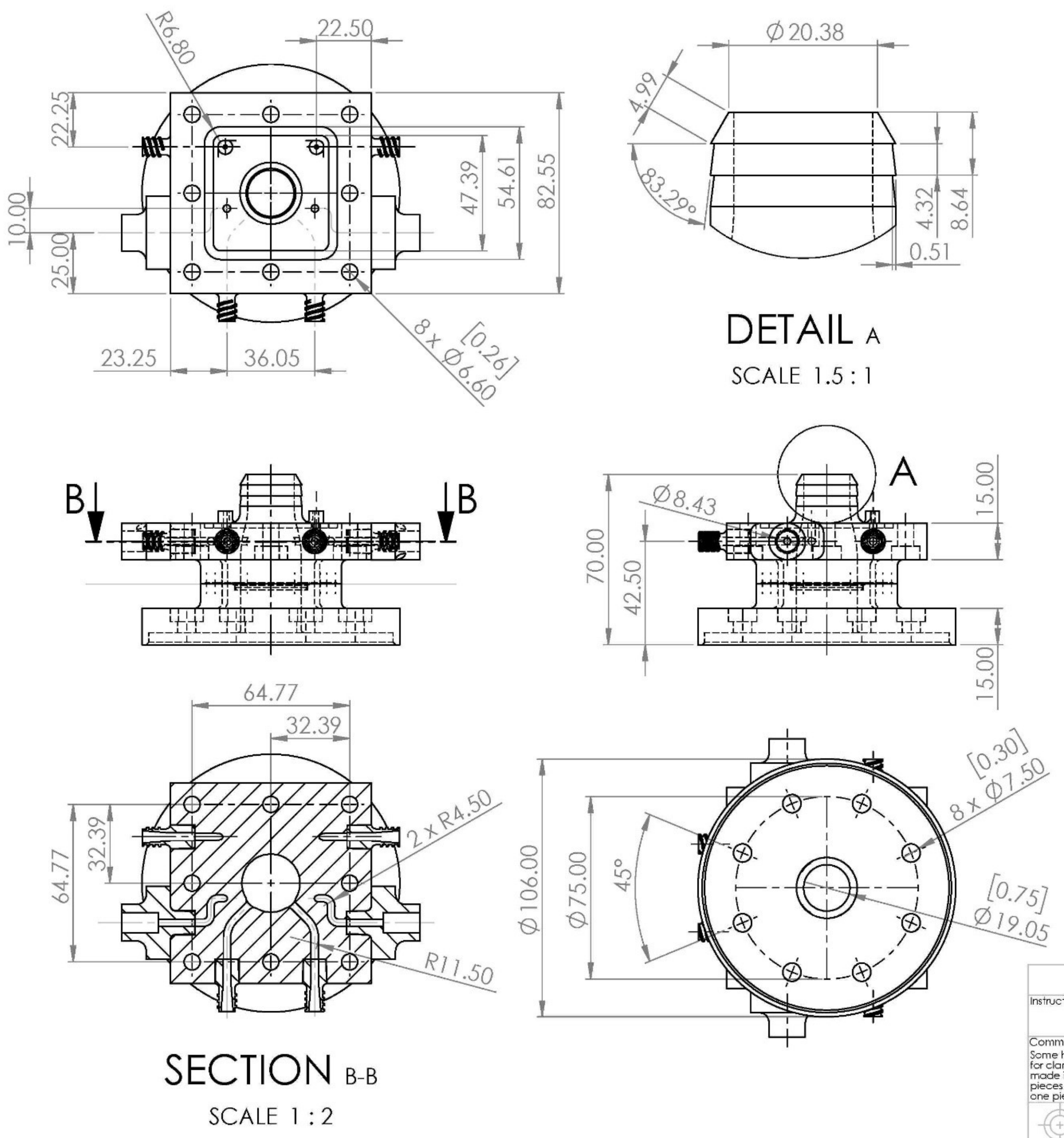


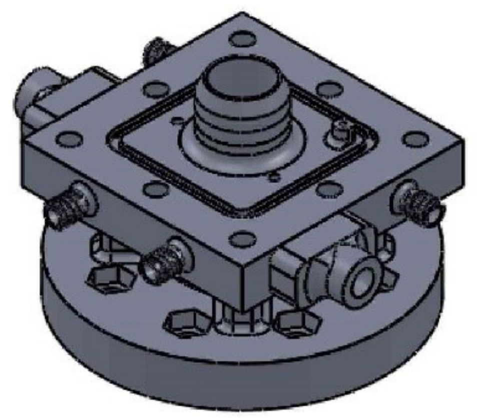
	UNLESS OTHERWISE SPECIFIED:	DRAWN BY:		The Department of Mechanical Engineering UNIVERSITY OF ALBERTA		
Instructor:	DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	**Sining Li**		TITLE: Valve Positive Mold (Bottom)		
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH µm 0.6	Lab Day				
		SM By	**Isobel Tetreau**			
	DO NOT SCALE DRAWING	TA Initials		SIZE **B**	Assignment Number	REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM				
FILE NAME: Aorta Pressure Chamber Assembly Rev P				SCALE: 1:1.5	Mass:	SHEET 6 OF 17





Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH 0.6 µm	Lab Day	TITLE: Valve Negative Mold (Top)
	DO NOT SCALE DRAWING	SM By **Isobel Tetreau**	
MATERIAL:		TA Initials	SIZE **B** / Assignment Number / REV
		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:1.5 / Mass: / SHEET 7 OF 17



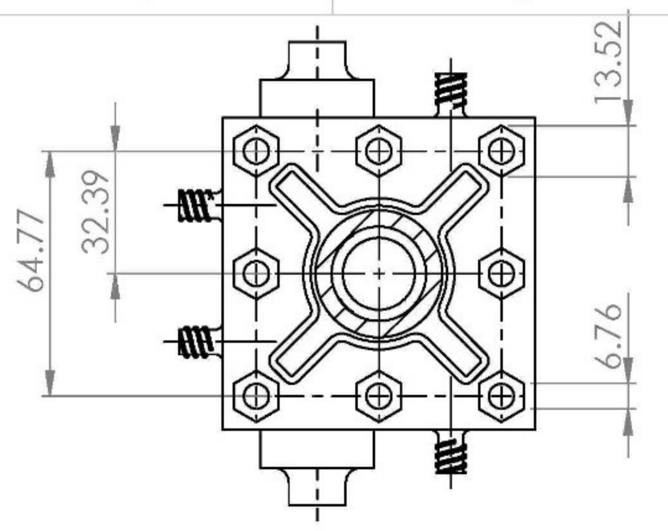


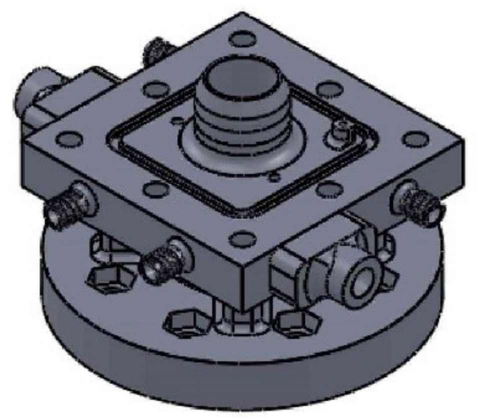
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Valve Casing 1/2
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B**
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	Assignment Number
FILE NAME: Aorta Pressure Chamber Assembly Rev P			REV
			SCALE:1:2
			Mass:
			SHEET 8 OF 17

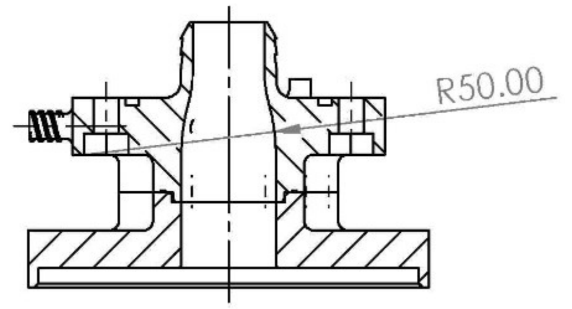
SOLIDWORKS Educational Product. For Instructional Use Only.

SECTION E-E

SCALE 1 : 2

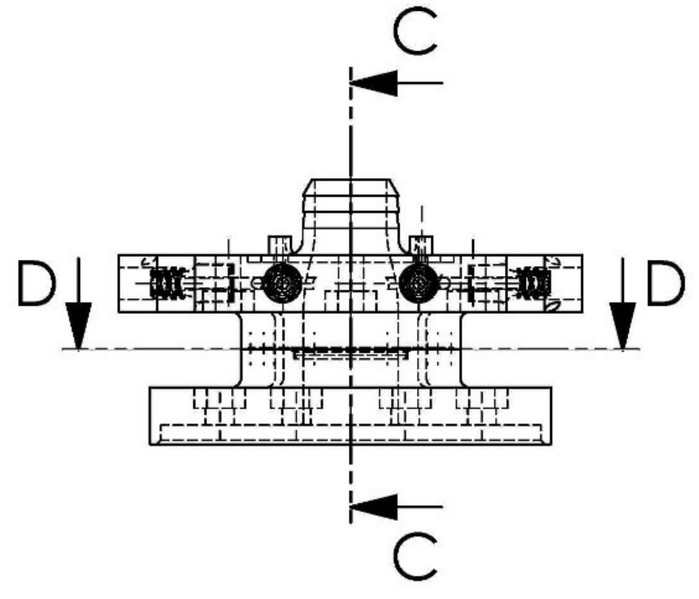


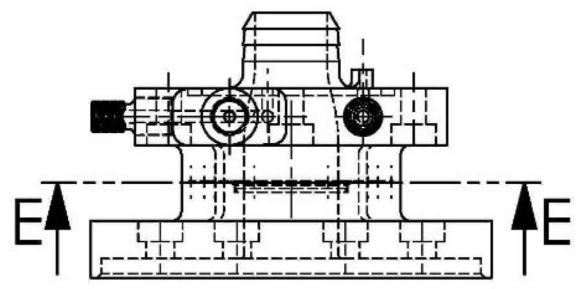


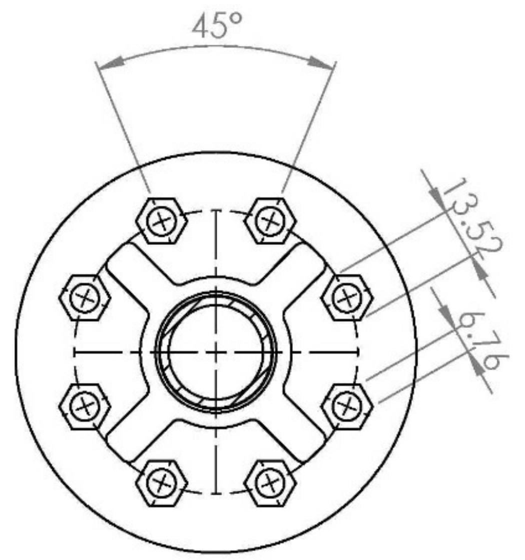


SECTION C-C

SCALE 1 : 2





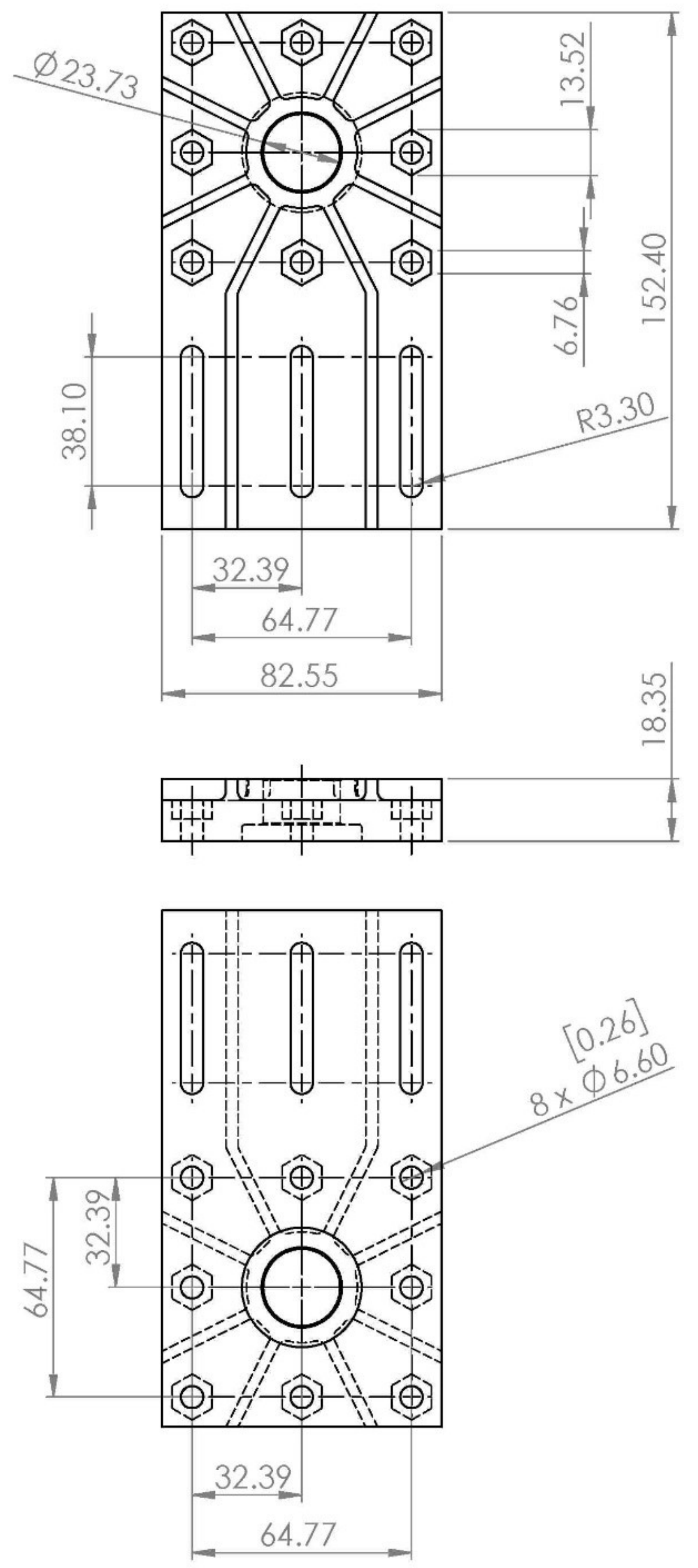


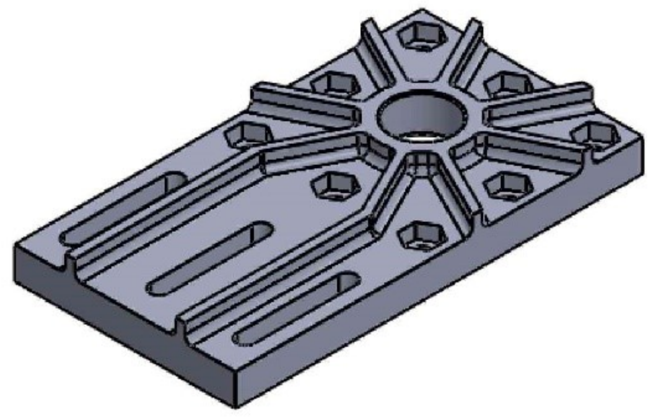
SECTION D-D

SCALE 1 : 2

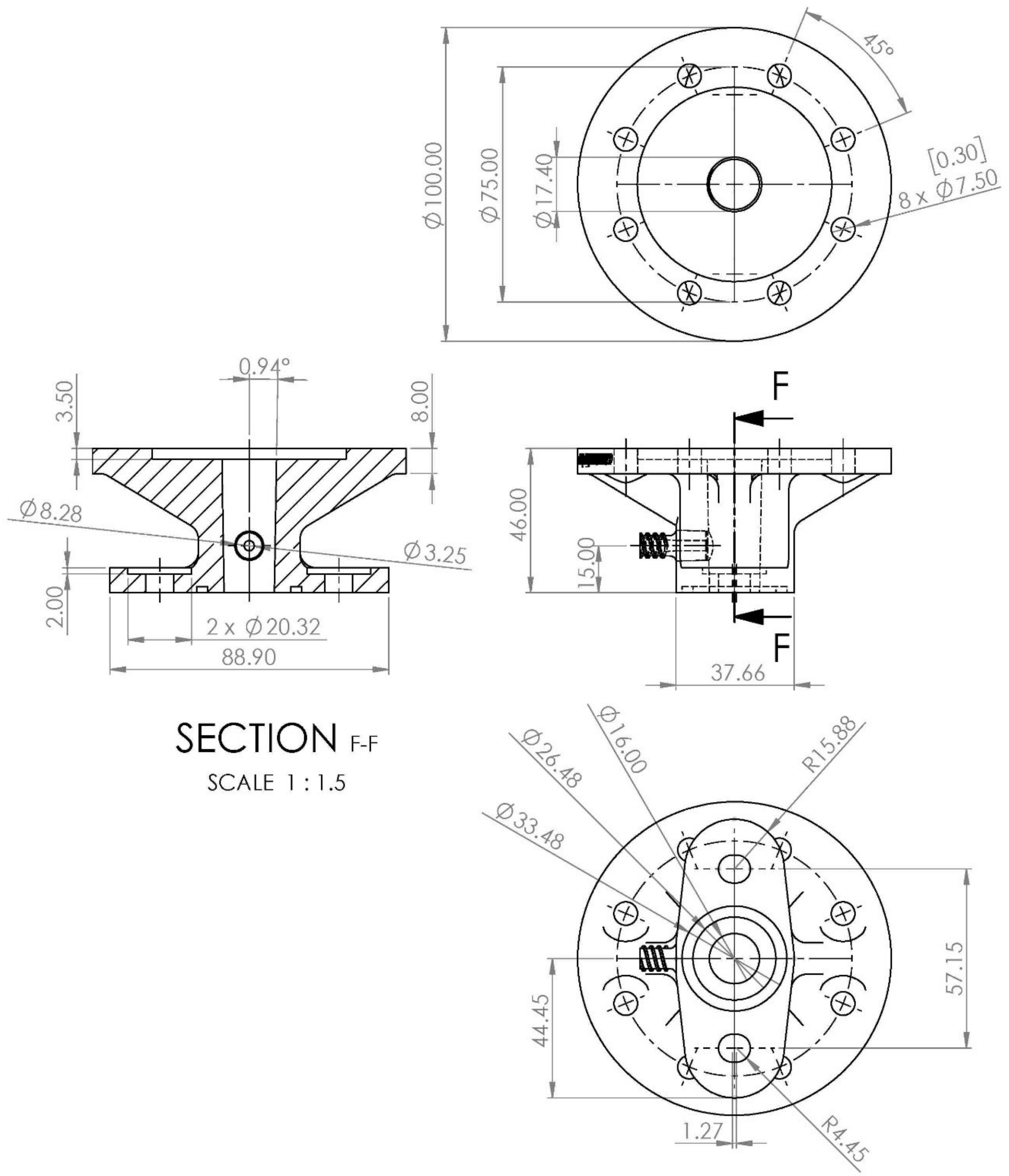
Note: Major dimensions on page 1/2. This page is for section views to show location of indented nut casings

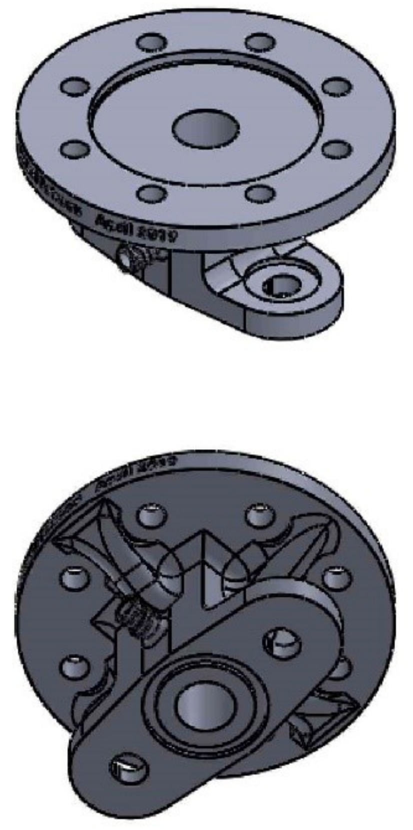
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH µm 0.6	Lab Day	TITLE: Valve Casing 2/2
	DO NOT SCALE DRAWING	SM By **Isobel Tetreau**	
MATERIAL:		TA Initials	SIZE **B** Assignment Number REV
		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE:1:2 Mass: SHEET 9 OF 17



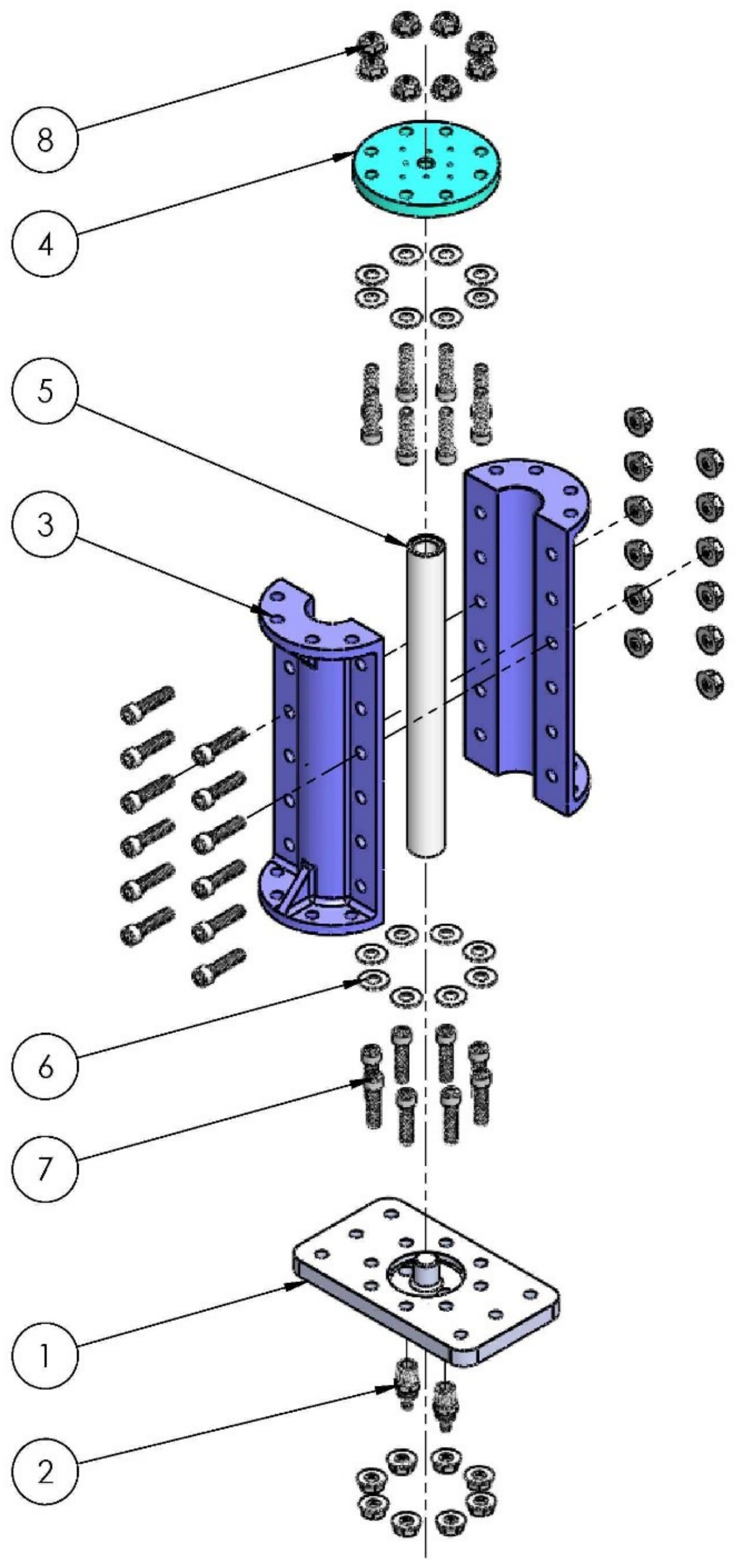


	UNLESS OTHERWISE SPECIFIED:	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Instructor:	DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	Lab Day	TITLE: Square Flange Cap
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH µm 0.6	SM By **Isobel Tetreau**	
	DO NOT SCALE DRAWING	TA Initials	SIZE **B** | Assignment Number | REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:5 | Mass: | SHEET 10 OF 17

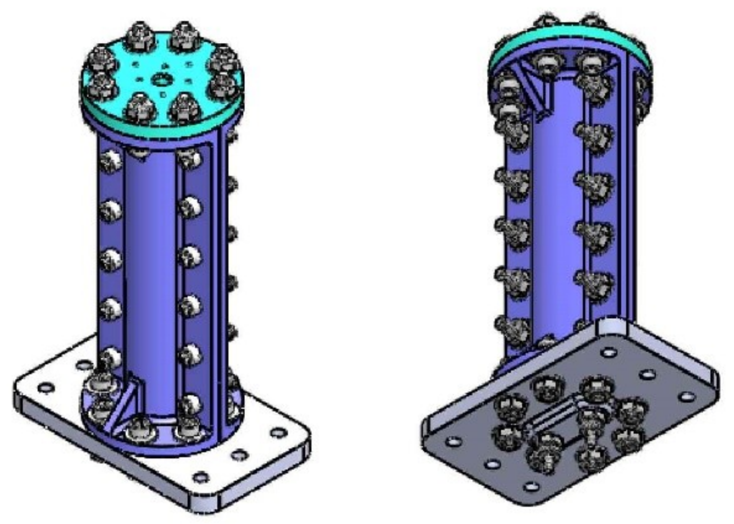




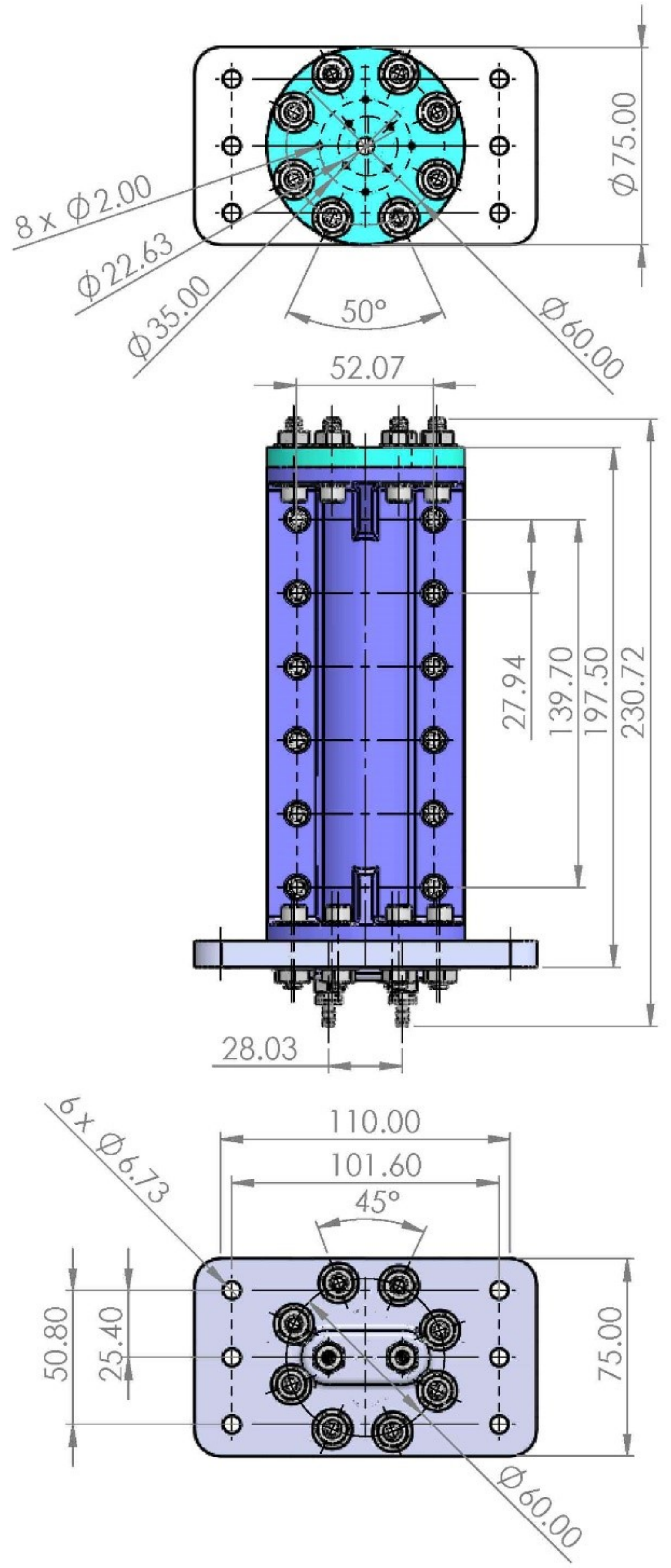
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Pump Connection Plate
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** Assignment Number REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:1.5 Mass: SHEET 11 OF 17

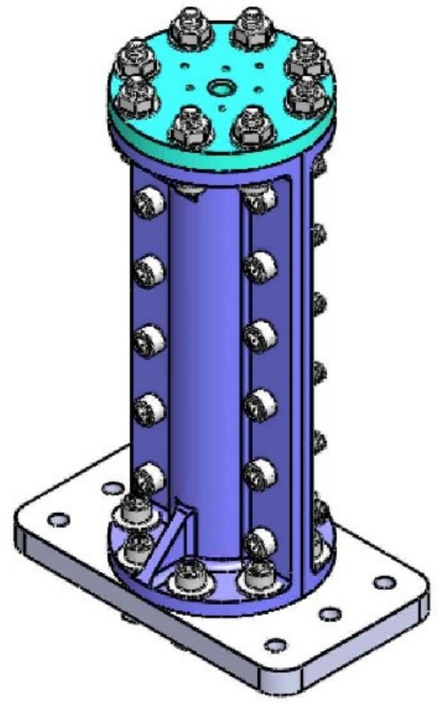


ITEM NO.	PART NAME	QTY.
1	Aorta mold base plate	1
2	0.125" barbed tube fitting	2
3	Aorta mold side piece	2
4	Aorta mold top piece	1
5	0.75" OD Delring tube core	1
6	0.25" flat washer	16
7	0.25" socket head cap screw	28
8	0.25" flange nut	28

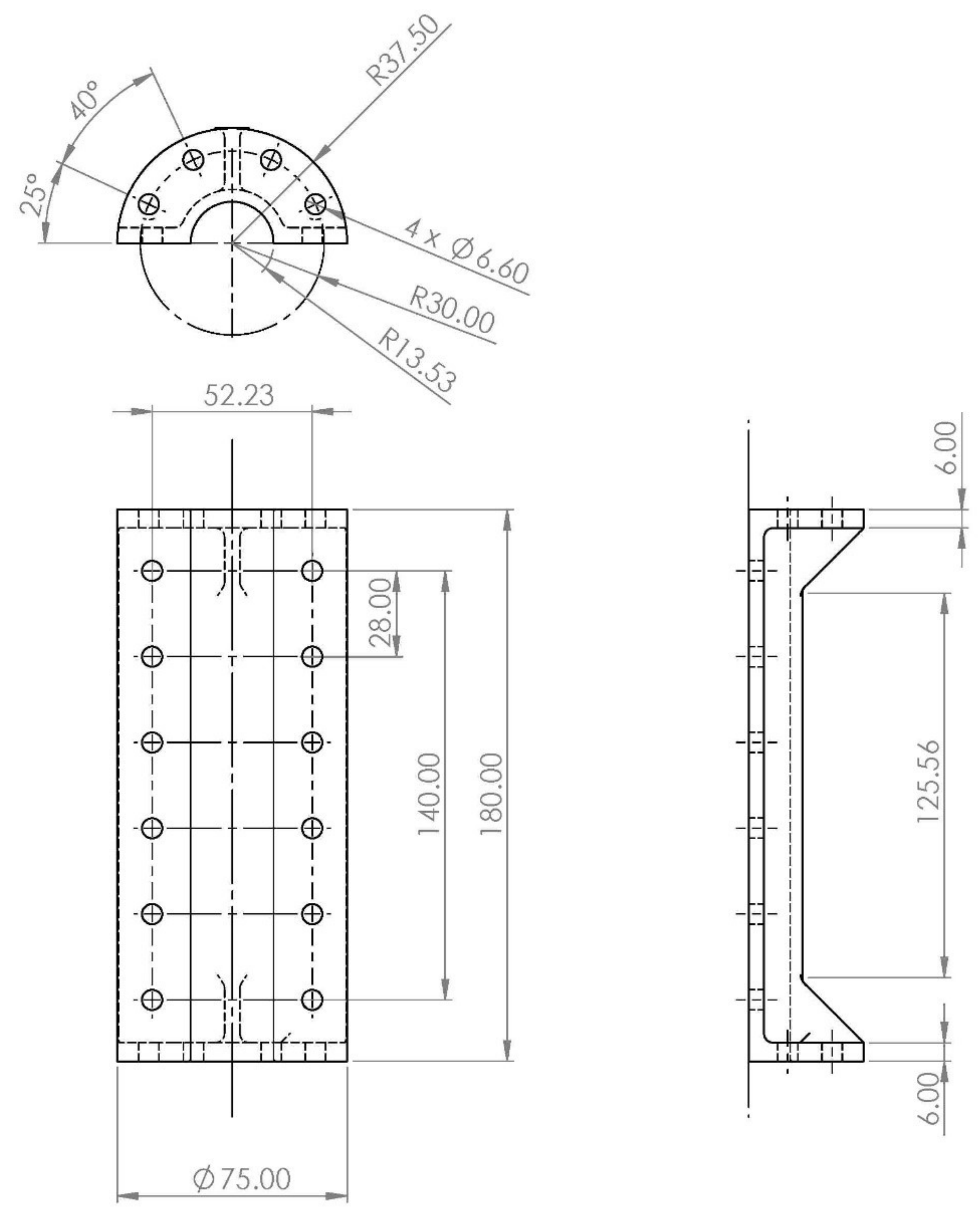


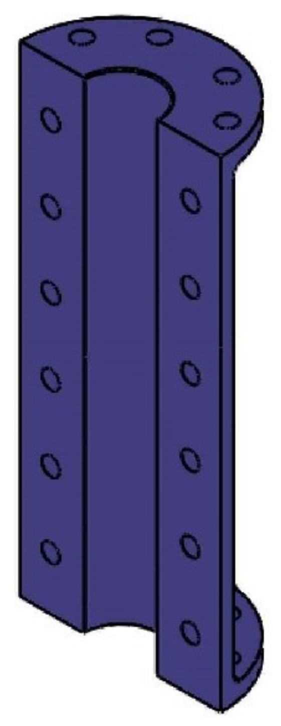
Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Aorta Mold Assembly
		SM By: **Isobel Tetreau and Sining Li**	
		TA Initials	SIZE **B** — Assignment Number — REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:3.5 — Mass: — SHEET 12 OF 17



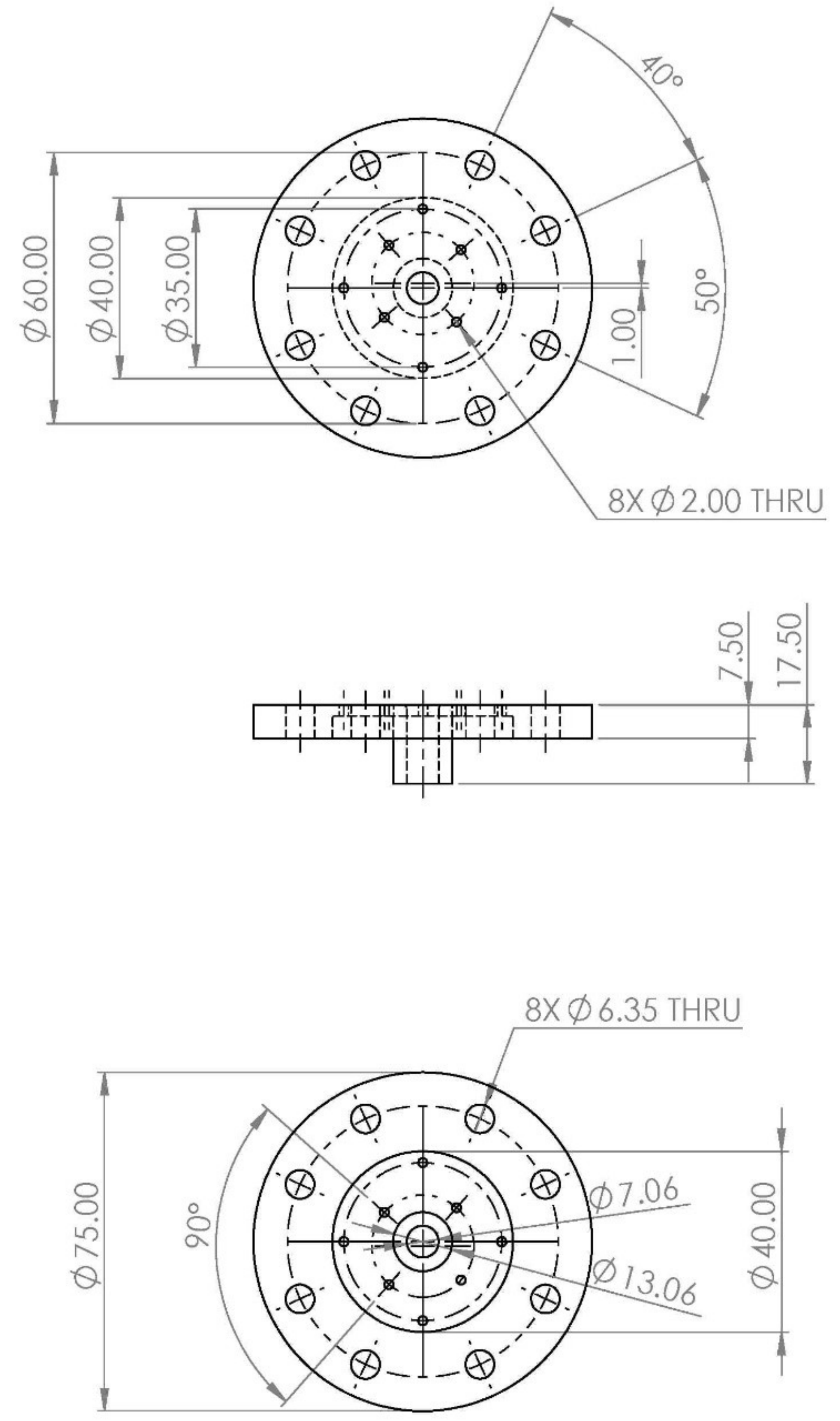


Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH µm 0.6	Lab Day	TITLE: Aorta mold assembly orthographic view
	DO NOT SCALE DRAWING	SM By **Isobel Tetreau**	SIZE **B** / Assignment Number / REV
MATERIAL:		TA Initials	
		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:2.5 Mass: SHEET 13 OF 17



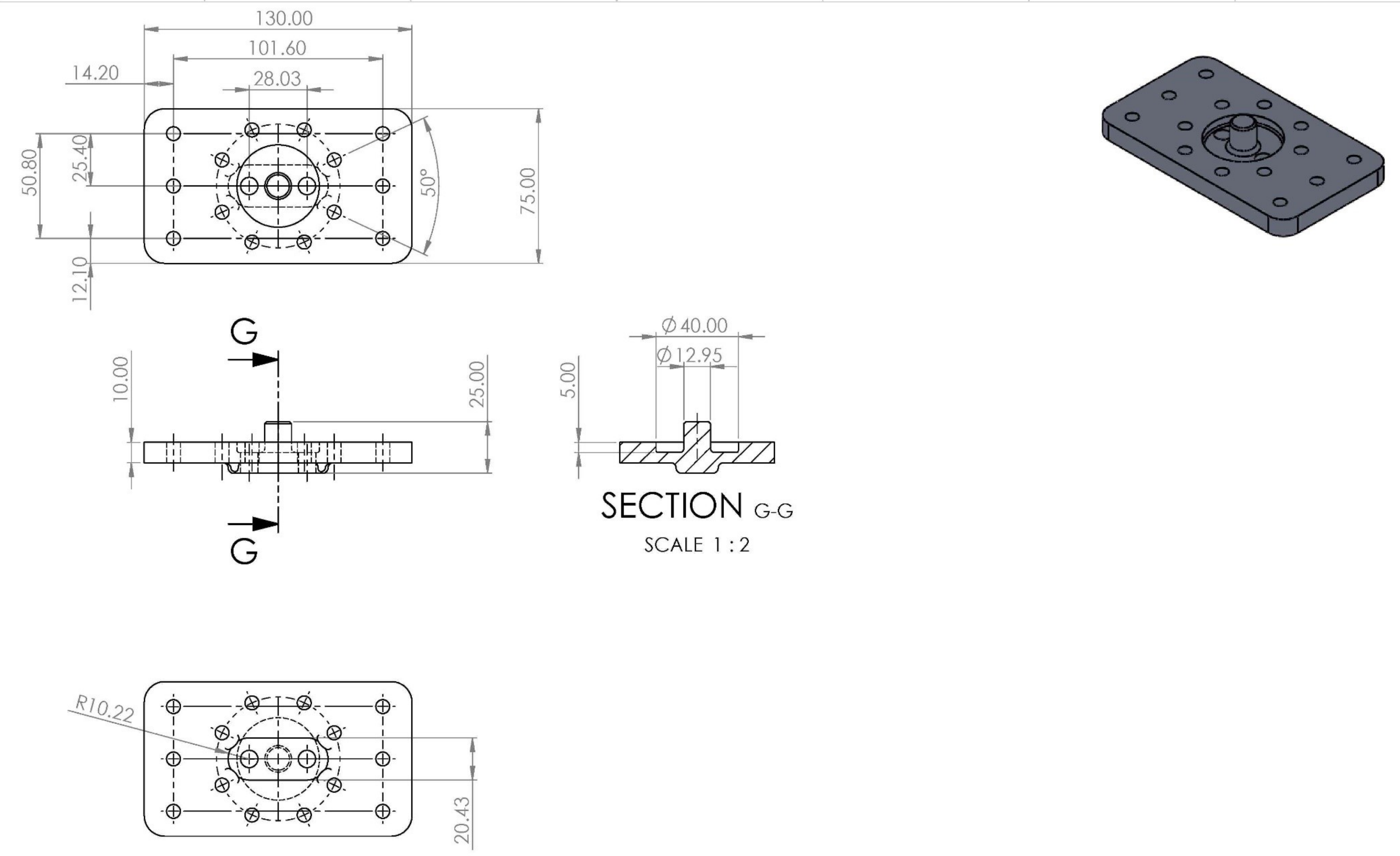


	UNLESS OTHERWISE SPECIFIED:	DRAWN BY:	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Instructor:	DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025	Sining Li	TITLE: Aorta mold side piece
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece	SURFACE FINISH 0.6 µm	Lab Day	
		SM By Isobel Tetreau	
	DO NOT SCALE DRAWING	TA Initials	SIZE B | Assignment Number | REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:2 | Mass: | SHEET 14 OF 17

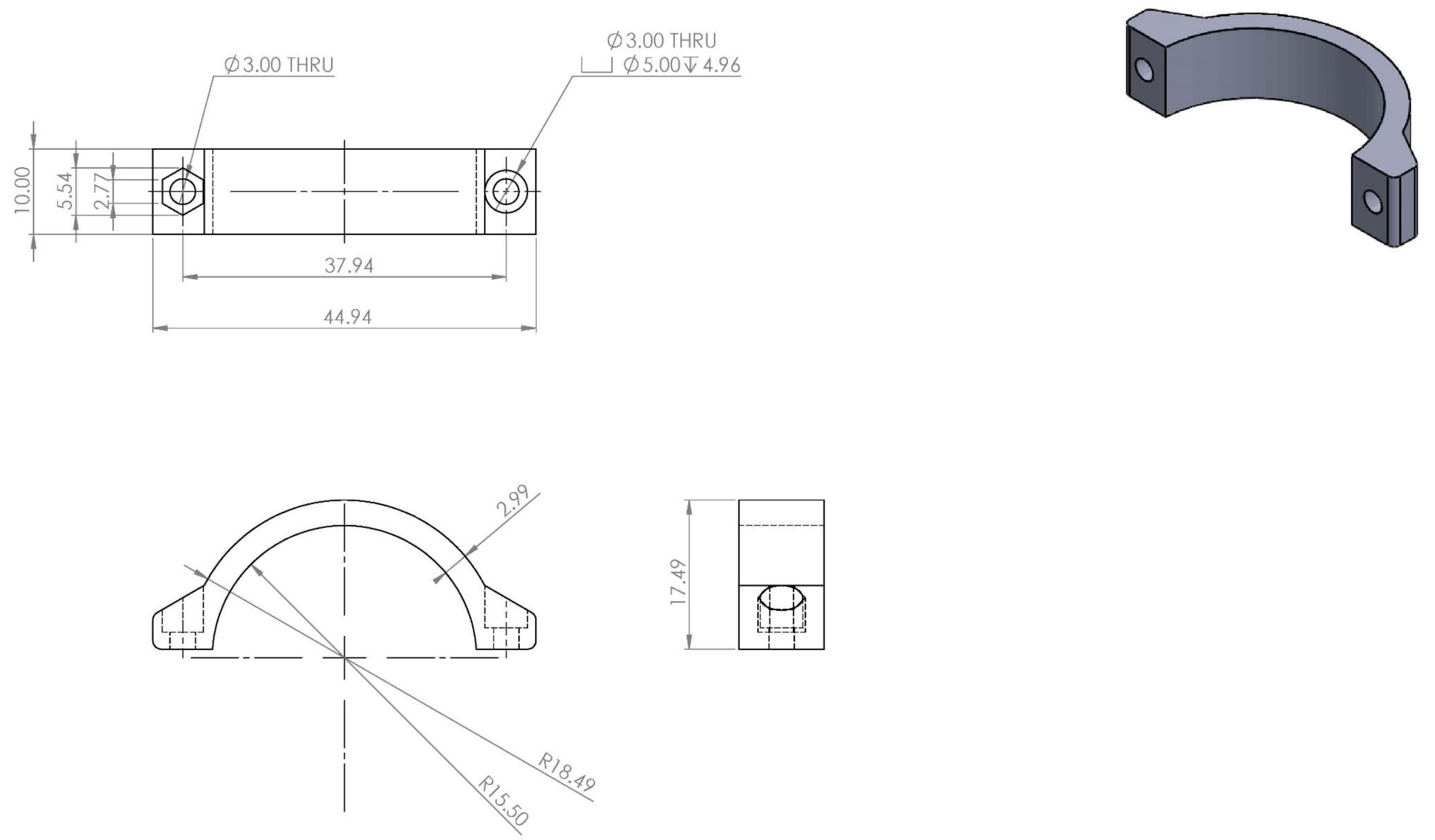


Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Aorta mold top piece
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** Assignment Number REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:1.5 Mass: SHEET 15 OF 17

SOLIDWORKS Educational Product. For Instructional Use Only.



Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: Aorta mold base plate
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** / Assignment Number / REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 1:5 / Mass: / SHEET 16 OF 17



Instructor:	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MM TOLERANCES: ANGULAR: ± 0.5° LINEAR X = ± 0.5 X.X = ± 0.1 X.XX = ± 0.025 SURFACE FINISH µm 0.6 DO NOT SCALE DRAWING	DRAWN BY: **Sining Li**	The Department of Mechanical Engineering UNIVERSITY OF ALBERTA
Comments: Some hidden lines removed for clarity. Valve casing was made in SolidWorks as two pieces, but 3D printed as one piece		Lab Day	TITLE: C-Clamp for aorta attachment
		SM By **Isobel Tetreau**	
		TA Initials	SIZE **B** / Assignment Number / REV
MATERIAL:		Jenny Desktop March 28, 2020 11:46:46 AM February 28, 2019 11:16:40 AM	
FILE NAME: Aorta Pressure Chamber Assembly Rev P			SCALE: 2:1 / Mass: / SHEET 17 OF 17

Appendix B. Code to Program Heart Waveform for Motor

Code author: Jake Hadfield, Department of Mechanical Engineering, University of Alberta

C++ code for motor

```
//Required include files
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include <time.h>
#include <Windows.h>
#include "pubSysCls.h"

using namespace sFnd;
bool stop = false;
size_t result;

// For reading in data
double * importAV(const char *instr) {
      FILE * pFile;
      long lSize;
      double * buffer;

      pFile = fopen(instr, "rb");
      if (pFile == NULL) { fputs("File error", stderr); exit(1); }

      // obtain file size:
      fseek(pFile, 0, SEEK_END);
      lSize = ftell(pFile);
      rewind(pFile);

      // allocate memory to contain the whole file:
      buffer = (double*)malloc(sizeof(double)*lSize);
      if (buffer == NULL) { fputs("Memory error", stderr); exit(2); }

      // copy the file into the buffer:
      result = fread(buffer, 8, lSize/8, pFile);
      if (result != lSize/8) { fputs("Reading error", stderr); exit(3);
}

      /* the whole file is now loaded in the memory buffer. */

      // terminate
      fclose(pFile);
      return buffer;
}

// Send message and wait for newline
```

```
void msgUser(const char *msg) {
	std::cout << msg;
	getchar();
}

// Allow user to break run loop sequence
DWORD WINAPI thread1(LPVOID pm)
{
	//check the getchar value
	int a = getchar();
	stop = true;
	return 0;
}

//for embedding time and date data in file name
char* mkFName(const struct tm *timeptr)
{
	static char result[34];
	sprintf(result, "MotorData_%.4d-%02d-%02d-%02d%02d%02d.data",
		1900+timeptr->tm_year,
		(timeptr->tm_mon)+1,
		timeptr->tm_mday, timeptr->tm_hour,
		timeptr->tm_min,
		timeptr->tm_sec);
	return result;
}

struct ARGS {
	INode &aNode;
};
// Grab position data
DWORD WINAPI thread2(LPVOID pm) {
	int i = 0;
	int daqArray[20000] = { 0 };
	int curPos;
	ARGS *args= (ARGS*)pm;
	INode &theNode = args->aNode;
	theNode.Status.Adv.CapturedHiResPosn.AutoRefresh(true);
	while (!stop) {
		if (i > 0)
		{
			curPos = theNode.Status.Adv.CapturedHiResPosn.Value();
			if (!(daqArray[i - 1] == curPos))
			{
				daqArray[i] = curPos;
				//printf("%i\n", curPos);
				i++;
			}
		}
		else
		{
```

```
                daqArray[i] =
theNode.Status.Adv.CapturedHiResPosn.Value();
                i++;
            }
      }
      int points=0;
      int j = 0;
      while (daqArray[j]>0 | daqArray[j]<0)
      {
            points++;
            j++;
      }
      time_t timer;
      struct tm * timeinfo;
      time(&timer);
      timeinfo = localtime(&timer);
      FILE *f = fopen(mkFName(timeinfo), "wb");

      fwrite(daqArray, sizeof(int), points, f);
      fclose(f);
      return 0;
};

//*****************************************************************
************
//This program will load configuration files onto each node connected
to the port, then executes
//sequential repeated moves on each axis.
//*****************************************************************
************

#define ACC_LIM_RPM_PER_SEC 1800
#define VEL_LIM_RPM             3000
int MOVE_DISTANCE_CNTS = 12000;
#define NUM_MOVES               1
#define TIME_TILL_TIMEOUT   20000 //The timeout used for homing(ms)

int main(int argc, char* argv[])
{
      msgUser("Motion Example starting. Press Enter to continue.");

      double *test[10];

      //double *pos=importAV("dis.bin");
      double *vel= importAV("vel.bin");
      //double *acc= importAV("acc.bin");

      size_t portCount = 0;
      std::vector<std::string> comHubPorts;
```

```
	//Create the SysManager object. This object will coordinate
actions among various ports
	// and within nodes. In this example we use this object to setup
and open our port.
	SysManager* myMgr = SysManager::Instance();
		//Create System Manager myMgr

	//This will try to open the port. If there is an error/exception
during the port opening,
	//the code will jump to the catch loop where detailed information
regarding the error will be displayed;
	//otherwise the catch loop is skipped over
	try
	{

		SysManager::FindComHubPorts(comHubPorts);
		printf("Found %d SC Hubs\n", comHubPorts.size());

		for (portCount = 0; portCount < comHubPorts.size() &&
portCount < NET_CONTROLLER_MAX; portCount++) {

			myMgr->ComHubPort(portCount,
comHubPorts[portCount].c_str()); 	//define the first SC Hub port
(port 0) to be associated
										// with COM
portnum (as seen in device manager)
		}

		if (portCount < 0) {

			printf("Unable to locate SC hub port\n");

			msgUser("Press any key to continue."); //pause so the
user can see the error message; waits for user to press a key

			return -1;  //This terminates the main program
		}
		//printf("\n I will now open port \t%i \n \n", portnum);
		myMgr->PortsOpen(portCount);				//Open the
port
		size_t i = 0;
		size_t iNode = 0;
		//for (size_t i = 0; i < portCount; i++) {
		IPort &myPort = myMgr->Ports(i);
		printf(" Port[%d]: state=%d, nodes=%d\n",
			myPort.NetNumber(), myPort.OpenState(),
myPort.NodeCount());



		//Once the code gets past this point, it can be assumed that
the Port has been opened without issue
```

```
            //Now we can get a reference to our port object which we
will use to access the node objects

            //for (size_t iNode = 0; iNode < myPort.NodeCount();
iNode++) {
                  // Create a shortcut reference for a node
            INode &theNode = myPort.Nodes(iNode);

            theNode.EnableReq(false);                  //Ensure Node is
disabled before loading config file

            myMgr->Delay(200);


            //theNode.Setup.ConfigLoad("Config File path");


            printf("   Node[%d]: type=%d\n", int(iNode),
theNode.Info.NodeType());
            printf("            userID: %s\n",
theNode.Info.UserID.Value());
            printf("        FW version: %s\n",
theNode.Info.FirmwareVersion.Value());
            printf("          Serial #: %d\n",
theNode.Info.SerialNumber.Value());
            printf("             Model: %s\n",
theNode.Info.Model.Value());

            //The following statements will attempt to enable the node.
First,
            // any shutdowns or NodeStops are cleared, finally the node
is enabled
            theNode.Status.AlertsClear();
      //Clear Alerts on node
            theNode.Motion.NodeStopClear();  //Clear Nodestops on Node

            theNode.EnableReq(true);                         //Enable
node
            //At this point the node is enabled
            printf("Node \t%zi enabled\n", iNode);
            //double timeout = myMgr->TimeStampMsec() +
TIME_TILL_TIMEOUT;    //define a timeout in case the node is unable to
enable
            //
                        //This will loop checking on the Real time values
of the node's Ready status
            //while (!theNode.Motion.IsReady()) {
            //    if (myMgr->TimeStampMsec() > timeout) {
            //          printf("Error: Timed out waiting for Node %d to
enable\n", iNode);
            //          msgUser("Press any key to continue."); //pause so
the user can see the error message; waits for user to press a key
```

```
          //          return -2;
          //     }
          //}
          ////At this point the Node is enabled, and we will now check
to see if the Node has been homed
          ////Check the Node to see if it has already been homed,
          //if (theNode.Motion.Homing.HomingValid())
          //{
          //     if (theNode.Motion.Homing.WasHomed())
          //     {
          //          printf("Node %d has already been homed, current
position is: \t%8.0f \n", iNode, theNode.Motion.PosnMeasured.Value());
          //          printf("Rehoming Node... \n");
          //     }
          //     else
          //     {
          //          printf("Node [%d] has not been homed.  Homing
Node now...\n", iNode);
          //     }
          //     //Now we will home the Node
          //     theNode.Motion.Homing.Initiate();

          //     timeout = myMgr->TimeStampMsec() + TIME_TILL_TIMEOUT;
     //define a timeout in case the node is unable to enable
          //
                    // Basic mode - Poll until disabled
          //     while (!theNode.Motion.Homing.WasHomed()) {
          //          if (myMgr->TimeStampMsec() > timeout) {
          //               printf("Node did not complete homing:  \n\t
-Ensure Homing settings have been defined through ClearView. \n\t -
Check for alerts/Shutdowns \n\t -Ensure timeout is longer than the
longest possible homing move.\n");
          //               msgUser("Press any key to continue.");
//pause so the user can see the error message; waits for user to press
a key
          //               return -2;
          //          }
          //     }
          //     printf("Node completed homing\n");
          //}
          //else {
          //     printf("Node[%d] has not had homing setup through
ClearView.  The node will not be homed.\n", iNode);
          //}

          //}


     /////////////////////////////////////////////////////////////////
//////////////////////
          //At this point we will execute 10 rev moves sequentially on
each axis
```

```
      ////////////////////////////////////////////////////////////////
/////////////////////

           //for (size_t i = 0; i < NUM_MOVES; i++)
           //{
           //    for (size_t iNode = 0; iNode < myPort.NodeCount();
iNode++) {
                       //Create a shortcut reference for a node
                      //INode &theNode = myPort.Nodes(iNode);

           theNode.Motion.MoveWentDone();
      //Clear the rising edge Move done register

           theNode.AccUnit(INode::COUNTS_PER_SEC2);
      //Set the units for Acceleration to RPM/SEC
           theNode.VelUnit(INode::COUNTS_PER_SEC);
           //Set the units for Velocity to RPM
           //theNode.AccUnit(INode::RPM_PER_SEC);
           //theNode.VelUnit(INode::RPM);

           //be ready to break loop
           printf("Press return key to stop.\n");
           //Start the threads to break the loop and record data
           HANDLE handle = CreateThread(NULL, 0, thread1, NULL, 0,
NULL);
           //test
           ARGS args = { theNode };
           //HANDLE handle2 = CreateThread(NULL, 0, thread2, &args, 0,
NULL);
           //theNode.Motion.PosnMeasured.AutoRefresh(true);
           //theNode.Status.Adv.CapturedHiResPosn.Refresh();
           int step = 0;
           int32_t inter;
           theNode.Motion.AccLimit = 100000;
           theNode.Motion.VelLimit = 30000;
           double vlim;
           if (vel[result - 1] < 0) {
                vlim = -vel[result - 1];
           }
           else{
                vlim = vel[result - 1];
           }
           //theNode.Motion.VelLimit = vlim;
           theNode.Motion.PosnMeasured.Refresh();
           inter = theNode.Motion.PosnMeasured.Value();
           int k = 0;
           int moveval = 21350;
           result = 3;
           while (!stop & k<200) {
                step = 0;
                k++;
                while (step < (result-1)) {
                      //theNode.Motion.MoveVelStart(-vel[step]);
      //accelerate to desired speed
                      theNode.Motion.MovePosnStart(moveval);

                      //if (pos[step+1] > pos[step])
                      //{
                      //
      theNode.Status.Adv.CapturedHiResPosn.Refresh();
                      //      inter =
theNode.Status.Adv.CapturedHiResPosn.Value();
                      //      while (inter < pos[step]) {
                      //
      theNode.Status.Adv.CapturedHiResPosn.Refresh();
                      //              inter =
theNode.Status.Adv.CapturedHiResPosn.Value();
                      //              printf("%i", inter);
                      //      }
                      //} //wait til we pass position
                      //else {
                      //
      theNode.Status.Adv.CapturedHiResPosn.Refresh();
                      //      inter =
theNode.Status.Adv.CapturedHiResPosn.Value();
                      //      while (inter > pos[step]) {
                      //
      theNode.Status.Adv.CapturedHiResPosn.Refresh();
                      //              inter =
theNode.Status.Adv.CapturedHiResPosn.Value(); }
                      //}
                      //while (!theNode.Motion.VelocityReachedTarget())
{}
                      step++;
                      while (!theNode.Motion.MoveIsDone()) {}
                      moveval = -moveval;
                }
                //theNode.Motion.MovePosnStart(inter,true);
           }
           //} // for each node
      //} // for each move



////////////////////////////////////////////////////////////////////////
///////////////////////
//After moves have completed Disable node, and close ports
////////////////////////////////////////////////////////////////////////
///////////////////////
           printf("Disabling nodes, and closing port\n");
           //Disable Nodes
```

```
			for (size_t iNode = 0; iNode < myPort.NodeCount(); iNode++)
{
				// Create a shortcut reference for a node
				myPort.Nodes(iNode).EnableReq(false);
			}
			//}
		}
		catch (mnErr& theErr)
		{
			printf("Failed to disable Nodes n\n");
			//This statement will print the address of the error, the
error code (defined by the mnErr class),
			//as well as the corresponding error message.
			printf("Caught error: addr=%d, err=0x%08x\nmsg=%s\n",
theErr.TheAddr, theErr.ErrorCode, theErr.ErrorMsg);

			msgUser("Press any key to continue."); //pause so the user
can see the error message; waits for user to press a key
			return 0;  //This terminates the main program
		}

		// Close down the ports
		myMgr->PortsClose();

		printf("Saving output file...\n");

		WaitForSingleObject(thread2, INFINITE);

		msgUser("Press any key to close program."); //pause so the user
can see the error message; waits for user to press a key
		return 0;			//End program
}
```

Matlab codes to import and create desired heart profiles

```
function data=motorRead(floc)
if nargin==0;
    floc='C:\Program Files (x86)\Teknic\ClearView\sdk\examples\3a-Example-
Motion\motorData_2019_02_13_17_05_57.data';
end
f=fopen(floc);
data=fread(f,'int32');
fclose(f);
end
```

Published with MATLAB® R2019a

```
function createMotorProfile
close all
movfreq=60;
movdist=8000;
v1=[(0:1/movfreq:1)' movdist/2*(cos(0:pi/movfreq*2:2*pi))'];
% custom rounding, adjust time to make steps round instead of hard
% rounding
v3=inteval(v1,round(v1(:,2)));
% figure
% plot(v3(:,1),v3(:,2))
v3=v3(1:end-1,:);
d(:,:)=[-v3(1:end-1,:)+v3(2:end,:);-v3(end,:)+v3(1,:)];
d(end,1)=1+d(end,1);
d(end,1)=d(end,1)*d(end,2)/d(end-1,2);
d(:,1)=d(:,1)/sum(d(:,1));
vf=(d(:,2)./d(:,1));
% vfa=[v3(end,2); v3(:,2); v3(1,2)];
% a=(vfa(3:end)+vfa(1:(end-2))-2*vfa(2:(end-1)))./d(:,1).^2;
% f=fopen('dis.bin','wb');
% fwrite(f,v3(2:end,2),'double');
% fclose(f);
g=fopen('vel.bin','wb');
fwrite(g,-vf,'double');
fclose(g);
% h=fopen('acc.bin','wb');
% fwrite(h,a(2:end),'double');
% fclose(h);
end

function d=inteval(a,p)
a=[a;[a(1,1)+1 a(1,2)]];
p=[p;p(1)];
times=zeros(length(a)-1,1);
for i=1:length(a)-1
    if p(i)<a(i)
        times(i,:)=(a(i,1)-a(i-1,1)).*(p(i)-a(i-1,2))./(a(i,2)-a(i-1,2))+a(i-1,1);
    else
        times(i,:)=(a(i+1,1)-a(i,1)).*(p(i)-a(i,2))./(a(i+1,2)-a(i,2))+a(i,1);
    end
end
% times=(a(2:end,1)-a(1:(end-1),1)).*(p(1:end-1)-a(1:(end-1),2))./(a(2:end,2)-a(1:(end-
1),2))+a(1:(end-1),1);
d=[times(1:(end),:) p(1:(end-1),:)];
end
```

Published with MATLAB® R2019a

```
function importMotorProfile
close all
movfreq=round(500*0.5);
dsamp=10;
```

```
movdist=round(22000*1.05); %22000 is the default. 0.8 is _just_ right.
v1=importdata('Voldata.csv',',');
% v1=sin(0:pi/128:2*pi);
rep=length(v1)-1;
v1(:,1)=v1(:,1)/max(v1(:,1));
v1(1,:)=[0 v1(end,2)];
v1(end,:)=[];
v1=[(v1(end-rep+1:end,:)-2*repmat([1 0],rep,1));(v1(end-rep+1:end,:)-repmat([1
0],rep,1));v1;(v1(1:rep,:)+repmat([1 0],rep,1));(v1(1:rep,:)+2*repmat([1 0],rep,1))];
% plot(v1(:,1),v1(:,2))
% hold on
% lpFilt = designfilt('lowpassiir','FilterOrder',12, ...
%     'HalfPowerFrequency',0.05,'DesignMethod','butter');
% v1(:,2)=filtfilt(lpFilt,v1(:,2));
% plot(v1(:,1),v1(:,2))
[v2(:,2),v2(:,1)]=resample(v1(:,2),v1(:,1),movfreq,'pchip');
lpFilt = designfilt('lowpassiir','FilterOrder',12, ...
    'HalfPowerFrequency',.02,'DesignMethod','butter');
v2(:,2)=filtfilt(lpFilt,v2(:,2));
v2=v2((3*movfreq/2):(7*movfreq/2),:);
% hold on
% plot(v2(:,1),v2(:,2),'k')
% figure
[pks,locs]=findpeaks(v2(:,2));
v3=v2(locs(1):locs(2),:);
v3=v3(1:dsamp:end,:);
v3(:,2)=v3(:,2)-min(v3(:,2));
v3(:,2)=1-v3(:,2)/max(v3(:,2));
v3(:,2)=v3(:,2)*movdist;
ind=find(v3(:,2)==0);
v3(:,2)=[v3(ind:end,2);v3(1:ind-1,2)];
v3(:,1)=v3(:,1)-v3(1,1);
v3(:,1)=v3(:,1)/(max(v3(:,1))-v3(1,1)+v3(2,1));
% figure
% plot(v3(:,1),v3(:,2))
% d(:,2)=[-v3(1:end-1,2)+v3(2:end,2);-v3(end,2)+v3(1,2)];
% custom rounding, adjust time to make steps round instead of hard
% rounding
v3=inteval(v3,round(v3(:,2)));
% figure
% plot(v3(:,1),v3(:,2))
d(:,:)=[-v3(1:end-1,:)+v3(2:end,:);-v3(end,:)+v3(1,:)];
% v3=[v3(2:end,:);v3(1,:)];
d(end,1)=1+d(end,1);
d(end,1)=d(end,1)*d(end,2)/d(end-1,2);
d(:,1)=d(:,1)/sum(d(:,1));
vf=(d(:,2)./d(:,1));
% vfa=[v3(end,2); v3(:,2); v3(1,2)];
% a=(vfa(3:end)+vfa(1:(end-2))-2*vfa(2:(end-1)))./d(:,1).^2;
% f=fopen('dis.bin','wb');
% fwrite(f,v3(2:end,2),'double');
% fclose(f);
vf(vf<0)=[];
%g=fopen('vel-h_f-150_sv-454.bin','wb');
```

```
g=fopen('vel.bin','wb');
fwrite(g,vf(2:end),'double');
fclose(g);
% h=fopen('acc.bin','wb');
% fwrite(h,a(2:end),'double');
% fclose(h);
end

function d=inteval(a,p)
a=[a;[a(1,1)+1 a(1,2)]];
p=[p;p(1)];
times=zeros(length(a)-1,1);
for i=1:length(a)-1
    if p(i)<a(i)
        times(i,:)=(a(i,1)-a(i-1,1)).*(p(i)-a(i-1,2))./(a(i,2)-a(i-1,2))+a(i-1,1);
    else
        times(i,:)=(a(i+1,1)-a(i,1)).*(p(i)-a(i,2))./(a(i+1,2)-a(i,2))+a(i,1);
    end
end
% times=(a(2:end,1)-a(1:(end-1),1)).*(p(1:end-1)-a(1:(end-1),2))./(a(2:end,2)-a(1:(end-
1),2))+a(1:(end-1),1);
d=[times(1:(end),:) p(1:(end-1),:)];
end
```

Published with MATLAB® R2019a

Appendix C.   MATLAB Code for Data Collection GUI

```
function varargout = V1(varargin)
% V1 MATLAB code for V1.fig
%      V1, by itself, creates a new V1 or raises the existing
%      singleton*.
%
%      H = V1 returns the handle to a new V1 or the handle to
%      the existing singleton*.
%
%      V1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in V1.M with the given input arguments.
%
%      V1('Property','Value',...) creates a new V1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before V1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to V1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help V1

% Last Modified by GUIDE v2.5 15-May-2019 12:39:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @V1_OpeningFcn, ...
    'gui_OutputFcn',  @V1_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT


% --- Executes just before V1 is made visible.
function V1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to V1 (see VARARGIN)

% Choose default command line output for V1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes V1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = V1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




function RateInput_Callback(hObject, eventdata, handles)
% hObject    handle to RateInput (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of RateInput as text
%        str2double(get(hObject,'String')) returns contents of RateInput as a double



% --- Executes during object creation, after setting all properties.
function RateInput_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RateInput (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function CaptureTimeInput_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to CaptureTimeInput (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CaptureTimeInput as text
%        str2double(get(hObject,'String')) returns contents of CaptureTimeInput as a double

% --- Executes during object creation, after setting all properties.
function CaptureTimeInput_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CaptureTimeInput (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in CaptureDataButton.
function CaptureDataButton_Callback(hObject, eventdata, handles)
% hObject    handle to CaptureDataButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(hObject,'Enable','off');

rate = str2double(get(handles.RateInput,'string'));
captime = str2double(get(handles.CaptureTimeInput,'string'));
%the user inputs
cla
%clears the current axes
% Create a session for the specified vendor.
vend = daq.getVendors() ;
vend = vend.ID ;
s = daq.createSession(vend);
% Set properties that are not using default values.
s.Rate = rate ;
s.NumberOfScans = captime*rate ;
s.IsContinuous=0;
%calculate the number of scans and display to the user
textlabel = sprintf('%d scans', s.NumberOfScans );
set(handles.TotalNumberOfScans, 'String', textlabel);

% Add channels and set channel properties, if any.

%for inputs
columnread=1;

c = get(handles.ai0Button,'Value') ;
if c
    channel1 = addAnalogInputChannel(s,'Dev1','ai0','Voltage');
    channel1.TerminalConfig = 'SingleEnded';
    channel1.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai1Button,'Value') ;
if c
    channel2 = addAnalogInputChannel(s,'Dev1','ai1','Voltage');
    channel2.TerminalConfig = 'SingleEnded';
    channel2.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai2Button,'Value') ;
if c
    channel3 = addAnalogInputChannel(s,'Dev1','ai2','Voltage');
    channel3.TerminalConfig = 'SingleEnded';
    channel3.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai3Button,'Value') ;
if c
    channel4 = addAnalogInputChannel(s,'Dev1','ai3','Voltage');
    channel4.TerminalConfig = 'SingleEnded';
    channel4.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai4Button,'Value') ;
if c
    channel5 = addAnalogInputChannel(s,'Dev1','ai4','Voltage');
    channel5.TerminalConfig = 'SingleEnded';
    channel5.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai5Button,'Value') ;
if c
    channel6 = addAnalogInputChannel(s,'Dev1','ai5','Voltage');
    channel6.TerminalConfig = 'SingleEnded';
    channel6.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai6Button,'Value') ;
if c
    channel7 = addAnalogInputChannel(s,'Dev1','ai6','Voltage');
    channel7.TerminalConfig = 'SingleEnded';
    channel7.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end

c = get(handles.ai7Button,'Value') ;
if c
```

```
    channel8 = addAnalogInputChannel(s,'Dev1','ai7','Voltage');
    channel8.TerminalConfig = 'SingleEnded';
    channel8.Range = [-10.000000 10.000000];
    columnread=columnread+1;
end
```

Set motor code's vel.bin file to the correct one based on user input

```
w_waveform=get(get(handles.waveform,'SelectedObject'),'Tooltip');
f_frequency=get(get(handles.frequency,'SelectedObject'),'Tooltip');
s_strokevolume=get(get(handles.strokevolume,'SelectedObject'),'Tooltip');

chosen_vel_id=fopen(['vel-' w_waveform '_f-' f_frequency '_sv-' s_strokevolume '.bin'],'r');
chosen_vel=fread(chosen_vel_id);

motor_vel=fopen('vel.bin','w');
fwrite(motor_vel, chosen_vel);

fclose(motor_vel);
fclose(chosen_vel_id);
```

Create a serial port object.

```
fg = instrfind('Type', 'serial', 'Port', 'COM9', 'Tag', '');
% Create the serial port object if it does not exist
% otherwise use the object that was found.
if isempty(fg)
    fg = serial('COM9');
else
    fclose(fg);
    fg = fg(1);
end

fopen(fg);
fprintf(fg,'TRIGGER:SOURCE BUS');
```

Open log file and create listener to prepare gathering data

```
fid1 = fopen('log.bin','w');
lh =
addlistener(s,'DataAvailable',@(src,event)fwrite(fid1,[event.TimeStamps,event.Data]','double'));
```

Start acquisition

```
startBackground(s);
```

```
% %% Trigger the function generator
pause('on')
pause(str2double(get(handles.CameraDelay,'String'))) %wait 1 s before starting the camera to
ensure data collection occurs after DAQ initiates
fprintf(fg, '*TRG');
fclose(fg);
delete(fg);
```

Start the motor

```
pause(str2double(get(handles.MotorDelay,'String'))) %wait 1 s before starting the motor to ensure
image of static tube is gathered before pump starts

!start MoveMyTeknicMatlab.lnk
```

Wait for data collection to complete

```
wait(s)

delete(lh);
fclose(fid1);
fid2 = fopen('log.bin','r');
[data,count] = fread(fid2,[columnread,inf],'double');
fclose(fid2);

clear fid1 fid2
delete log.bin

timestamps = transpose(data(1,:));
dataplot = transpose(data(2:end,:));
```

create folders

```
str = datestr(now, 'yyyy-mm-dd-HHMMSS');
filesavetext=get(handles.FileSaveTextDisplay,'String');

folderdate=datestr(now, 'yyyy-mm-dd');

mkdir(['datacollected\' folderdate '\data']);
mkdir(['datacollected\' folderdate '\images\' filesavetext(1:end-12)]);
```

write txt file

```
str = datestr(now, 'yyyy-mm-dd-HHMMSS') ;
filesavetext=get(handles.FileSaveTextDisplay,'String');
fname = strcat(filesavetext(1:end-12), '_', str, '.txt') ;
```

```
fileID = fopen(['datacollected\' folderdate '\data\' fname],'wt');
date= datestr(now);
formatspec = ('Date:   %s     \r \n');
fprintf(fileID, formatspec, date);

fprintf(fileID, 'Run Time (s) :   %d', captime);
fprintf(fileID, '\n');
fprintf(fileID, 'Rate (Hz) :  %d', rate);
fprintf(fileID, '\n');
fprintf(fileID, 'Samples Per Channel :  %d', s.NumberOfScans);
fprintf(fileID, '\n');

%get the rest of the information needed for the text file
txtoutput = timestamps ;

% fprintf(fileID, 'SerialTime                  ');
fprintf(fileID, 'Timestamp              ');

% Convert the acquired data and timestamps to a timetable in a workspace variable.
k = 2; hold on

c = get(handles.ai0Button,'Value') ;
if c
    ai0 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai0);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai0 ;
    plotlegend{k,1}='ai0';
    fprintf(fileID, 'ai0                    ');
    k = k+1 ;
end

c = get(handles.ai1Button,'Value') ;
if c
    ai1 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai1);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai1 ;
    plotlegend{k,1}='ai1';
    fprintf(fileID, 'ai1                    ');
    k = k+1 ;
end

c = get(handles.ai2Button,'Value') ;
if c
    ai2 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai2);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai2 ;
    plotlegend{k,1}='ai2';
    fprintf(fileID, 'ai2                    ');
    k = k+1 ;
end
```

```
c = get(handles.ai3Button,'Value') ;
if c
    ai3 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai3);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai3 ;
    plotlegend{k,1}='ai3';
    fprintf(fileID, 'ai3                    ');
    k = k+1;
end

c = get(handles.ai4Button,'Value') ;
if c
    ai4 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai4);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai4 ;
    plotlegend{k,1}='ai4';
    fprintf(fileID, 'ai4                    ');
    k=k+1;
end

c = get(handles.ai5Button,'Value') ;
if c
    ai5 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai5);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai5 ;
    plotlegend{k,1}='ai5';
    fprintf(fileID, 'ai5                    ');
    k=k+1;
end

c = get(handles.ai6Button,'Value') ;
if c
    ai6 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai6);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai6 ;
    plotlegend{k,1}='ai6';
    fprintf(fileID, 'ai6                    ');
    k=k+1;
end

c = get(handles.ai7Button,'Value') ;
if c
    ai7 = transpose(data(k,:));
    DAQ_3 = timetable(seconds(timestamps),ai7);
    plot(DAQ_3.Time, DAQ_3.Variables)
    txtoutput(:,k) = ai7 ;
    plotlegend{k,1}='ai7';
    fprintf(fileID, 'ai7                    ');
    k=k+1;
```

```
end

%put the txtoutput into the txt file
fprintf(fileID, '\n');
[r,c] = size(txtoutput) ;
for i = 1:r
    for j = 1:c
        fprintf(fileID, '%.16f     ', txtoutput(i,j)) ;
    end
    fprintf(fileID, '\n') ;
end

legend(plotlegend{1:end,1})%(DAQ_3.Properties.VariableNames)
% xlabel('Time')
% ylabel('Amplitude (V)')

% Clear the session and channels, if any.
hold off

checklist=fopen(['datacollected\' 'checklist_' folderdate '.txt'], 'w+');
fprintf(checklist, filesavetext(1:end-12));

% if exist(['datacollected\' 'checklist_' folderdate '.txt'])~=0
% checklist=fopen(['datacollected\' 'checklist_' folderdate '.txt'], 'r');
% i=1;
% tline=fgetl(checklist);
% A{i} = tline;
% while ischar(tline)
%     i = i+1;
%     tline = fgetl(checklist);
%     A{i} = tline;
% end
% Adone=strcmp(A, filesavetext(1:end-12));
% A{Adone}=sprintf([filesavetext(1:end-12) '\t' 'done' '\n']);
% fclose(checklist);
% checklist=fopen(['datacollected\' 'checklist_' folderdate '.txt'], 'w');
% for i = 1:numel(A)
%     if A{i+1} == -1
%         fprintf(checklist,'%s', A{i});
%         break
%     else
%         fprintf(checklist,'%s \n', A{i});
%     end
% end
% end


fclose all;
set(hObject,'Enable','on');
clear;


% --- Executes on button press in ai0Button.
```

```
function ai0Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai0Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai0Button


% --- Executes on button press in ai1Button.
function ai1Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai1Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai1Button


% --- Executes on button press in ai2Button.
function ai2Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai2Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai2Button


% --- Executes on button press in ai3Button.
function ai3Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai3Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai3Button


% --- Executes on button press in ai4Button.
function ai4Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai4Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai4Button


% --- Executes on button press in ai5Button.
function ai5Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai5Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai5Button


% --- Executes on button press in ai6Button.
```

```
function ai6Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai6Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai6Button


% --- Executes on button press in ai7Button.
function ai7Button_Callback(hObject, eventdata, handles)
% hObject    handle to ai7Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ai7Button


% --- Executes on button press in LiveOnButton.
function LiveOnButton_Callback(hObject, eventdata, handles)
% hObject    handle to LiveOnButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of LiveOnButton

%clear current axis
cla

%user inputs
rate = str2double(get(handles.RateInput,'string'));
captime = str2double(get(handles.CaptureTimeInput,'string'));

% Create a session for the specified vendor.
vend = daq.getVendors() ;
vend = vend.ID ;
s = daq.createSession(vend);
% Set properties that are not using default values.
s.Rate = rate ;
s.NumberOfScans = captime*rate ;
s.IsContinuous=1;
%calculate the number of scans and display to the user

% Add channels and set channel properties, if any.

%for inputs

c = get(handles.ai0Button,'Value') ;
if c
    channel1 = addAnalogInputChannel(s,'Dev1','ai0','Voltage');
    channel1.TerminalConfig = 'SingleEnded';
    channel1.Range = [-10.000000 10.000000];
end

c = get(handles.ai1Button,'Value') ;
```

```
if c
    channel2 = addAnalogInputChannel(s,'Dev1','ai1','Voltage');
    channel2.TerminalConfig = 'SingleEnded';
    channel2.Range = [-10.000000 10.000000];
end

c = get(handles.ai2Button,'Value') ;
if c
    channel3 = addAnalogInputChannel(s,'Dev1','ai2','Voltage');
    channel3.TerminalConfig = 'SingleEnded';
    channel3.Range = [-10.000000 10.000000];
end

c = get(handles.ai3Button,'Value') ;
if c
    channel4 = addAnalogInputChannel(s,'Dev1','ai3','Voltage');
    channel4.TerminalConfig = 'SingleEnded';
    channel4.Range = [-10.000000 10.000000];
end

c = get(handles.ai4Button,'Value') ;
if c
    channel5 = addAnalogInputChannel(s,'Dev1','ai4','Voltage');
    channel5.TerminalConfig = 'SingleEnded';
    channel5.Range = [-10.000000 10.000000];
end

c = get(handles.ai5Button,'Value') ;
if c
    channel6 = addAnalogInputChannel(s,'Dev1','ai5','Voltage');
    channel6.TerminalConfig = 'SingleEnded';
    channel6.Range = [-10.000000 10.000000];
end

c = get(handles.ai6Button,'Value') ;
if c
    channel7 = addAnalogInputChannel(s,'Dev1','ai6','Voltage');
    channel7.TerminalConfig = 'SingleEnded';
    channel7.Range = [-10.000000 10.000000];
end

c = get(handles.ai7Button,'Value') ;
if c
    channel8 = addAnalogInputChannel(s,'Dev1','ai7','Voltage');
    channel8.TerminalConfig = 'SingleEnded';
    channel8.Range = [-10.000000 10.000000];
end

lh = addlistener(s,'DataAvailable',@(src,event)plot(event.TimeStamps,event.Data));
```

Start acquisition

```
startBackground(s);

set(hObject,'Enable','off');
set(handles.LiveModeTooltip,'Visible','on');

waitforbuttonpress;

stop(s)

delete(lh);

set(handles.LiveModeTooltip,'Visible','off');
set(hObject,'Enable','on');

clear s channel1 channel2 channel3 channel4 channel5 channel6 channel7  k

% --- Executes on button press in LiveButtonOff.
function LiveButtonOff_Callback(hObject, eventdata, handles)
% hObject    handle to LiveButtonOff (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of LiveButtonOff


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over LiveOnButton.
function LiveOnButton_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to LiveOnButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on key press with focus on LiveOnButton and none of its controls.
function LiveOnButton_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to LiveOnButton (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%      Key: name of the key that was pressed, in lower case
%      Character: character interpretation of the key(s) that was pressed
%      Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)



function FileSaveTextInput_Callback(hObject, eventdata, handles)
% hObject    handle to FileSaveTextInput (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of FileSaveTextInput as text
%        str2double(get(hObject,'String')) returns contents of FileSaveTextInput as a double
%filesavetext=[a_aorta '_' l_length '_' v_valve '_' w_waveform '_' b_backpressure '_'
c_chamberpressure '_datecode.txt'];

% filesavetext=[...
%     'a-' get(get(handles.aorta,'SelectedObject'), 'String') '_' ...
%     'l-' get(get(handles.length,'SelectedObject'), 'String') '_' ...
%     'v-' get(get(handles.valve,'SelectedObject'), 'String') '_' ...
%     'w-' get(get(handles.waveform,'SelectedObject'), 'String') '_' ...
%     'b-' get(get(handles.backpressure,'SelectedObject'), 'String') '_' ...
%     'c-' get(get(handles.chamberpressure,'SelectedObject'), 'String') '_' ...
%     'datecode.txt'];
% set(hObject, 'String', filesavetext);

% --- Executes during object creation, after setting all properties.
function FileSaveTextInput_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FileSaveTextInput (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over TotalNumberOfScans.
function TotalNumberOfScans_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to TotalNumberOfScans (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function CameraFrequency_Callback(hObject, eventdata, handles)
% hObject    handle to CameraFrequency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CameraFrequency as text
%        str2double(get(hObject,'String')) returns contents of CameraFrequency as a double


% --- Executes during object creation, after setting all properties.
function CameraFrequency_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CameraFrequency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function MotorFrequency_Callback(hObject, eventdata, handles)
% hObject    handle to MotorFrequency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MotorFrequency as text
%        str2double(get(hObject,'String')) returns contents of MotorFrequency as a double


% --- Executes during object creation, after setting all properties.
function MotorFrequency_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MotorFrequency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function aorta_Callback(hObject, eventdata, handles)
% hObject    handle to aorta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of aorta as text
%        str2double(get(hObject,'String')) returns contents of aorta as a double


% --- Executes during object creation, after setting all properties.
function aorta_CreateFcn(hObject, eventdata, handles)
% hObject    handle to aorta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function length_Callback(hObject, eventdata, handles)
% hObject    handle to length (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of length as text
%        str2double(get(hObject,'String')) returns contents of length as a double


% --- Executes during object creation, after setting all properties.
function length_CreateFcn(hObject, eventdata, handles)
% hObject    handle to length (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function valve_Callback(hObject, eventdata, handles)
% hObject    handle to valve (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of valve as text
%        str2double(get(hObject,'String')) returns contents of valve as a double


% --- Executes during object creation, after setting all properties.
function valve_CreateFcn(hObject, eventdata, handles)
% hObject    handle to valve (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called




function waveform_Callback(hObject, eventdata, handles)
% hObject    handle to waveform (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of waveform as text
%        str2double(get(hObject,'String')) returns contents of waveform as a double


% --- Executes during object creation, after setting all properties.
function waveform_CreateFcn(hObject, eventdata, handles)
% hObject    handle to waveform (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function backpressure_Callback(hObject, eventdata, handles)
% hObject    handle to backpressure (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of backpressure as text
%        str2double(get(hObject,'String')) returns contents of backpressure as a double


% --- Executes during object creation, after setting all properties.
function backpressure_CreateFcn(hObject, eventdata, handles)
% hObject    handle to backpressure (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Executes during object creation, after setting all properties.
function chamberpressure_CreateFcn(hObject, eventdata, handles)
% hObject    handle to length (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



% --- Executes during object creation, after setting all properties.
function FileSaveTextDisplay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to FileSaveTextDisplay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

%filesavetext=[a_aorta '_' l_length '_' v_valve '_' w_waveform '_' b_backpressure '_'
c_chamberpressure '_datecode.txt'];
% filesavetext=[...
%     'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
%     'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
%     'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
%     'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
%     's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
%     'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
%     'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
%     'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
%     'datecode.txt'];
% set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes when selected object is changed in aorta.
function aorta_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in aorta
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% a_aorta=['a-' get(get(hObject,'SelectedObject'), 'String')];
if get(handles.a_other,'Value')==0
    set(handles.a_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);

% --- Executes when selected object is changed in length.
```

```
function length_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in length
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% l_length=['l-' get(get(hObject,'SelectedObject'), 'String')];
if get(handles.l_other,'Value')==0
    set(handles.l_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);

% --- Executes when selected object is changed in valve.
function valve_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in valve
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% v_valve=['v-' get(get(hObject,'SelectedObject'), 'String')];
if get(handles.v_other,'Value')==0
    set(handles.v_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);

% --- Executes when selected object is changed in waveform.
function waveform_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in waveform
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% w_waveform=['w-' get(get(hObject,'SelectedObject'), 'String')];

if get(handles.w_other,'Value')==0
    set(handles.w_other_edit,'Enable','off');
end
```

```
filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);



% --- Executes when selected object is changed in backpressure.
function backpressure_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in backpressure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% b_backpressure=['b-' get(get(hObject,'SelectedObject'), 'String')];
if get(handles.b_other,'Value')==0
    set(handles.b_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);

% --- Executes when selected object is changed in chamberpressure.
function chamberpressure_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in chamberpressure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% c_chamberpressure=['c-' get(get(hObject,'SelectedObject'), 'String')];
if get(handles.c_other,'Value')==0
    set(handles.c_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


function c_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to c_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of c_other_edit as text
%        str2double(get(hObject,'String')) returns contents of c_other_edit as a double


% --- Executes during object creation, after setting all properties.
function c_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on c_other_edit and none of its controls.
function c_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to c_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%       Key: name of the key that was pressed, in lower case
%       Character: character interpretation of the key(s) that was pressed
%       Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
if isequal(eventdata.Key,'return')
    set(handles.c_other, 'Tooltip',get(handles.c_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in c_other.
function c_other_Callback(hObject, eventdata, handles)
% hObject    handle to c_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of c_other

set(handles.c_other_edit,'Enable','on');



function b_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to b_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of b_other_edit as text
%        str2double(get(hObject,'String')) returns contents of b_other_edit as a double


% --- Executes during object creation, after setting all properties.
function b_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to b_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function w_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to w_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of w_other_edit as text
%        str2double(get(hObject,'String')) returns contents of w_other_edit as a double


% --- Executes during object creation, after setting all properties.
function w_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to w_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on b_other_edit and none of its controls.
function b_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to b_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%      Key: name of the key that was pressed, in lower case
%      Character: character interpretation of the key(s) that was pressed
%      Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
if isequal(eventdata.Key,'return')
    set(handles.b_other, 'Tooltip',get(handles.b_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in b_other.
function b_other_Callback(hObject, eventdata, handles)
% hObject    handle to b_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of b_other
set(handles.b_other_edit,'Enable','on');


% --- Executes on key press with focus on w_other_edit and none of its controls.
```

```
function w_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to w_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%       Key: name of the key that was pressed, in lower case
%       Character: character interpretation of the key(s) that was pressed
%       Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
if isequal(eventdata.Key,'return')
    set(handles.w_other, 'Tooltip',get(handles.w_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in w_other.
function w_other_Callback(hObject, eventdata, handles)
% hObject    handle to w_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of w_other
set(handles.w_other_edit,'Enable','on');




function l_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to l_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of l_other_edit as text
%        str2double(get(hObject,'String')) returns contents of l_other_edit as a double


% --- Executes during object creation, after setting all properties.
```

```
function l_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to l_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function v_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to v_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of v_other_edit as text
%        str2double(get(hObject,'String')) returns contents of v_other_edit as a double


% --- Executes during object creation, after setting all properties.
function v_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function a_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to a_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a_other_edit as text
%        str2double(get(hObject,'String')) returns contents of a_other_edit as a double


% --- Executes during object creation, after setting all properties.
function a_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on a_other_edit and none of its controls.
function a_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to a_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%	Key: name of the key that was pressed, in lower case
%	Character: character interpretation of the key(s) that was pressed
%	Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
if isequal(eventdata.Key,'return')
    set(handles.a_other, 'Tooltip',get(handles.a_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on key press with focus on l_other_edit and none of its controls.
function l_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to l_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%	Key: name of the key that was pressed, in lower case
%	Character: character interpretation of the key(s) that was pressed
%	Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
if isequal(eventdata.Key,'return')
    set(handles.l_other, 'Tooltip',get(handles.l_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on key press with focus on v_other_edit and none of its controls.
function v_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to v_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%       Key: name of the key that was pressed, in lower case
%       Character: character interpretation of the key(s) that was pressed
%       Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)

if isequal(eventdata.Key,'return')
    set(handles.v_other, 'Tooltip',get(handles.v_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in a_other.
```

```
function a_other_Callback(hObject, eventdata, handles)
% hObject    handle to a_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of a_other
set(handles.a_other_edit,'Enable','on');


% --- Executes on button press in l_other.
function l_other_Callback(hObject, eventdata, handles)
% hObject    handle to l_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of l_other
set(handles.l_other_edit,'Enable','on');


% --- Executes on button press in v_other.
function v_other_Callback(hObject, eventdata, handles)
% hObject    handle to v_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of v_other
set(handles.v_other_edit,'Enable','on');


% --- Executes during object creation, after setting all properties.
function LiveModeTooltip_CreateFcn(hObject, eventdata, handles)
% hObject    handle to LiveModeTooltip (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in SelectAll.
function SelectAll_Callback(hObject, eventdata, handles)
% hObject    handle to SelectAll (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.ai0Button,'Value',1);
set(handles.ai1Button,'Value',1);
set(handles.ai2Button,'Value',1);
set(handles.ai3Button,'Value',1);
set(handles.ai4Button,'Value',1);
set(handles.ai5Button,'Value',1);
set(handles.ai6Button,'Value',1);
set(handles.ai7Button,'Value',1);


% --- Executes on button press in SelectNone.
function SelectNone_Callback(hObject, eventdata, handles)
% hObject    handle to SelectNone (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.ai0Button,'Value',0);
set(handles.ai1Button,'Value',0);
set(handles.ai2Button,'Value',0);
set(handles.ai3Button,'Value',0);
set(handles.ai4Button,'Value',0);
set(handles.ai5Button,'Value',0);
set(handles.ai6Button,'Value',0);
set(handles.ai7Button,'Value',0);



function s_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to s_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of s_other_edit as text
%        str2double(get(hObject,'String')) returns contents of s_other_edit as a double


% --- Executes during object creation, after setting all properties.
function s_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to s_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function f_other_edit_Callback(hObject, eventdata, handles)
% hObject    handle to f_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of f_other_edit as text
%        str2double(get(hObject,'String')) returns contents of f_other_edit as a double


% --- Executes during object creation, after setting all properties.
function f_other_edit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f_other_edit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

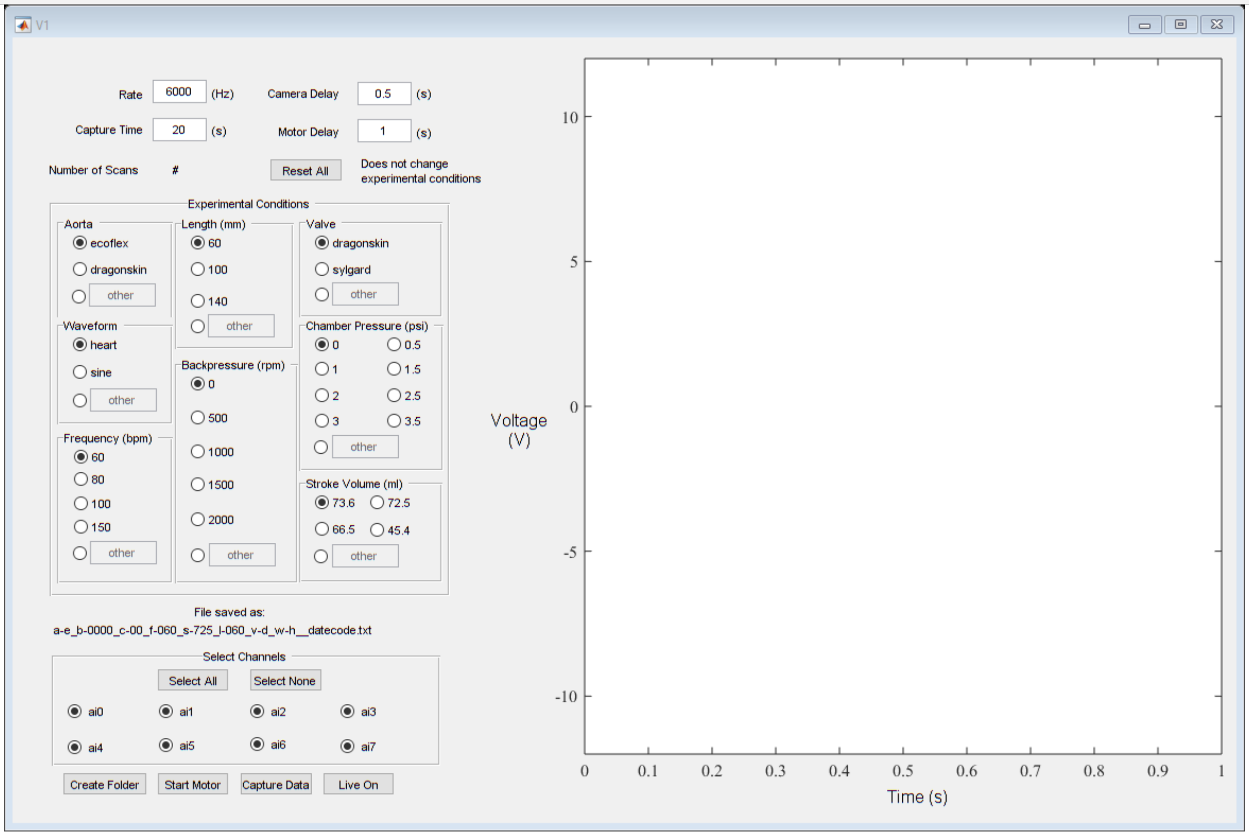
```
    set(hObject,'BackgroundColor','white');
end


% --- Executes on key press with focus on f_other_edit and none of its controls.
function f_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to f_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%       Key: name of the key that was pressed, in lower case
%       Character: character interpretation of the key(s) that was pressed
%       Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)

if isequal(eventdata.Key,'return')
    set(handles.f_other, 'Tooltip',get(handles.f_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on key press with focus on s_other_edit and none of its controls.
function s_other_edit_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to s_other_edit (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTROL)
%       Key: name of the key that was pressed, in lower case
%       Character: character interpretation of the key(s) that was pressed
%       Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)
if isequal(eventdata.Key,'return')
    set(handles.s_other, 'Tooltip',get(handles.s_other_edit,'String'));
    import java.awt.Robot;
    import java.awt.event.KeyEvent;
    robot=Robot;
    robot.keyPress(KeyEvent.VK_ENTER);
    pause on
    pause(0.01)
    robot.keyRelease(KeyEvent.VK_ENTER);
    pause(0.01)
    pause off
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in f_other.
function f_other_Callback(hObject, eventdata, handles)
% hObject    handle to f_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of f_other
set(handles.f_other_edit,'Enable','on');


% --- Executes on button press in s_other.
function s_other_Callback(hObject, eventdata, handles)
% hObject    handle to s_other (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of s_other
set(handles.s_other_edit,'Enable','on');


% --- Executes when selected object is changed in frequency.
function frequency_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in frequency
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.f_other,'Value')==0
    set(handles.f_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes when selected object is changed in strokevolume.
function strokevolume_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in strokevolume
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if get(handles.s_other,'Value')==0
    set(handles.s_other_edit,'Enable','off');
end

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);



function MotorDelay_Callback(hObject, eventdata, handles)
% hObject    handle to MotorDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of MotorDelay as text
%        str2double(get(hObject,'String')) returns contents of MotorDelay as a double


% --- Executes during object creation, after setting all properties.
function MotorDelay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to MotorDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function CameraDelay_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to CameraDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of CameraDelay as text
%        str2double(get(hObject,'String')) returns contents of CameraDelay as a double


% --- Executes during object creation, after setting all properties.
function CameraDelay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to CameraDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in StartMotor.
function StartMotor_Callback(hObject, eventdata, handles)
% hObject    handle to StartMotor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
!start MoveMyTeknicMatlab.lnk


% --- Executes on button press in CreateFolders.
function CreateFolders_Callback(hObject, eventdata, handles)
% hObject    handle to CreateFolders (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
filesavetext=get(handles.FileSaveTextDisplay,'String');
folderdate=datestr(now, 'yyyy-mm-dd');
mkdir(['datacollected\' folderdate '\data']);
mkdir(['datacollected\' folderdate '\images\' filesavetext(1:end-12)]);

a_aorta={'e' 'd'};
b_backpressure={'0000' '0500' '1000' '1500' '2000'};
c_chamberpressure={'00' '05' '10' '15' '20' '25' '30' '35'};
f_frequency={'060_s-736' '080_s-725' '100_s-665' '150_s-454'};
%s_strokevolume={'070' '050'};
l_length={'060' '100' '140'};
v_valve={'d' 's'};
w_waveform={'h'};%, 's'};

for a=1:length(a_aorta)
    for b=1:length(b_backpressure)
        for c=1:length(c_chamberpressure)
            for f=1:length(f_frequency)
                for l=1:length(l_length)
                    %for s=1:length(s_strokevolume)
```

```
                        for v=1:length(v_valve)
                            for w=1:length(w_waveform)
                                filesavetext=[...
                                    'a-' a_aorta{a} '_' ...
                                    'b-' b_backpressure{b} '_' ...
                                    'c-' c_chamberpressure{c} '_' ...
                                    'f-' f_frequency{f} '_' ...%'s-' s_strokevolume{s} '_' ...
                                    'l-' l_length{l} '_' ...
                                    'v-' v_valve{v} '_' ...
                                    'w-' w_waveform{w} '_'];
                                mkdir(['datacollected\' folderdate '\images\' filesavetext]);
                            end
                        end
                    %end
                end
            end
        end
    end
end



% --- Executes on button press in ResetAll.
function ResetAll_Callback(hObject, eventdata, handles)
% hObject    handle to ResetAll (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Re-enable all disabled buttons
set(handles.CaptureDataButton, 'Enable','on');
set(handles.LiveOnButton,'Enable','on');

%Reset user inputs
set(handles.RateInput,'String','6000');
set(handles.CaptureTimeInput,'String','20');
set(handles.CameraDelay,'String','0.5');
set(handles.MotorDelay,'String','1');

%Reset experimental conditions

% @aorta_SelectionChangedFcn;
% @backpressure_SelectionChangedFcn;
% @chamberpressure_SelectionChangedFcn;
% @frequency_SelectionChangedFcn;
% @length_SelectionChangedFcn;
% @strokevolume_SelectionChangedFcn;
% @valve_SelectionChangedFcn;
% @waveform_SelectionChangedFcn;

%Reselect all channels
set(handles.ai0Button,'Value',1);
set(handles.ai1Button,'Value',1);
set(handles.ai2Button,'Value',1);
set(handles.ai3Button,'Value',1);
set(handles.ai4Button,'Value',1);
set(handles.ai5Button,'Value',1);
set(handles.ai6Button,'Value',1);
set(handles.ai7Button,'Value',1);

%Redisplay filesavetext
filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in CreateChecklist.
function CreateChecklist_Callback(hObject, eventdata, handles)
% hObject    handle to CreateChecklist (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a_aorta={'e' 'd'};
b_backpressure={'0000' '0500' '1000' '1500' '2000'};
c_chamberpressure={'00' '05' '10' '15' '20' '25' '30' '35'};
f_frequency={'060_s-736' '080_s-725' '100_s-665' '150_s-454'};
%s_strokevolume={'070' '050'};
l_length={'060' '100' '140'};
v_valve={'d' 's'};
w_waveform={'h'};%, 's'};

folderdate=datestr(now, 'yyyy-mm-dd');
fileID = fopen(['datacollected\' 'checklist_' folderdate '.txt'],'wt');

for a=1:length(a_aorta)
    for b=1:length(b_backpressure)
        for c=1:length(c_chamberpressure)
            for f=1:length(f_frequency)
                for l=1:length(l_length)
                    %for s=1:length(s_strokevolume)
                        for v=1:length(v_valve)
                            for w=1:length(w_waveform)
                                filesavetext=[...
                                    'a-' a_aorta{a} '_' ...
                                    'b-' b_backpressure{b} '_' ...
                                    'c-' c_chamberpressure{c} '_' ...
                                    'f-' f_frequency{f} '_' ...%'s-' s_strokevolume{s} '_' ...
                                    'l-' l_length{l} '_' ...
                                    'v-' v_valve{v} '_' ...
                                    'w-' w_waveform{w} '_'];
                                fprintf(fileID, [filesavetext '\n']);
                            end
                        end
                    %end
                end
            end
        end
    end
end
fclose all;


% --- Executes on button press in f_060.
function f_060_Callback(hObject, eventdata, handles)
% hObject    handle to f_060 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of f_060
set(handles.s_736, 'Value',1);

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);



% --- Executes on button press in f_080.
function f_080_Callback(hObject, eventdata, handles)
% hObject    handle to f_080 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of f_080
set(handles.s_725, 'Value',1);

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);
```

```
% --- Executes on button press in f_100.
function f_100_Callback(hObject, eventdata, handles)
% hObject    handle to f_100 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of f_100
set(handles.s_665, 'Value',1);

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in f_150.
function f_150_Callback(hObject, eventdata, handles)
% hObject    handle to f_150 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of f_150
set(handles.s_454, 'Value',1);

filesavetext=[...
    'a-' get(get(handles.aorta,'SelectedObject'), 'Tooltip') '_' ...
    'b-' get(get(handles.backpressure,'SelectedObject'), 'Tooltip') '_' ...
    'c-' get(get(handles.chamberpressure,'SelectedObject'), 'Tooltip') '_' ...
    'f-' get(get(handles.frequency,'SelectedObject'), 'Tooltip') '_' ...
    's-' get(get(handles.strokevolume,'SelectedObject'), 'Tooltip') '_' ...
    'l-' get(get(handles.length,'SelectedObject'), 'Tooltip') '_' ...
    'v-' get(get(handles.valve,'SelectedObject'), 'Tooltip') '_' ...
    'w-' get(get(handles.waveform,'SelectedObject'), 'Tooltip') '_' ...
    'datecode.txt'];
set(handles.FileSaveTextDisplay, 'String', filesavetext);


% --- Executes on button press in CreateFolder.
function CreateFolder_Callback(hObject, eventdata, handles)
% hObject    handle to CreateFolder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
str = datestr(now, 'yyyy-mm-dd-HHMMSS');
filesavetext=get(handles.FileSaveTextDisplay,'String');

folderdate=datestr(now, 'yyyy-mm-dd');
```

```
mkdir(['datacollected\' folderdate '\data']);
mkdir(['datacollected\' folderdate '\images\' filesavetext(1:end-12)]);
```



Rate 6000 (Hz) Camera Delay 0.5 (s)

Capture Time 20 (s) Motor Delay 1 (s)

Number of Scans #

Does not change experimental conditions

Experimental Conditions

Aorta: [✓ ecoflex] dragonskin other

Length (mm): [✓ 60] 100 140 other

Valve: [✓ dragonskin] sylgard other

Waveform: [✓ heart] sine other

Backpressure (rpm): [✓ 0] 500 1000 1500 2000 other

Chamber Pressure (psi): [✓ 0] 0.5 1 1.5 2 2.5 3 3.5 other

Frequency (bpm): [✓ 60] 80 100 150 other

Stroke Volume (ml): [✓ 73.6] 72.5 66.5 45.4 other

File saved as:
a-e_b-0000_c-00_f-060_s-725_l-060_v-d_w-h__datecode.txt

Select Channels: [✓ ai0] [✓ ai1] [✓ ai2] [✓ ai3] [✓ ai4] [✓ ai5] [✓ ai6] [✓ ai7]

Published with MATLAB® R2019a

Appendix D. MATLAB Code for File Sorting

1.1. Introduction and Motivation........1

1.2. Ex Vivo Heart Perfusion (EVHP)........3

1.3. Experimental Aorta Phantoms........5

1.4. The Cardiac Cycle........7

1.5. Pulsatile Flow........10

1.6. Experimental Pulsatile Systems........12

1.7. Newtonian vs Non-Newtonian Fluid........15

1.8. Research Objectives........17

1.9. Content and Outline of Thesis........18

2.1. Introduction........19

2.2. The Pulsatile Pump........22

2.3. Pump Drive Mechanism........28

2.4. Compliant Components........30

2.5. Pressure-Chamber........44

2.6. Reservoir and Backpressure Pump........54

2.7. Data Acquisition........55

2.8. Limitations........59

2.9. Summary........63

3.1. Introduction........64

3.2. Processing of Imaging Data 64

3.3. Processing of Pressure Data 70

3.4. Processing of Pump Stroke Distance 80

3.5. Volumetric Flow Rate, Q(t) 82

3.6. Pump Energy Calculation....89

3.7. Limitations 91

3.8. Summary 92

4.1. Introduction 93

4.2. Impact of Experimental Parameters on Pump Energy 94

4.3. Impact of Aorta Properties on Aorta Elastic Response....115

4.4. Impact of Aorta Elastic Response on Pump Energy 128

4.5. Limitations 134

4.6. Summary 136

5.1. Conclusion....140

5.2. Limitations and Future Work 142

```
% Filtering pressure data gathered from the DAQ
```

Initialization

```
%load data.mat

fclose all
close all
clearvars -except data
clc
```

```
ans =

     0
```

variables and info

```
samplerate = 6000; %DAQ sample rate at 6000 Hz
order = 6;
framerate = 200; %camera trigger frame rate
for n=1:length(data.namelistgood)

    disp(n)

    name=data.namelistgood{n,1};
%     name=strrep(name, '-', 'x');

    %determining cut off frequency from the file name motor frequency
    cutofffreq= 20;

    [~,motorcyc]=findpeaks(data.experiments.(name).data.ai2(42000:end),'MinPeakDistance',900,
'MinPeakProminence',1,'NPeaks',4);

    motorcyc=motorcyc+42000;


    for r=4:8

        %DFT
        y=fft(data.experiments.(name).pressure.raw.(data.pressurelocations{r,2}));
        f=6000*(0:(72000/2))/72000;
        L=72000;
        p2=abs(y/L);
        p1=p2(1:L/2+1);
        p1(2:end-1)=2*p1(2:end-1);

        %applying butterworth filter
        data.experiments.(name).pressure.filtered.(data.pressurelocations{r,2})=...
            LPButterworthFilt(order, samplerate, cutofffreq,...
            data.experiments.(name).pressure.raw.(data.pressurelocations{r,2}));

    end

end

% save('data.mat','data', '-v7.3')
```

Published with MATLAB® R2019a

Appendix E. MATLAB Code for Image Processing

```
%this is a code to process the images, with fixed distension
close all
reset(groot)
clearvars -except data
clc

if exist('data','var')==0
    load data.mat
end

r=1;
R=length(fieldnames(data.experiments));
e=1;
```

processing

```
while r<R%:length(fieldnames(data.experiments))
% for r=1
    disp(r)
        try
    name=data.namelist{r,1}; %defines which case within the structure

    name=strrep(name, '-', 'x');

    %organizing folders and images

    images=dir([data.experiments.(name).imagepath '\*.tiff']);
    imagecount=length(images);

    %Calculate ref image tube width
    Aref=imread([data.experiments.(name).imagepath images(5).name]);
    bwref=Aref<175; %binarizes the image

    %     figure(1)
    %     imshow(bwref)

    s = regionprops(bwref); %finds connected regions in ref image

    %     for i=1:size(s)
    %     rectangle('Position',s(i).BoundingBox,'EdgeColor','r')
    %     end
    %
    %     axis on

    %finding bounding box imensions for xmax and ymax. bbox turns into [xmin ymin xmax ymax]
    bbox=floor(vertcat(s.BoundingBox));
    bbox(:,3)=bbox(:,1)+bbox(:,3);
    bbox(:,4)=bbox(:,2)+bbox(:,4);
```

```
%%code to find cropping edges

%top edge
cropt=max(bbox(bbox(:,4)<40,4));

if isempty(cropt)
    cropt=0;
end

%bot edge
cropb=min(bbox(bbox(:,2)>560,2));

if isempty(cropb)
    cropb=600;
end

%left edge
cropl=max(bbox(bbox(:,3)<100 & bbox(:,2)>cropt & bbox(:,4)<cropb,3));

if isempty(cropl)
    cropl=0;
end

%right edge
cropr=min(bbox(bbox(:,1)>700 & bbox(:,2)>cropt & bbox(:,4)<cropb,1));

if isempty(cropr)
    cropr=800;
end

cropwidth=cropr-cropl;
cropheight=cropb-cropt;
cropedge=[cropl+1,cropt+1,cropwidth, cropheight]; %crop bounding box

%     Aref=imcrop(Aref,cropedge);
%     figure(2)
%     imshow(Aref)
%     axis on

%looping through images to calculate maximum and minimum lengths
for rr=1:imagecount

    A=imread([data.experiments.(name).imagepath images(rr).name]);
    A=imcrop(A, cropedge);
    A(A<150)=A(A<150)*0.25; %reduce intensity values of all non white space pixels

    %             imshow(A)

    for c=1:size(A,2)
        I(:,c) = smooth(double(A(:,c)),9);

        % generate plot of dI vs h at selected row and frame
        dI(:,c) = I(2:end,c) - I(1:end-1,c);
        [e,locs_dI(1,c)] = findpeaks(-
dI(1:round(cropheight/2),c),'NPeaks',1,'SortStr','descend'); %finds top edge
            [f,locs_dI(2,c)] =
findpeaks(dI(round(cropheight/2):end,c),'NPeaks',1,'SortStr','descend'); %finds bottom edge

            locs_dI(2,c)=locs_dI(2,c)+round(cropheight/2);
            % tube intensity plot

            %                                 set(0,'defaultTextFontName','Times New Roman',...
            %                     'defaultAxesFontName','Times New Roman',...
            %                     'defaultAxesFontSize',12,...
            %                     'defaultAxesBox','on',...
            %                     'defaultFigurePosition',[0 0 16.3 10],...
            %                     'defaultFigureUnits','centimeters');
            %
            %                     figure()
            %                     plot(-dI(1:round(cropheight/2),c), '-k');
            %                     hold on
            %
a=length(dI(round(cropheight/2):end,c)):1:length(dI(round(cropheight/2):end,c))*2-1;
            %
            %                     plot(locs_dI2(1,c),g(1), 'rs');
            %                     plot(locs_dI2(2,c),g(2), 'rs');
            %                     plot(locs_dI2(3,c),h(1), 'rs');
            %                     plot(locs_dI2(4,c),h(2), 'rs');
            %
            %                     plot(a,dI(round(cropheight/2):end,c), '-k');
            %
            %
            %
            %                     hold off
            %
            %                     axis on
            %                     xlabel('y, Tube Width [px]');
            %                     ylabel('dI, Peak Intensity [px]');
            %
            %                     saveas(gcf, char(strcat('D:\compliant
tube\code\processing\plots\imageprocessing\peakintensity_',...
            %                         'b-',data.namelistgood{list(n),1}(7:10),'_',...
            %                         'c-',data.namelistgood{list(n),1}(14:16),'_',...
            %                         'f-',data.namelistgood{list(n),1}(20:22),...
            %                         '.png')));

        end

        data.experiments.(name).distend(rr,:)=locs_dI(2,:)-locs_dI(1,:); %calculates the width of
the tube vs length for each image

    end

    r=r+1;
    disp(r)

            save('data.mat','data', '-v7.3')
            pause(1)

    clearvars -except data R r e

        catch
                    save('data.mat','data', '-v7.3')
                    pause(1)
                    errors(e)=r; %saves the case number for the case that is broken
                    e=e+1
                    r=r+1
                    clearvars -except data R r e
            continue;
        end

end
```

Published with MATLAB® R2019a

Appendix F. MATLAB Code for Aorta Diameter Distension

```
close all
reset(groot)
clearvars -except data
clc

if exist('data','var')==0
    load data.mat
end

% %half page figure sizes
set(0,'defaultTextFontName','Garamond',...
    'defaultAxesFontName','Garamond',...
    'defaultAxesFontSize',12,...
    'defaultAxesBox','on',...
    'defaultFigurePosition',[0 0 16.3 10],...
    'defaultFigureUnits','centimeters');
%file names for determining experimental case
bpm=20:22;
chamber_pressure=14:16;
back_pressure=7:10;
tube_length=32:34;
stroke_volume=26:28;
material=3;

youngs=[82.7 151.7]; %young's modulus in kPa for 1: ecoflex and 2: dragonskin
D=19.05; %tube diameter in mm
h=4; %tube wall thickness in mm
mmhg2kpa=0.133322; %1 mmhg=0.133322 kPa

order = 6;
fc=50; %cutoff frequency
fs=200; %camera sample rate at 200 hz

for n=1:length(data.namelistgood)

    name=data.namelistgood{n,1};

    camtime=data.experiments.(name).time.camera;

    for cyc=1:8
        cycstart=data.experiments.(name).time.experiment(data.experiments.(name).cycle(cyc));
        [val,data.experiments.(name).camcycle(cyc)]=min(abs(camtime-cycstart));
    end


    for rr=1:size(data.experiments.(name).tube.distend,2)

        filtereddist=LPButterworthFilt(order, fs, fc,...
            data.experiments.(name).tube.distend(:,rr));
        [max,cyclesmax]=findpeaks(filtereddist(data.experiments.(name).camcycle(1):end),...
            'MinPeakProminence',0.2,'NPeaks',8);

%         findpeaks(filtereddist(data.experiments.(name).camcycle(1):end),...
%             'MinPeakProminence',0.2,'NPeaks',8);

        %         findpeaks(filtereddist(data.experiments.(name).camcycle(1):end),...
        %             'MinPeakProminence',0.1,'NPeaks',8);

        %         plot(max,'o')
        %         ylim([0 inf])

        tubemax(rr)=mean(max);


    end

            data.experiments.(name).tube.maxpercent=tubemax;

    tubemaxmax=tubemax(300:500);
    TF=isoutlier(tubemaxmax,'median');

     data.experiments.(name).tube.maxpercentmean=mean(tubemaxmax(~TF));
%     plot(tubemaxmax(~TF))

    %     data.experiments.(name).tube.maxpercent=LPButterworthFilt(order, fs, fc,...
    %             tubemax);

end
```

Published with MATLAB® R2019a

```
close all
reset(groot)
clearvars -except data
clc

if exist('data','var')==0
    load data.mat
end

% %half page figure sizes
set(0,'defaultTextFontName','Garamond',...
    'defaultAxesFontName','Garamond',...
    'defaultAxesFontSize',12,...
    'defaultAxesBox','on',...
    'defaultFigurePosition',[0 0 16.3 10],...
    'defaultFigureUnits','centimeters');
%file names for determining experimental case
bpm=20:22;
chamber_pressure=14:16;
back_pressure=7:10;
```

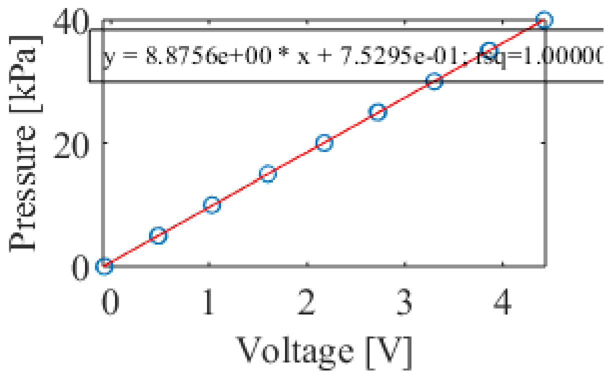
```
tube_length=32:34;
stroke_volume=26:28;
material=3;

youngs=[82.7 151.7]; %young's modulus in kPa for 1: ecoflex and 2: dragonskin
D=19.05; %tube diameter in mm
h=4; %tube wall thickness in mm
mmhg2kpa=0.133322; %1 mmhg=0.133322 kPa

order = 6;
fc=50; %cutoff frequency
fs=200; %camera sample rate at 200 hz

for n=1:length(data.namelistgood)

    name=data.namelistgood{n,1};

    camtime=data.experiments.(name).time.camera;
%
%     for cyc=1:8
%         cycstart=data.experiments.(name).time.experiment(data.experiments.(name).cycle(cyc));
%         [val,data.experiments.(name).camcycle(cyc)]=min(abs(camtime-cycstart));
%     end


    for rr=1:size(data.experiments.(name).tube.distend,2)

                filtereddist=LPButterworthFilt(order, fs, fc,...
            data.experiments.(name).tube.distend(:,rr));

%         filtereddist=LPButterworthFilt(order, fs, fc,...
%             data.experiments.(name).distend(:,rr));


        [min,cyclesmin]=findpeaks(-filtereddist(data.experiments.(name).camcycle(1):end),...
            'MinPeakProminence',0.2,'NPeaks',8);

% findpeaks(-filtereddist(data.experiments.(name).camcycle(1):end),...
%             'MinPeakProminence',0.2,'NPeaks',8);

        %         plot(max,'o')
        %         ylim([0 inf])

        tubemin(rr)=mean(min);


    end

            data.experiments.(name).tube.minpercent=tubemin;
% data.experiments.(name).tube.minmean=tubemin;

    tubeminmin=tubemin(300:500);
    TF=isoutlier(tubeminmin,'median');
    data.experiments.(name).tube.minpercentmean=mean(tubeminmin(~TF));
%     plot(tubemaxmax(~TF))

    %     data.experiments.(name).tube.maxpercent=LPButterworthFilt(order, fs, fc,...
    %             tubemax);

end
```

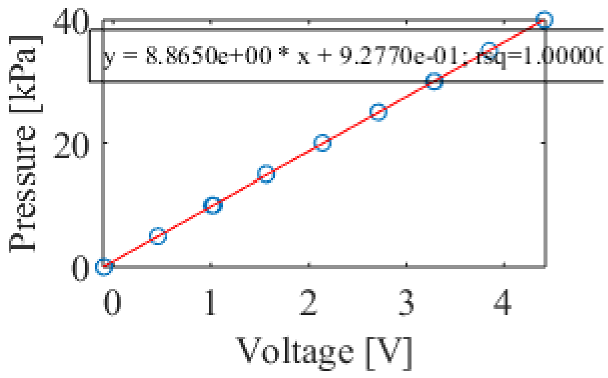
Published with MATLAB® R2019a

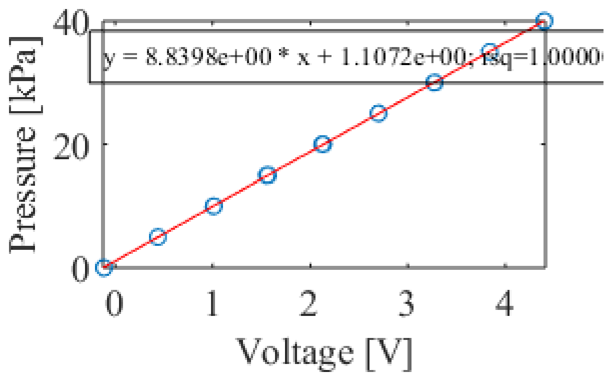
Appendix G.   MATLAB Code for Pressure Calibration

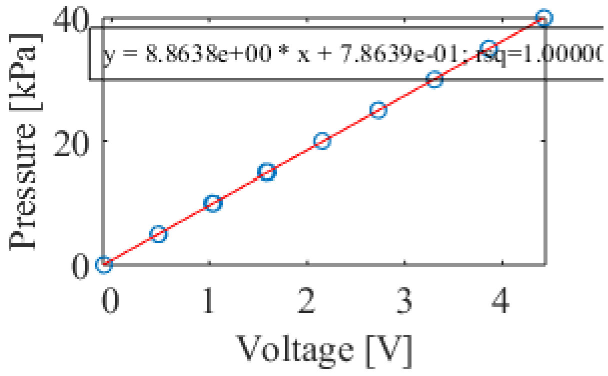
```
function [calarray,errarray]=transducercal(calLoc,aiarray,inclusions)
% Takes in all 8-channel DAQ inputs in same folder, selects channels based
% on aiarray, performs linear and quadratic calibrations. Only uses the
% files indexed by inclusions. requires naming scheme:
% calibration_##KPa_yyyymmdd_etc.log where ## represents input pressure in
% KPa
% accurate to 1 decimal places.
% Inputs:   calLoc      - Location of calibration data
%           aiarray     - Channel inputs to calibrate
%           inclusions  - which data files to include
if nargin<3
    close all
    calLoc='D:\compliant tube\calibration\2019-02-22\';
    aiarray=[3 4 5 6 7];
%     inclusions=[1:9];
end

% find files
files=dir([calLoc '*.log']);
% files=files(inclusions);

% create arrays
calarray=zeros(length(files),1+length(aiarray));
errarray=zeros(length(files),1+length(aiarray));
assembly=[];
ppar1=zeros(2,2);
ppar2=zeros(2,3);

% loop through files
for i=1:length(files)
    % load a file, reading from with a row offset of 7 and a column offset
    % of 0
    data=dlmread([calLoc files(i).name],'\t',7,0);

    %     % get the water column height
    %     calarray(i,1)=str2double(files(i).name(5))+str2double(['0.' files(i).name(7:9)]);

    % get the pressure
    calarray(i,1)=str2double(files(i).name(15:16));


    % fill in the error term for the pressure
    errarray(i,1)=0.1;

    % fill in the arrays for each desired input column
    thisassembly=ones(length(data),1)*calarray(i,1);

    for j=1:length(aiarray)
        thisassembly=[thisassembly data(:,aiarray(j)+2)];
        calarray(i,j+1)=mean(data(:,aiarray(j)+2));
        errarray(i,j+1)=3*std(data(:,aiarray(j)+2));
    end

    assembly=[assembly; thisassembly];

end
% do the polynomial fitting
for j=1:length(aiarray)
    ppar1(j,:)=polyfit(assembly(:,1),assembly(:,j+1),1);
    yfit1(j,:)=polyval(ppar1(j,:),assembly(:,1));
    yr1(:,j)=assembly(:,j+1)-yfit1(j);
    ssr1(j)=sum(yr1(j).^2);
    sst1(j)=length(assembly(:,j+1)-1)*var(assembly(:,j+1));
    rsq1(j)=1-ssr1(j)/sst1(j);
    % stuff below is for 2nd order polyfits
    %     ppar2(j,:)=polyfit(assembly(:,1),assembly(:,j+1),2);
    %     yfit2(j,:)=polyval(ppar2(j,:),assembly(:,1));
    %     yr2(:,j)=assembly(:,j+1)-yfit2(j);
    %     ssr2(j)=sum(yr2(j).^2);
    %     sst2(j)=length(assembly(:,j+1)-1)*var(assembly(:,j+1));
    %     rsq2(j)=1-ssr2(j)/sst2(j)*length(assembly(:,j+1)-1)/(length(assembly-length(ppar2)));
end
% rdif=rsq2-rsq1;

% create and save calibration plots
for i=1:length(aiarray)
    figure
    scatter(assembly(:,i+1),assembly(:,1))
    xlabel('Voltage [V]')
    ylabel('Pressure [kPa]')
    set(findall(gcf,'-property','FontSize'),'FontSize',14)
    set(findall(gcf,'-property','FontName'),'FontName','Times New Roman')
    hold on
    dim = [.15 .6 .3 .3];
    str = sprintf('y = %1.4d * x + %1.4d; rsq=%1.6d',1/ppar1(i,1),-
ppar1(i,2)/ppar1(i,1),rsq1(i));
    annotation('textbox',dim,'String',str,'FitBoxToText','on');
    plot(yfit1(i,:)',assembly(:,1),'-r')
    saveas(gcf,[calLoc 'TransducerCal_ai' sprintf('%1i',aiarray(i)) '_' files(1).name(19:28)
'.png'])
    hold off
end

%save calibration variables
% save([calLoc 'TransducerCal_' files(1).name(19:28) '.mat'],'ppar1','rsq1')
```

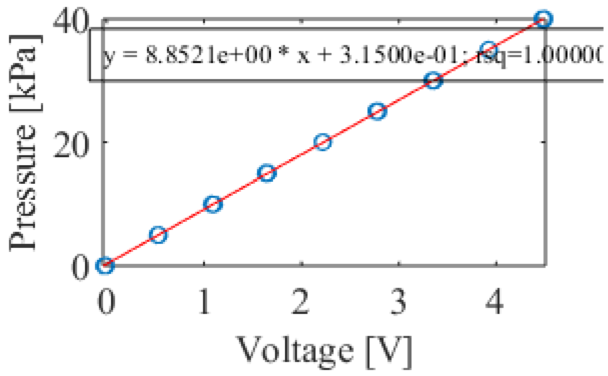
```
ans =

         0   -0.0704   -0.0919   -0.1121   -0.0760   -0.0224
    5.0000    0.4810    0.4591    0.4401    0.4754    0.5291
   10.0000    1.0308    1.0242    1.0063    1.0400    1.0945
   15.0000    1.5964    1.5761    1.5601    1.5921    1.6473
   20.0000    2.1691    2.1418    2.1276    2.1582    2.2136
   25.0000    2.7218    2.7077    2.6950    2.7238    2.7811
   30.0000    3.2958    3.2835    3.2729    3.2997    3.3576
   35.0000    3.8633    3.8489    3.8397    3.8652    3.9238
   40.0000    4.4291    4.4135    4.4056    4.4304    4.4891
```











Published with MATLAB® R2019a

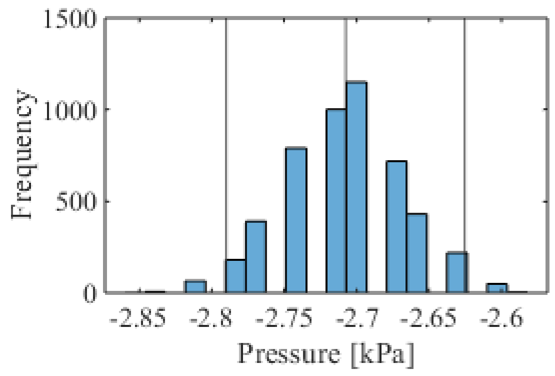
Appendix H. MATLAB code for Backpressure Unit Conversion

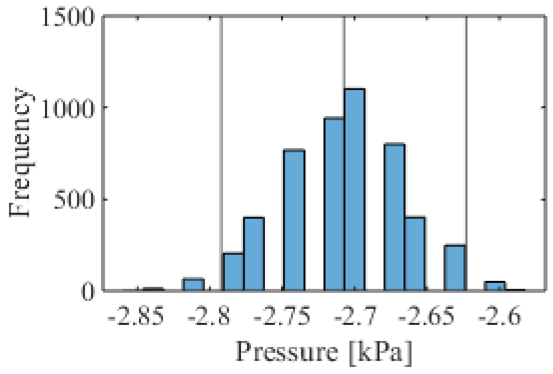
```
%This code does the calibration for the system backpressure (baseline pressure)
close all
reset(groot)
clearvars -except data
clc

if exist('data','var')==0
    load data.mat
end

% %half page figure sizes
set(0,'defaultTextFontName','Times New Roman',...
    'defaultAxesFontName','Times New Roman',...
    'defaultAxesFontSize',12,...
    'defaultAxesBox','on',...
    'defaultFigurePosition',[0 0 8 5],...
    'defaultFigureUnits','centimeters');

%file names for determining experimental case
bpm=20:22;
chamber_pressure=14:16;
back_pressure=7:10;
tube_length=32:34;
stroke_volume=26:28;
material=3;


legendentry={};
pointcolor={'.r','.b','.g','.m','.k'};
linecolor1={'-or','-ob','-og','-om','-ok','-oc'};
linecolor2={'-*r','-*b','-*g','-*m','-*k','-*c'};

b=[0 500 1000 1500 2000];
c=[0 50 100 150 200 250 300 350 400 450 500];
a={'e','d'};

calibration.rpm0000.values=[];
calibration.rpm0500.values=[];
calibration.rpm1000.values=[];
calibration.rpm1500.values=[];
calibration.rpm2000.values=[];

calibration.rpm0000.count=[];
calibration.rpm0500.count=[];
calibration.rpm1000.count=[];
calibration.rpm1500.count=[];
calibration.rpm2000.count=[];


i=1;
```

```
for n=1:length(data.namelistgood)

    % using find on file name to determine specific cases to plot
    name=data.namelistgood{n,1};
    a=find(data.namelistgood{n,1}(material)=='d' ...
        ... && str2double(data.namelistgood{n,1}(tube_length))==140 ...
        );

    % determining legend entries
    if a==1
        list(i)=n;
        %          legendentry{1,end+1}=['Aorta length '
num2str(str2double(data.namelistgood{n,1}(32:34))) ' mm'];
        i=i+1;
    end
end


%making the calibration .mat

%initialization
for n=1:length(list)

    name=data.namelistgood{list(n),1};

    bp=strcat('rpm', name(back_pressure));

    tl=strcat('l', name(tube_length));

    calibration.(bp).(tl).values=[];

end

for n=1:length(list)

    name=data.namelistgood{list(n),1};

    Pin=data.experiments.(name).pressure.raw.pumpinlet(500:5500);
    Pout=data.experiments.(name).pressure.raw.aortainlet(500:5500);

    Pdiff=Pin-Pout;

    bp=strcat('rpm', name(back_pressure));

    tl=strcat('l', name(tube_length));

    calibration.(bp).(tl).Pin=[calibration.(bp).(tl).values; Pin];
    calibration.(bp).(tl).Pout=[calibration.(bp).(tl).values; Pout];
    calibration.(bp).(tl).Pdiff=-[calibration.(bp).(tl).values; Pdiff];
    calibration.(bp).(tl).std.Pdiff=std(calibration.(bp).(tl).Pdiff);
    calibration.(bp).(tl).std.Pout=std(calibration.(bp).(tl).Pout);

end
```

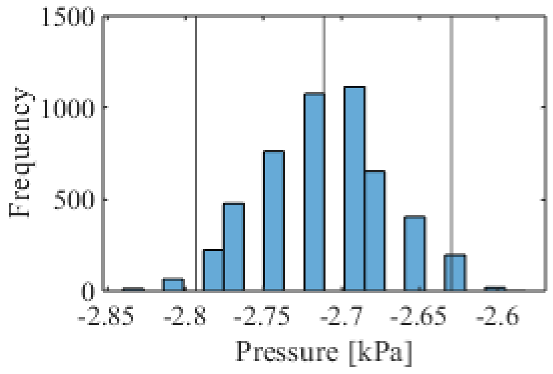
```
% calculating the mean and std

bp={'rpm0000','rpm0500','rpm1000','rpm1500','rpm2000'};

tl={'1060','1100','1140'};

for n=1:length(bp)

    for m=1:length(tl)
        % mean

        calibration.(bp{n}).(tl{m}).Pinmean=mean(calibration.(bp{n}).(tl{m}).Pin);
        calibration.(bp{n}).(tl{m}).Poutmean=mean(calibration.(bp{n}).(tl{m}).Pout);
        calibration.(bp{n}).(tl{m}).Pdiffmean=mean(calibration.(bp{n}).(tl{m}).Pdiff);


        f1=figure();
        histogram(calibration.(bp{n}).(tl{m}).Pdiff(:,1),20)

        hold on

        xline(calibration.(bp{n}).(tl{m}).Pdiffmean-2*calibration.(bp{n}).(tl{m}).std.Pdiff);
        xline(calibration.(bp{n}).(tl{m}).Pdiffmean+2*calibration.(bp{n}).(tl{m}).std.Pdiff);
        xline(calibration.(bp{n}).(tl{m}).Pdiffmean);

        xlabel('Pressure [kPa]');
        ylabel('Frequency');

        ylim([0 1500])

        %         title([bp{n} ' ' tl{m}])

        saveas(f1, char(strcat('D:\compliant
tube\code\processing\plots\NEW\backpressure_calibration\Pdiff_',...
            bp{n}, '', tl{m}, '.png')));

        %         f2=figure();
        %         histogram(calibration.(bp{n}).(tl{m}).Pout(:,1),20)
        %
        %         hold on
        %
        %         xline(calibration.(bp{n}).(tl{m}).Poutmean-
2*calibration.(bp{n}).(tl{m}).std.Pout);
        %
xline(calibration.(bp{n}).(tl{m}).Poutmean+2*calibration.(bp{n}).(tl{m}).std.Pout);
        %         xline(calibration.(bp{n}).(tl{m}).Poutmean);
        %
        %         xlabel('Pressure [kPa]');
        %         ylabel('Frequency');
        %
        %         %         title([bp{n} ' ' tl{m}])
        %
        %         saveas(f2, char(strcat('D:\compliant
tube\code\processing\plots\NEW\backpressure_calibration\Pout_',...
```

```
        %              bp{n}, '', tl{m}, '.png')));
        %
        %
        calibration.meanlist(m,n)=calibration.(bp{n}).(tl{m}).Pinmean-
calibration.(bp{n}).(tl{m}).Poutmean;
        calibration.stdlist(m,n)=calibration.(bp{n}).(tl{m}).std.Pdiff;

        calibration.Poutlist(m,n)=calibration.(bp{n}).(tl{m}).Poutmean;
        calibration.Pinlist(m,n)=calibration.(bp{n}).(tl{m}).Pinmean;


    end

    calibration.(bp{n}).Pdiffvalues=[calibration.(bp{n}).l060.Pdiff; ...
        calibration.(bp{n}).l100.Pdiff; ...
        calibration.(bp{n}).l140.Pdiff];

    calibration.(bp{n}).Pdiffstd=[calibration.(bp{n}).l060.std.Pdiff; ...
        calibration.(bp{n}).l100.std.Pdiff; ...
        calibration.(bp{n}).l140.std.Pdiff];

    calibration.(bp{n}).Poutvalues=[calibration.(bp{n}).l060.Pout; ...
        calibration.(bp{n}).l100.Pout; ...
        calibration.(bp{n}).l140.Pout];

    calibration.(bp{n}).Pinvalues=[calibration.(bp{n}).l060.Pin; ...
        calibration.(bp{n}).l100.Pin; ...
        calibration.(bp{n}).l140.Pin];

    calibration.meanalllengths(n)=mean(calibration.(bp{n}).Pdiffvalues);
    calibration.stdalllengths(n)=mean(calibration.(bp{n}).Pdiffstd);
    calibration.Poutalllengths(n)=mean(calibration.(bp{n}).Poutvalues);
    calibration.Pinalllengths(n)=mean(calibration.(bp{n}).Pinvalues);

end

save('backpressurecalibration.mat','calibration', '-v7.3')
```

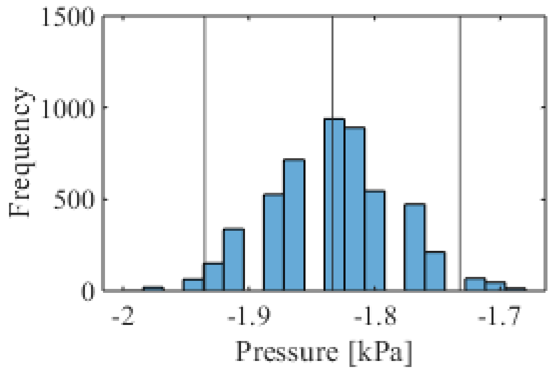




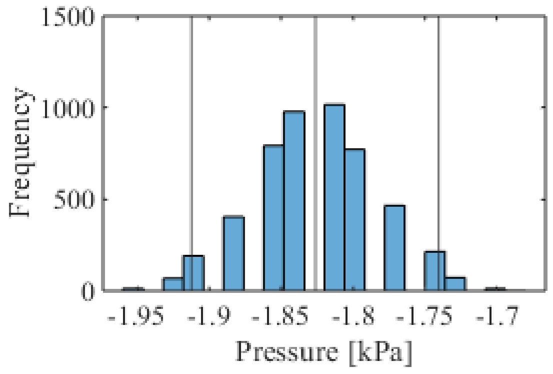
1500
1000
500
0
Frequency
-2.85
-2.8
-2.75
-2.7
-2.65
-2.6
Pressure [kPa]



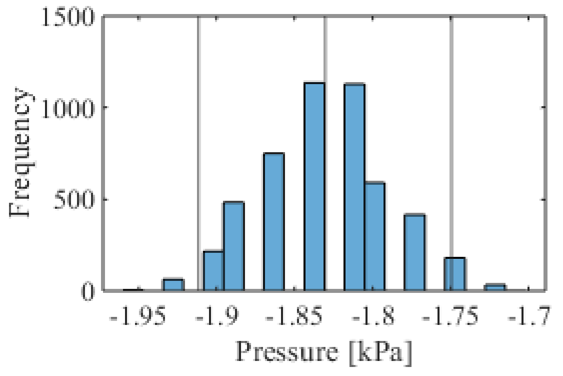
1500
1000
500
0
Frequency
-2.85
-2.8
-2.75
-2.7
-2.65
-2.6
Pressure [kPa]

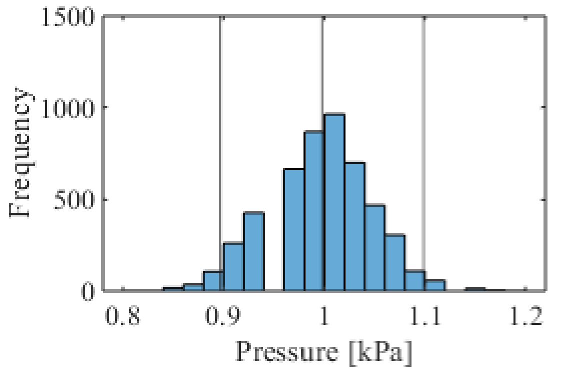


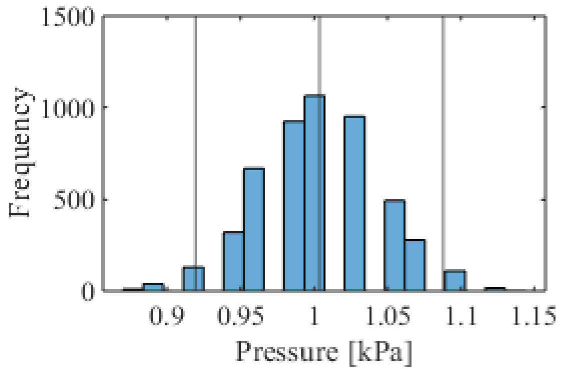
1500
1000
500
0
Frequency
-2
-1.9
-1.8
-1.7
Pressure [kPa]

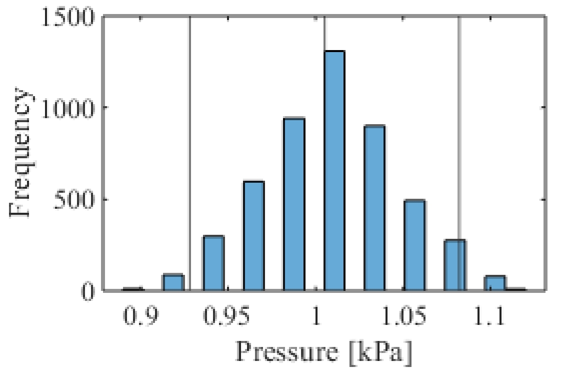


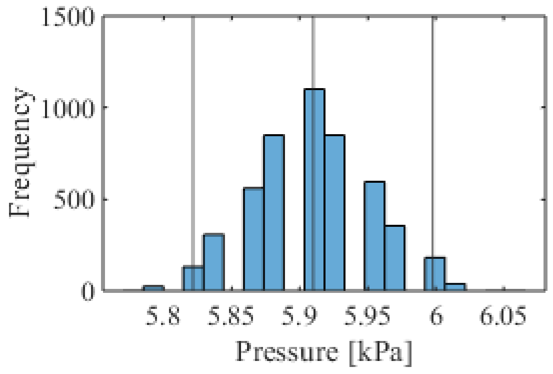
1500
1000
500
0
Frequency
-1.95
-1.9
-1.85
-1.8
-1.75
-1.7
Pressure [kPa]

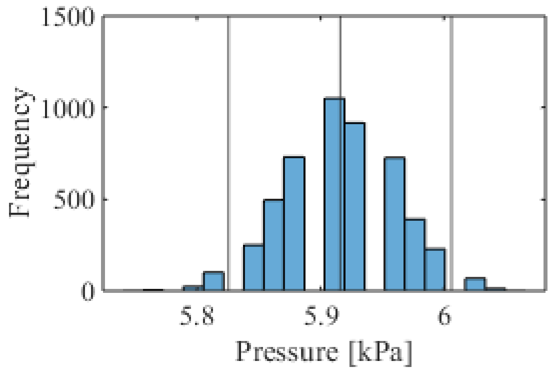

1500
1000
500
0
Frequency
-1.95
-1.9
-1.85
-1.8
-1.75
-1.7
Pressure [kPa]

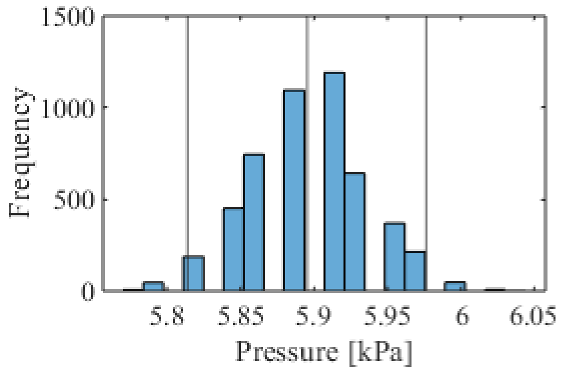

1500
1000
500
0
Frequency
0.8
0.9
1
1.1
1.2
Pressure [kPa]

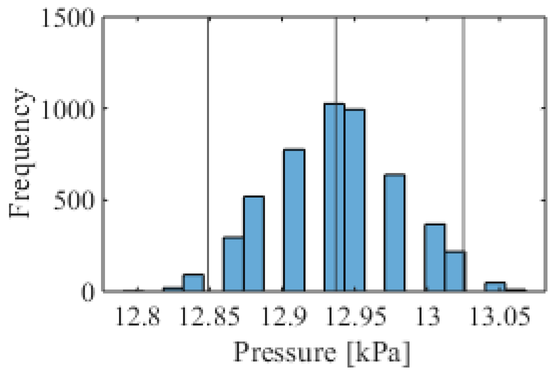

1500
1000
500
0
Frequency
0.9
0.95
1
1.05
1.1
1.15
Pressure [kPa]

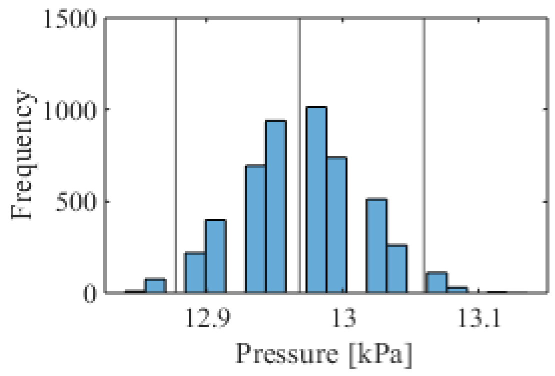

1500
1000
500
0
Frequency
0.9
0.95
1
1.05
1.1
Pressure [kPa]

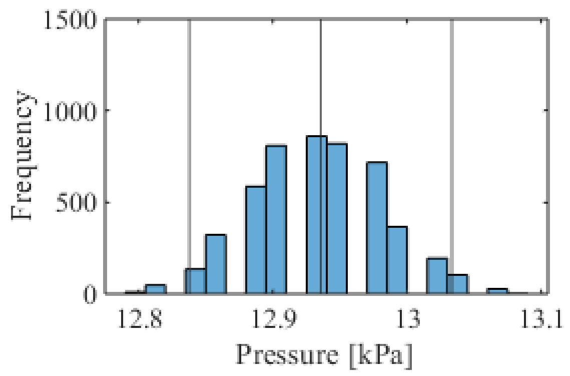

1500
1000
500
0
Frequency
5.8
5.85
5.9
5.95
6
6.05
Pressure [kPa]


1500
1000
500
0
Frequency
5.8
5.9
6
Pressure [kPa]


1500
1000
500
0
Frequency
5.8
5.85
5.9
5.95
6
6.05
Pressure [kPa]


1500
1000
500
0
Frequency
12.8
12.85
12.9
12.95
13
13.05
Pressure [kPa]





Published with MATLAB® R2019a

Appendix I. MATLAB Code for Filtering Pressure Waveforms

```
% Filtering pressure data gathered from the DAQ
```

Initialization

```
%load data.mat

fclose all
close all
clearvars -except data
clc
```

```
ans =

     0
```

variables and info

```
samplerate = 6000; %DAQ sample rate at 6000 Hz
order = 6;
framerate = 200; %camera trigger frame rate
for n=1:length(data.namelistgood)

    disp(n)

    name=data.namelistgood{n,1};
%     name=strrep(name, '-', 'x');

    %determining cut off frequency from the file name motor frequency
    cutofffreq= 20;

    [~,motorcyc]=findpeaks(data.experiments.(name).data.ai2(42000:end),'MinPeakDistance',900,
'MinPeakProminence',1,'NPeaks',4);

    motorcyc=motorcyc+42000;


    for r=4:8

        %DFT
        y=fft(data.experiments.(name).pressure.raw.(data.pressurelocations{r,2}));
        f=6000*(0:(72000/2))/72000;
        L=72000;
        p2=abs(y/L);
        p1=p2(1:L/2+1);
        p1(2:end-1)=2*p1(2:end-1);
        %applying butterworth filter
        data.experiments.(name).pressure.filtered.(data.pressurelocations{r,2})=...
            LPButterworthFilt(order, samplerate, cutofffreq,...
            data.experiments.(name).pressure.raw.(data.pressurelocations{r,2}));

    end

end

% save('data.mat','data', '-v7.3')
```

Published with MATLAB® R2019a

Appendix J. MATLAB Code for Calculating Pump Energy

```
%This version uses a polynomial fit curve for lvdt (dx)
close all
reset(groot)
clearvars -except data
clc

if exist('data','var')==0
    load data.mat
end

set(0,'defaultTextFontName','Times New Roman',...
    'defaultAxesFontName','Times New Roman',...
    'defaultAxesFontSize',12,...
    'defaultAxesBox','on',...
    'defaultFigurePosition',[0 0 16.3 10],...
    'defaultFigureUnits','centimeters');

date=datestr(datetime('now'),29);
% save(['data_' date '.mat'],'data', '-v7.3')

g=9.81; %acceleration of gravity, m/s^2
rho=1000; %density of water, kg/m^3

for n=1:length(data.namelistgood)

    disp(n)

    name=data.namelistgood{n,1};



    for ploc=1:5
        for cyc=1:7
            disp(ploc)
            disp(cyc)
            %volumetric flow rate in L/s
            %
data.experiments.(name).vdot=str2double(data.namelistgood{n,1}(26:28))*0.0001*...
            %
data.experiments.(name).lvdt.filtered(data.experiments.(name).cycle(cyc):data.experiments.(name).
cycle(cyc+1))/...
            %                 data.experiments.(name).lvdt.stroke(cyc);

            %cycle time in s

t_cyc=data.experiments.(name).time.experiment(data.experiments.(name).cycle(cyc):data.experiments
.(name).cycle(cyc+1));

            %change in lvdt distance

x_cyc=data.experiments.(name).lvdt.filtered(data.experiments.(name).cycle(cyc):data.experiments.(
name).cycle(cyc+1));
            [~,x_change]=findpeaks(-x_cyc,'NPeaks',1);

            %the method to find the direct change in pump stroke distance, dx/dt
            %             dx_dt=diff(-x_cyc)./diff(t_cyc);
            %             dx_dt(x_change:end)=0;

            %represent the change in pump stroke distance by polynomial fit curve
            p1=polyfit(t_cyc(1:x_change),x_cyc(1:x_change),2);
            y1=polyval(p1,t_cyc(1:x_change));

            p2=polyfit(t_cyc(x_change:end),x_cyc(x_change:end),6);
            y2=polyval(p2,t_cyc(x_change:end));

            dx_dt_in=diff(y1)./diff(t_cyc(1:x_change)); %dx/dt pump in stroke
            dx_dt_out=diff(y2)./diff(t_cyc(x_change:end)); %dx/dt pump out stroke
            %             dx_dt_in2=0*dx_dt_in;
            %             dx_dt=[dx_dt_in;dx_dt_out];

            p3=polyfit(t_cyc(2:x_change),dx_dt_in,1);
            y3=polyval(p3,t_cyc(2:x_change));

            p4=polyfit(t_cyc(x_change+1:end),dx_dt_out,5);
            y4=polyval(p4,t_cyc(x_change+1:end));

            dx_dt=[y3;y4];
%             dxdt=diff(x_cyc)./diff(t_cyc);
%             plot(dx_dt)

            %             plot(dx_dt)

                        if ploc==2||ploc==3||ploc==4||ploc==5
                            dx_dt(dx_dt<0)=0; %sets flow rate for in-stroke at locations
downstream of pump outlet to 0
                        elseif ploc==1
                            dx_dt(dx_dt>0)=0; %sets flow rate for out-stroke at locations
upstream of pump inlet to 0
                        end

            %             plot(dx_dt)

%             if ploc==2||ploc==3||ploc==4
%                 dx_dt(1:x_change)=dx_dt(x_change); %sets flow rate for in-stroke at locations
downstream of pump outlet to 0
%             elseif ploc==1
%                 dx_dt(x_change+1:end)=dx_dt(x_change+1); %sets flow rate for out-stroke at
locations upstream of pump inlet to 0
%             end
            %              plot(dx_dt)
            %figure to plot original LVDT data against polyfit curve
%             f1=figure(1);
%             plot(t_cyc(1:x_change),x_cyc(1:x_change),'ro',t_cyc(1:x_change),y1,'k-
','LineWidth',2)
%             hold on
%             plot(t_cyc(x_change:end),x_cyc(x_change:end),'bo',t_cyc(x_change:end),y2,'y-
','LineWidth',2)
%             xlabel('Time [s]'); ylabel('Distance [m]');
%             legend('raw in-stroke','polynomial in-stroke','raw out-stroke','polynomial
outstroke','location','southeast')
%             saveas(f1,'D:\compliant tube\code\processing\plots\test\lvdt_vs_fit_2.png');
%             hold off

%             f2=figure(2);
%             plot(t_cyc(1:x_change),y1,t_cyc(x_change:end),y2)
%             xlabel('Time [s]'); ylabel('Distance [m]')
%             saveas(f2,'D:\compliant
tube\code\processing\plots\test\lvdt_fit_innout_stroke_2.png');
%
%             f3=figure(3);
%             plot(t_cyc(2:x_change),y3,t_cyc(x_change+1:end),y4,'LineWidth',2)
%             xlabel('Time [s]'); ylabel('dx/dt [m]');
%             legend('dx/dt in-stroke','dx/dt out-stroke','location','southeast')
%             saveas(f3,'D:\compliant tube\code\processing\plots\test\lvdt_dxdt_2.png');
%
%                         f4=figure(4);
%             plot(t_cyc(2:x_change),dx_dt_in,'ro',t_cyc(2:x_change),y3,'k-','LineWidth',2)
%             hold on
%             plot(t_cyc(x_change+1:end),dx_dt_out,'bo',t_cyc(x_change+1:end),y4,'y-
','LineWidth',2)
%             xlabel('Time [s]'); ylabel('Distance [m]');
%             legend('raw in-stroke','polynomial in-stroke','raw out-stroke','polynomial
outstroke','location','southeast')
% %             saveas(f4,'D:\compliant tube\code\processing\plots\test\lvdt_vs_fit_2.png');
%             hold off

            %volumetric flow rate is in 0.1 ml, to convert to m^3, multiply
            %by 0.0000001
            Qdot_t=str2double(data.namelistgood{n,1}(26:28))*1e-7.*dx_dt/...
                data.experiments.(name).lvdt.stroke(cyc);

            %             plot(Qdot_t)

            %diameter in m
            d=data.pressurelocations{ploc+4,3};

            %area
            A=pi/4*(d^2);

            %density of water:
            rho=1000;
```

pressure

```
            %pressure data is originally in kPa. Converted to Pa

P_t=1000.*data.experiments.(name).pressure.filtered.(data.pressurelocations{ploc+4,2})(data.exper
iments.(name).cycle(cyc):data.experiments.(name).cycle(cyc+1));


            % %pump energy calculated in J
            %

data.experiments.(name).energy.pressure.(data.pressurelocations{ploc+4,2}).cycles(cyc)=...
                trapz(t_cyc(2:end), Qdot_t.*P_t(2:end));
            %
```

kinetic

```
            % pump energy calculated in J
```

```
            Ekin_int=(Qdot_t).*((Qdot_t./A).^2);


data.experiments.(name).energy.kinetic.(data.pressurelocations{ploc+4,2}).cycles(cyc)=...
                rho*0.5*trapz(t_cyc(2:end), Ekin_int);
```

static

```
            %              % pump energy calculated in J

            h_length=data.namelistgood{n,1}(32:34);

            switch ploc
                case {1,2,3} %for pump inlet and outlet, and aorta inlet
                    Estat_int=rho*g*data.pressurelocations{ploc+4,4}*...
                        trapz(t_cyc(2:end), Qdot_t);

                case {4,5} %for aorta outlet (height changes)

                     switch h_length
                        case '060'
                            Estat_int=rho*g*data.pressurelocations{ploc+4,4}*...
                                trapz(t_cyc(2:end), Qdot_t);
                        case '100'
                            Estat_int=rho*g*data.pressurelocations{ploc+4,5}*...
                                trapz(t_cyc(2:end), Qdot_t);
                        case '140'
                            Estat_int=rho*g*data.pressurelocations{ploc+4,6}*...
                                trapz(t_cyc(2:end), Qdot_t);
                    end
            end

data.experiments.(name).energy.static.(data.pressurelocations{ploc+4,2}).cycles(cyc)=...
                -Estat_int;
```

local acceleration

pump energy calculated in J

```
            dv_dt=diff(Qdot_t/A)./diff(t_cyc(2:end));

            switch ploc
                case {1,2,3} %for pump inlet and outlet, and aorta inlet
                    Elocaccel_int=rho*data.pressurelocations{ploc+4,4}*...
                        trapz(t_cyc(3:end), (Qdot_t(2:end)).*dv_dt);

                case {4,5} %for aorta outlet (height changes)

                    h_length=data.namelistgood{n,1}(32:34);

                    switch h_length
                        case '060'
                            Elocaccel_int=rho*data.pressurelocations{ploc+4,4}*...
                                trapz(t_cyc(3:end), Qdot_t(2:end).*dv_dt);
                        case '100'
                            Elocaccel_int=rho*data.pressurelocations{ploc+4,5}*...
                                trapz(t_cyc(3:end), Qdot_t(2:end).*dv_dt);
                        case '140'
                            Elocaccel_int=rho*data.pressurelocations{ploc+4,6}*...
                                trapz(t_cyc(3:end), Qdot_t(2:end).*dv_dt);
                    end
            end

data.experiments.(name).energy.locaccel.(data.pressurelocations{ploc+4,2}).cycles(cyc)=...
                Elocaccel_int;

%              rmfield(data.experiments.(name).energy,total);


data.experiments.(name).energy.total.(data.pressurelocations{ploc+4,2}).cycles(cyc)=...

data.experiments.(name).energy.pressure.(data.pressurelocations{ploc+4,2}).cycles(cyc)+...

data.experiments.(name).energy.kinetic.(data.pressurelocations{ploc+4,2}).cycles(cyc)+...

data.experiments.(name).energy.static.(data.pressurelocations{ploc+4,2}).cycles(cyc)+...

data.experiments.(name).energy.locaccel.(data.pressurelocations{ploc+4,2}).cycles(cyc);
        end
        %
        data.experiments.(name).energy.pressure.(data.pressurelocations{ploc+4,2}).mean=...

mean(data.experiments.(name).energy.pressure.(data.pressurelocations{ploc+4,2}).cycles);

        data.experiments.(name).energy.kinetic.(data.pressurelocations{ploc+4,2}).mean=...

mean(data.experiments.(name).energy.kinetic.(data.pressurelocations{ploc+4,2}).cycles);


        data.experiments.(name).energy.static.(data.pressurelocations{ploc+4,2}).mean=...

mean(data.experiments.(name).energy.static.(data.pressurelocations{ploc+4,2}).cycles);

        data.experiments.(name).energy.locaccel.(data.pressurelocations{ploc+4,2}).mean=...

mean(data.experiments.(name).energy.locaccel.(data.pressurelocations{ploc+4,2}).cycles);

        data.experiments.(name).energy.total.(data.pressurelocations{ploc+4,2}).mean=...
            data.experiments.(name).energy.pressure.(data.pressurelocations{ploc+4,2}).mean+...
            data.experiments.(name).energy.kinetic.(data.pressurelocations{ploc+4,2}).mean+...
            data.experiments.(name).energy.static.(data.pressurelocations{ploc+4,2}).mean+...
            data.experiments.(name).energy.locaccel.(data.pressurelocations{ploc+4,2}).mean;
    end
end
%

% save('data.mat','data', '-v7.3')

% for n=1:length(data.namelistgood)
%
%     disp(n)
%
%     name=data.namelistgood{n,1};
%
%
%
%     for ploc=1:4
%         for cyc=1:7
%          data.experiments.(name).energy.total.(data.pressurelocations{ploc+4,2})=...
%
data.experiments.(name).energy.pressure.(data.pressurelocations{ploc+4,2}).mean+...
%              data.experiments.(name).energy.kinetic.(data.pressurelocations{ploc+4,2}).mean+...
%              data.experiments.(name).energy.static.(data.pressurelocations{ploc+4,2}).mean+...
%              data.experiments.(name).energy.locaccel.(data.pressurelocations{ploc+4,2}).mean;
%         end
%     end
% end
```

Published with MATLAB® R2019a

Appendix K. MATLAB Code for Calculating Percent Energy Decrease from the Aorta Inlet to the Aorta Outlet

```
close all
reset(groot)
clearvars -except data
clc

if exist('data','var')==0
    load data.mat
end

for n=1:length(data.namelistgood)

    name=data.namelistgood{n,1};

    data.experiments.(name).energy.diff.AortaOutVsAortaIn=...
    data.experiments.(name).energy.total.aortaoutlet.mean-...
    data.experiments.(name).energy.total.aortainlet.mean;


    data.experiments.(name).energy.percentdiff.AortaOutVsAortaIn=...
        ((data.experiments.(name).energy.total.aortaoutlet.mean-...
    data.experiments.(name).energy.total.aortainlet.mean)./ ...
    data.experiments.(name).energy.total.aortainlet.mean).*100;

data.experiments.(name).energy.diff.AortaOutShortVsAortaIn=...
    data.experiments.(name).energy.total.aortaoutletshort.mean-...
    data.experiments.(name).energy.total.aortainlet.mean;


    data.experiments.(name).energy.percentdiff.AortaOutShortVsAortaIn=...
        ((data.experiments.(name).energy.total.aortaoutletshort.mean-...
    data.experiments.(name).energy.total.aortainlet.mean)./ ...
    data.experiments.(name).energy.total.aortainlet.mean).*100;


end
```

Published with MATLAB® R2019a

Appendix L. *E*, Pump Energy Per Cycle Vs Backpressure Plots Separated by Aorta Length

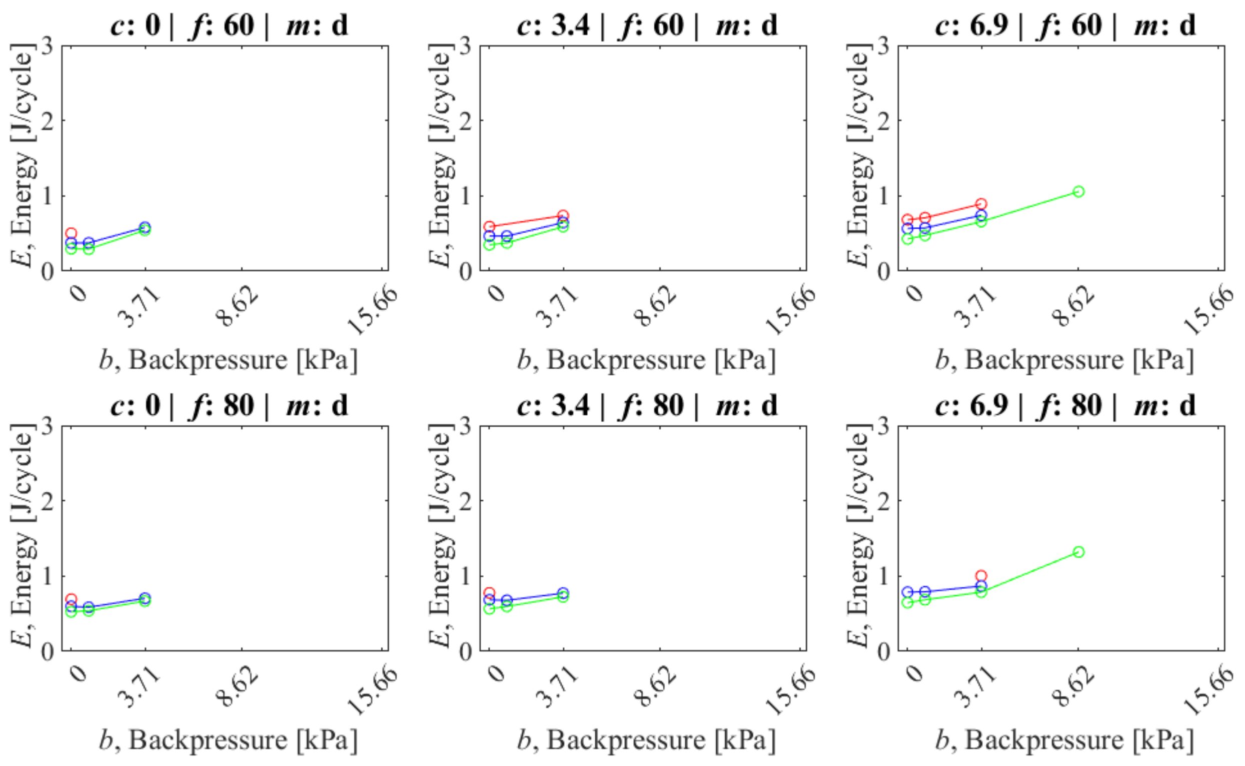
c: chamber pressure [kPa]

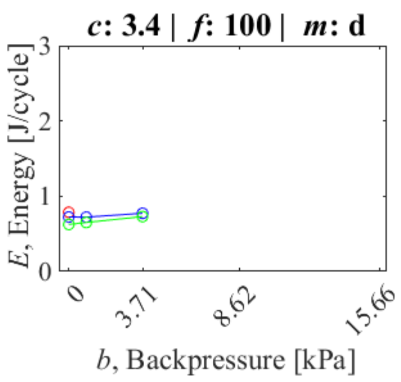
f: pulse frequency [bpm]

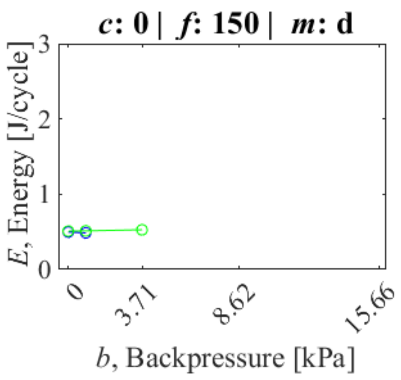
m: material, e for Ecoflex, d for Dragon Skin

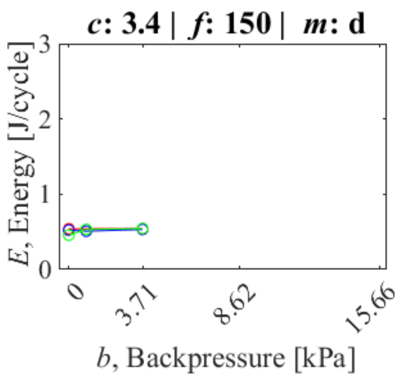
l: aorta lengths [mm] of red: 60; blue: 100; green: 140.

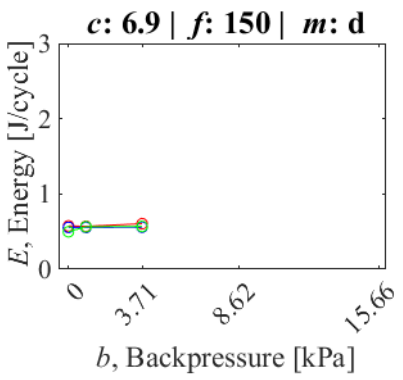
All empty spaces signify insufficient data points to generate plot for that case.

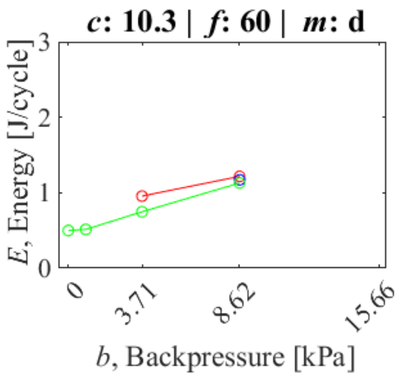


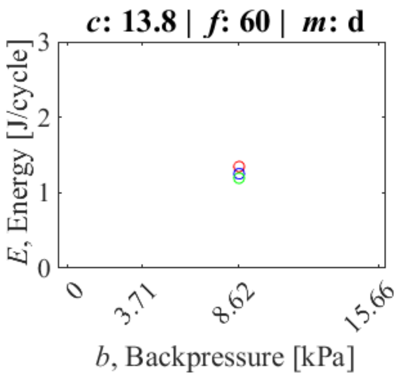

c: 3.4 | f: 100 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
3.71
8.62
15.66

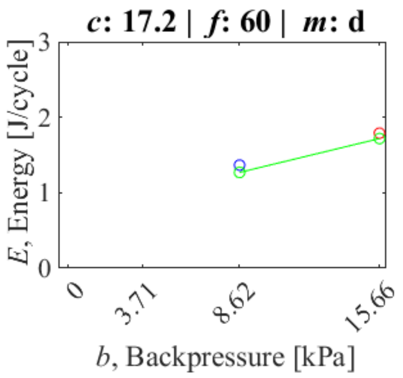

c: 0 | f: 150 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

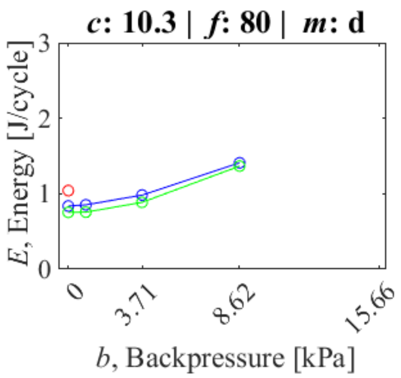

c: 3.4 | f: 150 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

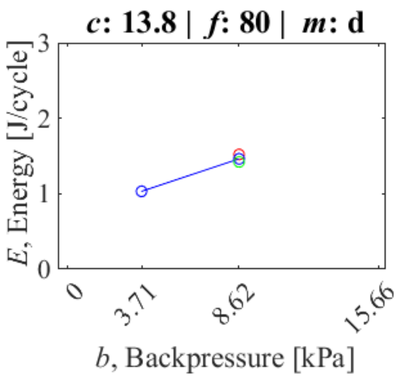

c: 6.9 | f: 150 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

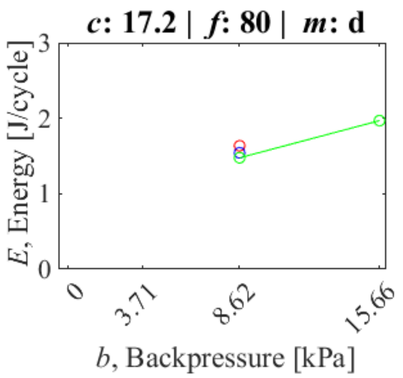

c: 10.3 | f: 60 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

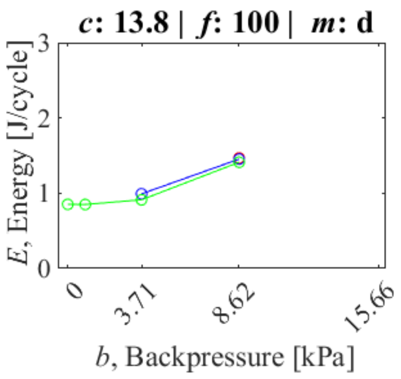

c: 13.8 | f: 60 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

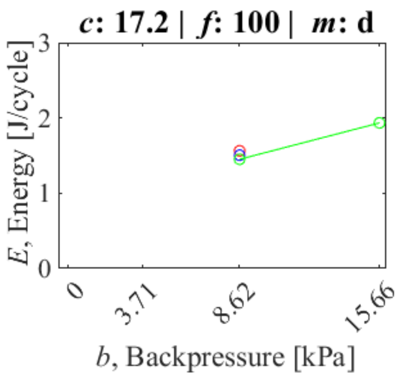

c: 17.2 | f: 60 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

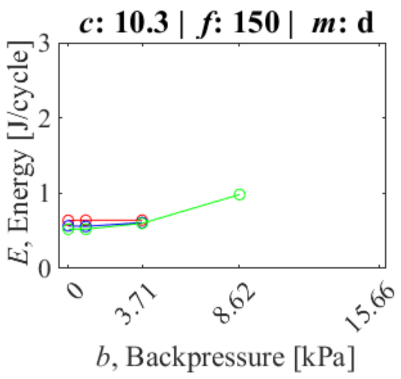

c: 10.3 | f: 80 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

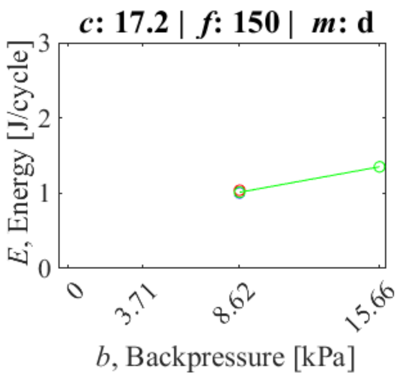

c: 13.8 | f: 80 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

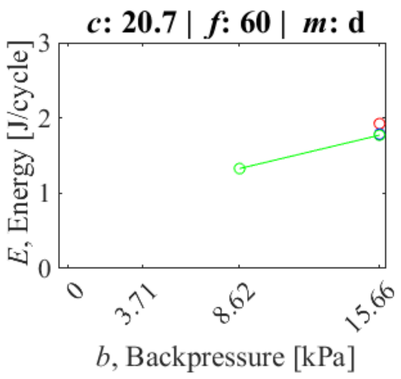

c: 17.2 | f: 80 | m: d
E, Energy [J/cycle]
b, Backpressure [kPa]

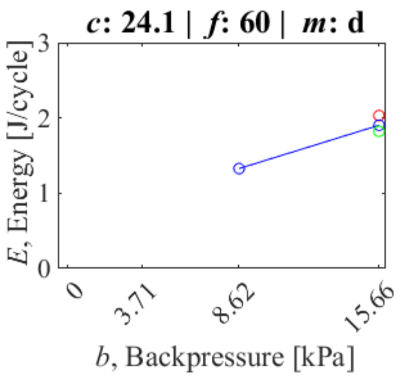

c: 13.8 | *f*: 100 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

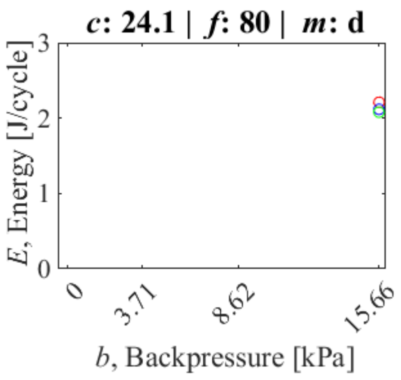

c: 17.2 | *f*: 100 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

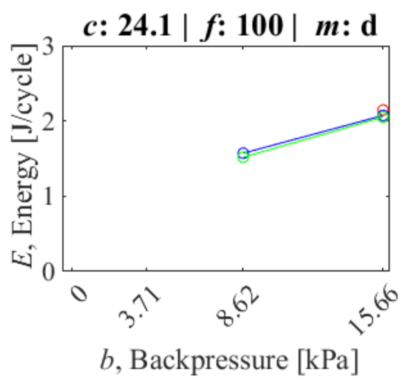

c: 10.3 | *f*: 150 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

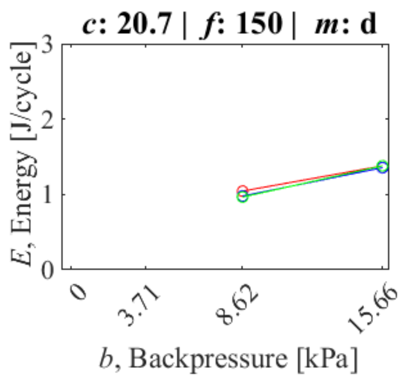

c: 17.2 | *f*: 150 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

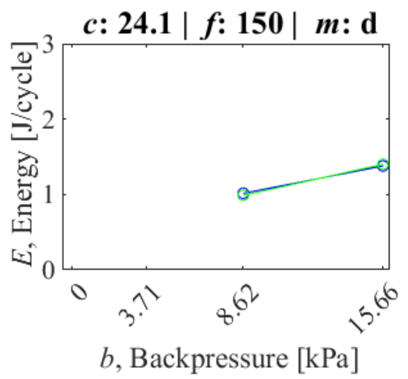

c: 20.7 | *f*: 60 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

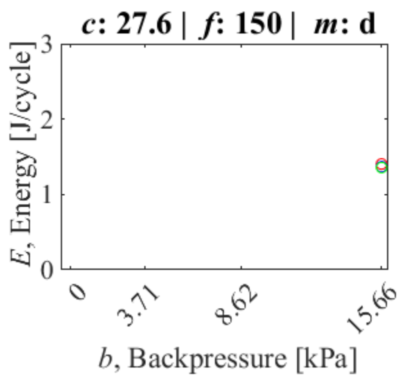

c: 24.1 | *f*: 60 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

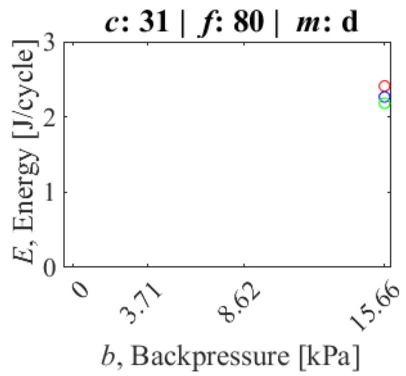

c: 24.1 | *f*: 80 | *m*: d
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

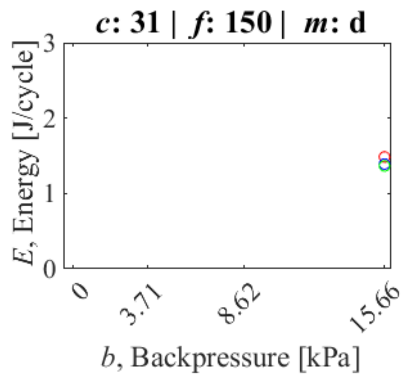

c: 24.1 | *f*: 100 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66

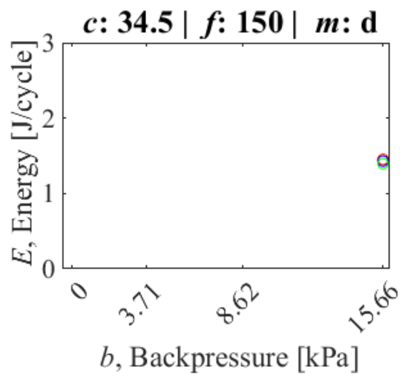

c: 20.7 | *f*: 150 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66

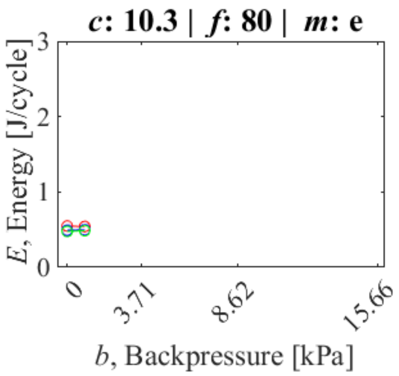

c: 24.1 | *f*: 150 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66

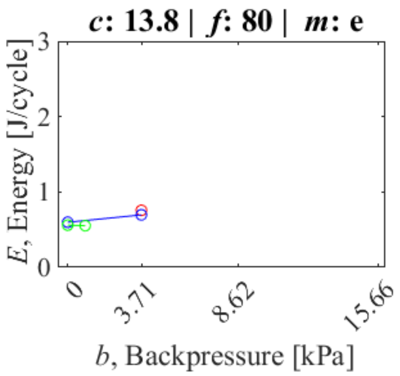

c: 27.6 | *f*: 150 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66

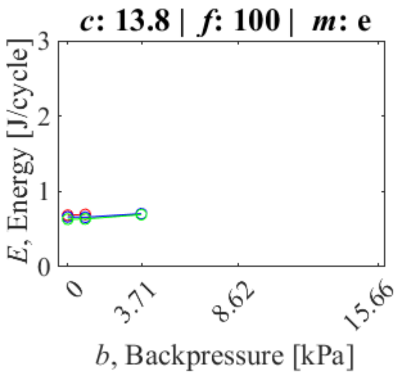

c: 31 | *f*: 80 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66

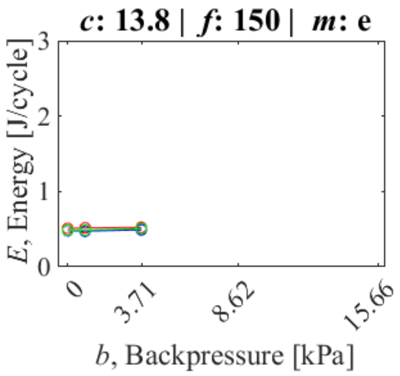

c: 31 | *f*: 150 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66


c: 34.5 | *f*: 150 | *m*: d
E, Energy [J/cycle]
b, Backpressure [kPa]
0
1
2
3
0
3.71
8.62
15.66


c: 10.3 | f: 80 | m: e
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 13.8 | f: 80 | m: e
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 13.8 | f: 100 | m: e
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 13.8 | f: 150 | m: e
E, Energy [J/cycle]
0
1
2
3
0
3.71
8.62
15.66
b, Backpressure [kPa]

Appendix M. *ΔE*, Percent Energy Decrease Vs Backpressure Plots Separated by Aorta Length

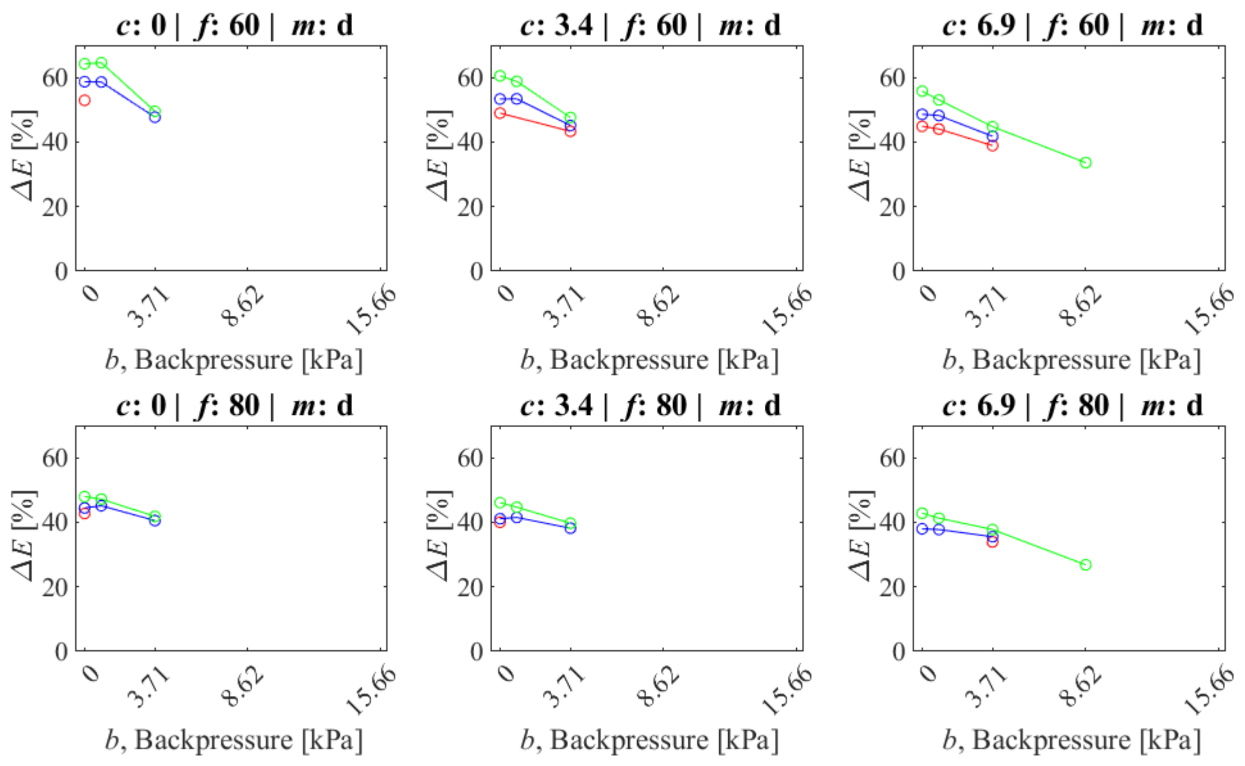
c: chamber pressure [kPa]

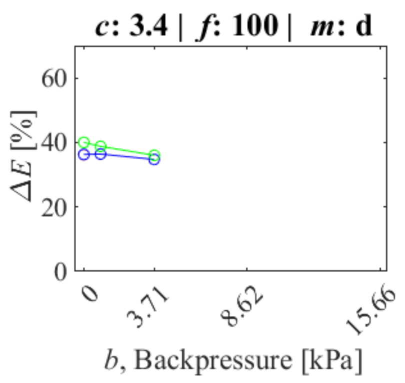
f: pulse frequency [bpm]

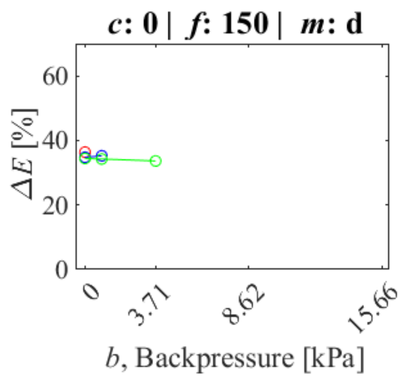
m: material, e for Ecoflex, d for Dragon Skin

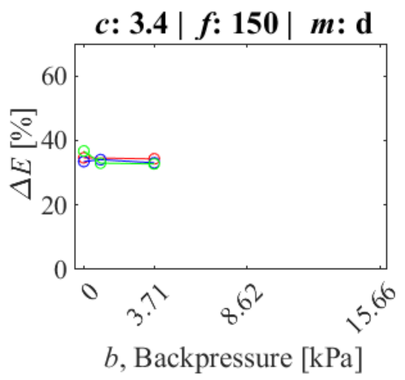
l: aorta lengths [mm] of red: 60; blue: 100; green: 140.

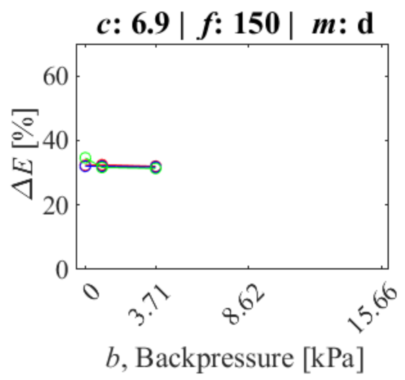
All empty spaces signify insufficient data points to generate plot for that case.

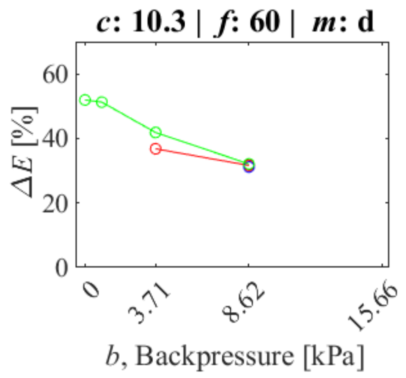


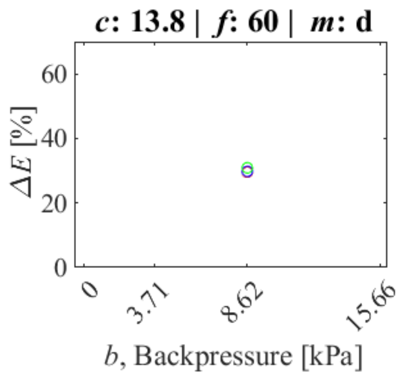

c: 3.4 | f: 100 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

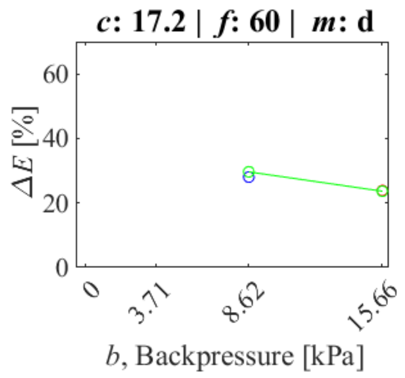

c: 0 | f: 150 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

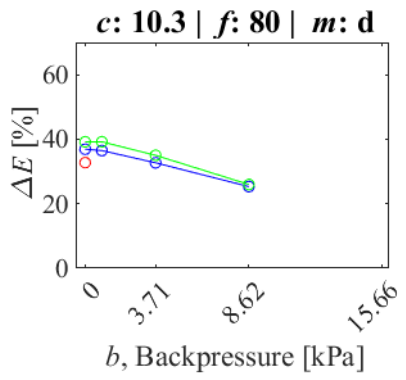

c: 3.4 | f: 150 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

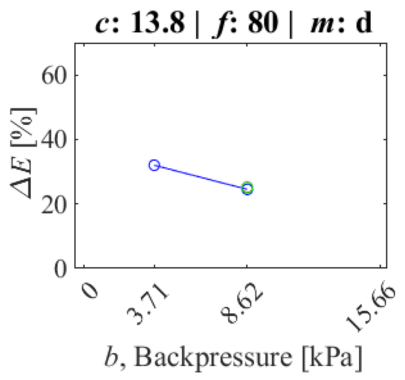

c: 6.9 | f: 150 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

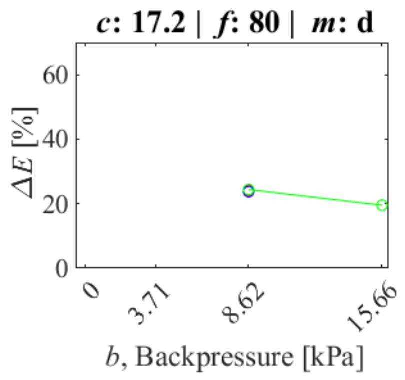

c: 10.3 | f: 60 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

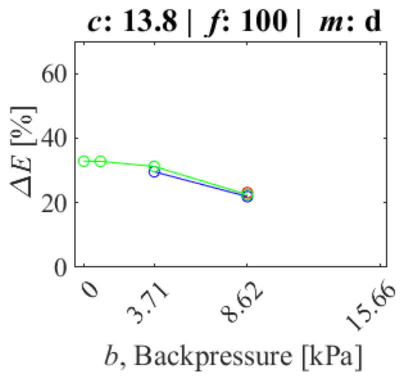

c: 13.8 | f: 60 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

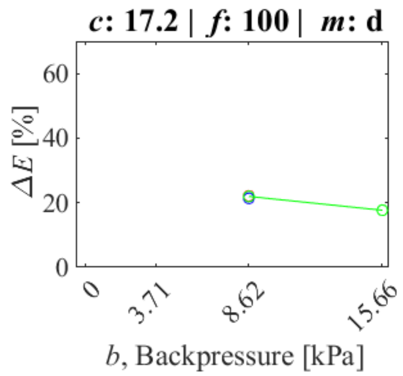

c: 17.2 | f: 60 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

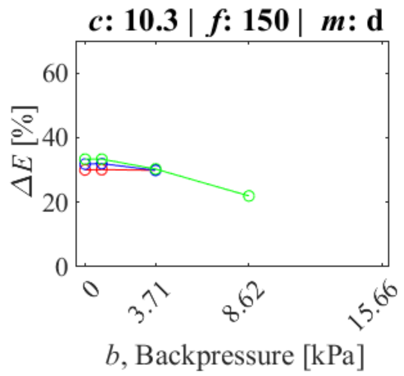

c: 10.3 | f: 80 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

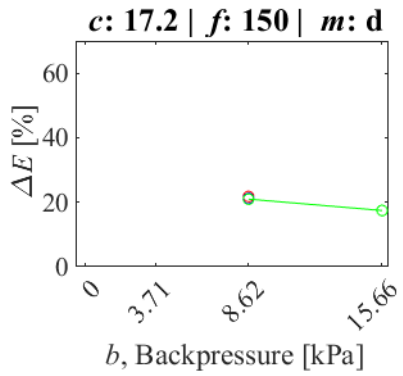

c: 13.8 | f: 80 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

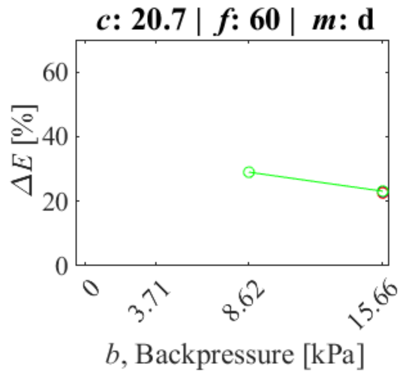

c: 17.2 | f: 80 | m: d
ΔE [%]
60
40
20
0
0
3.71
8.62
15.66
b, Backpressure [kPa]

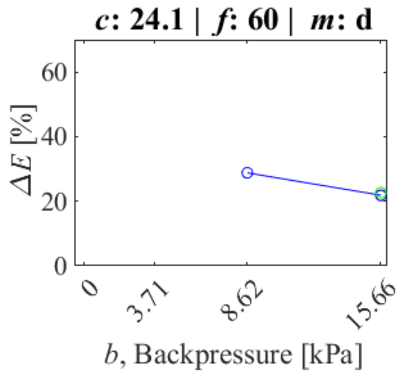

c: 13.8 | *f*: 100 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]

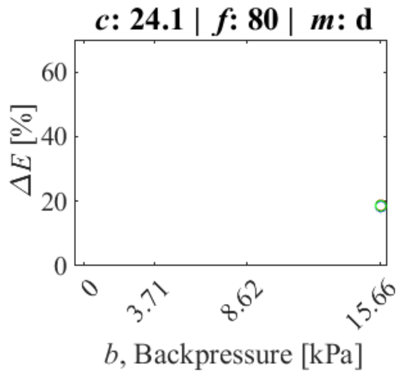

c: 17.2 | *f*: 100 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]

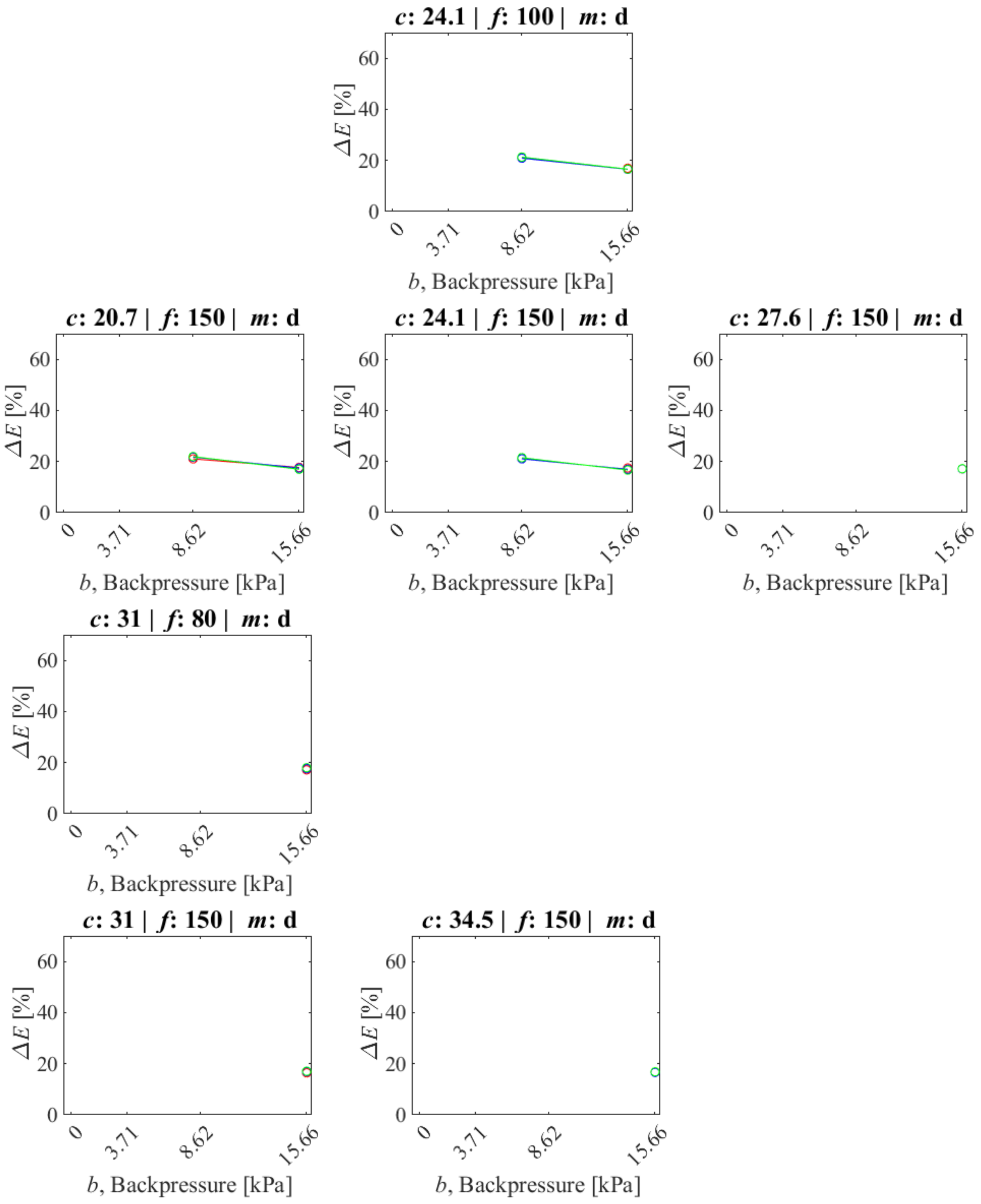

c: 10.3 | *f*: 150 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]

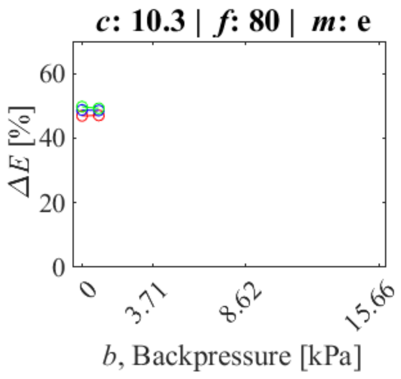

c: 17.2 | *f*: 150 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 20.7 | *f*: 60 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]

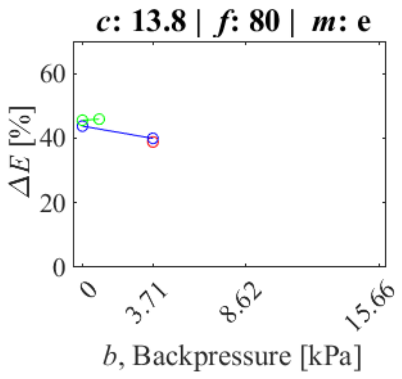

c: 24.1 | *f*: 60 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 24.1 | *f*: 80 | *m*: d
0
20
40
60
Δ*E* [%]
0
3.71
8.62
15.66
b, Backpressure [kPa]

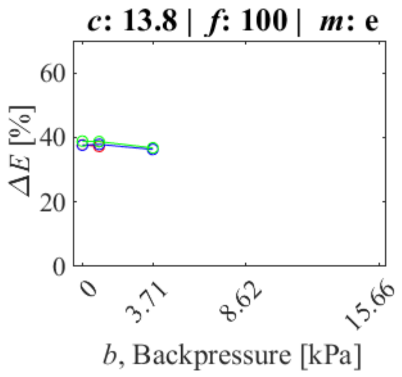

c: 24.1 | f: 100 | m: d
c: 20.7 | f: 150 | m: d
c: 24.1 | f: 150 | m: d
c: 27.6 | f: 150 | m: d
c: 31 | f: 80 | m: d
c: 31 | f: 150 | m: d
c: 34.5 | f: 150 | m: d
ΔE [%]
0
20
40
60
0
3.71
8.62
15.66
b, Backpressure [kPa]



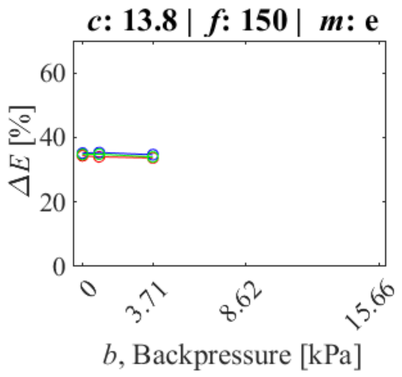
c: 10.3 | f: 80 | m: e
ΔE [%]
b, Backpressure [kPa]



c: 13.8 | f: 80 | m: e
ΔE [%]
b, Backpressure [kPa]



c: 13.8 | f: 100 | m: e
ΔE [%]
b, Backpressure [kPa]



c: 13.8 | f: 150 | m: e
ΔE [%]
b, Backpressure [kPa]

Appendix N. *ΔE*, Percent Energy Change Vs Pulse Frequency Plots Separated by Aorta Length

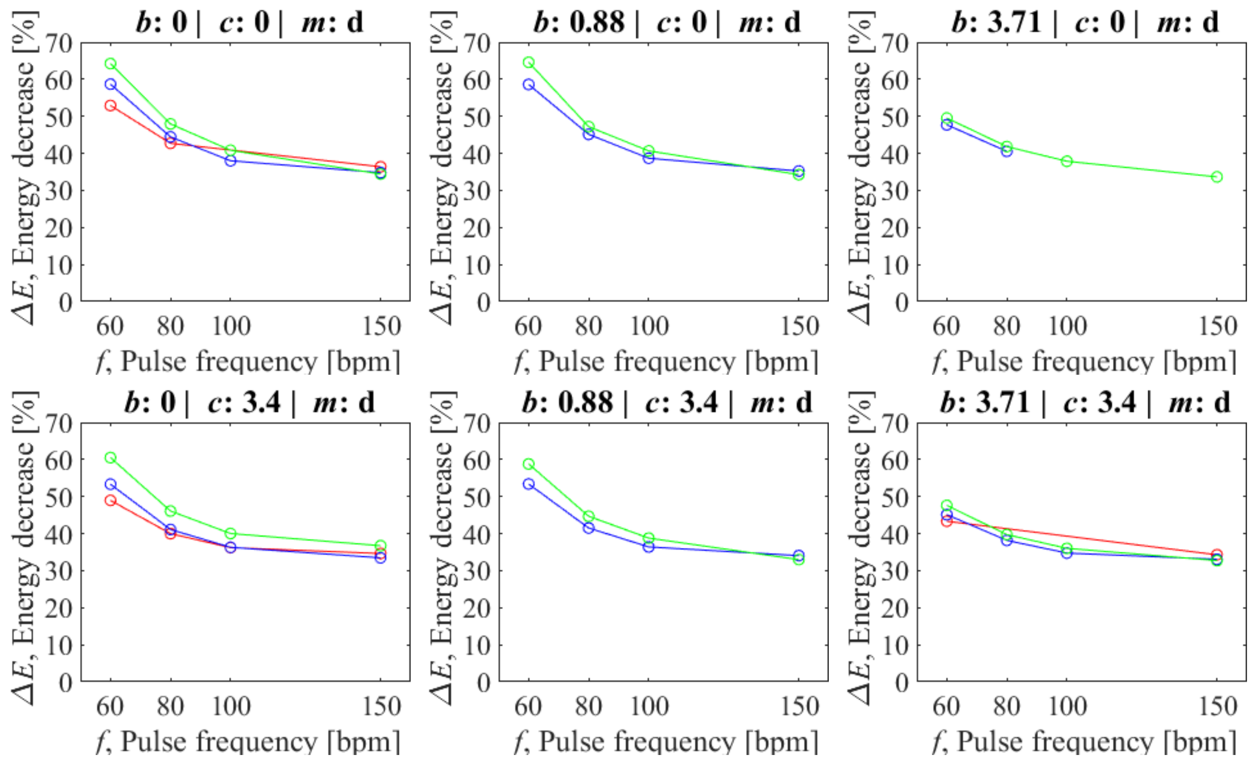
b: backpressure [kPa]

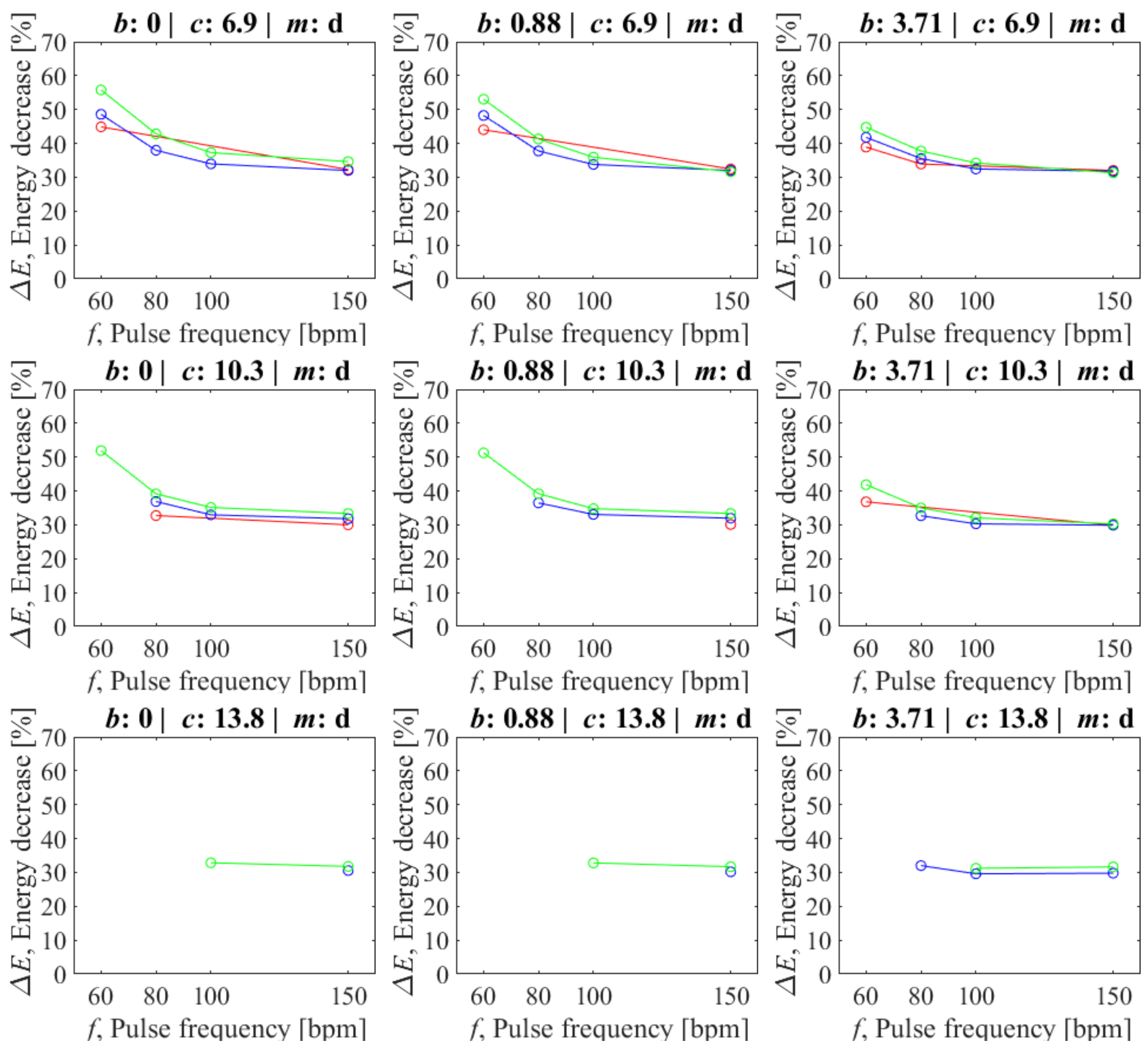
c: chamber pressure [kPa]

m: material, e for Ecoflex, d for Dragon Skin

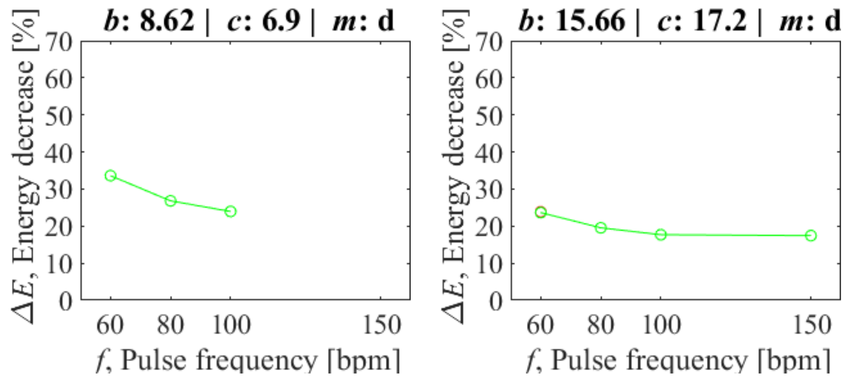
l: aorta lengths [mm] of red: 60; blue: 100; green: 140.

All empty spaces signify insufficient data points to generate plot for that case.

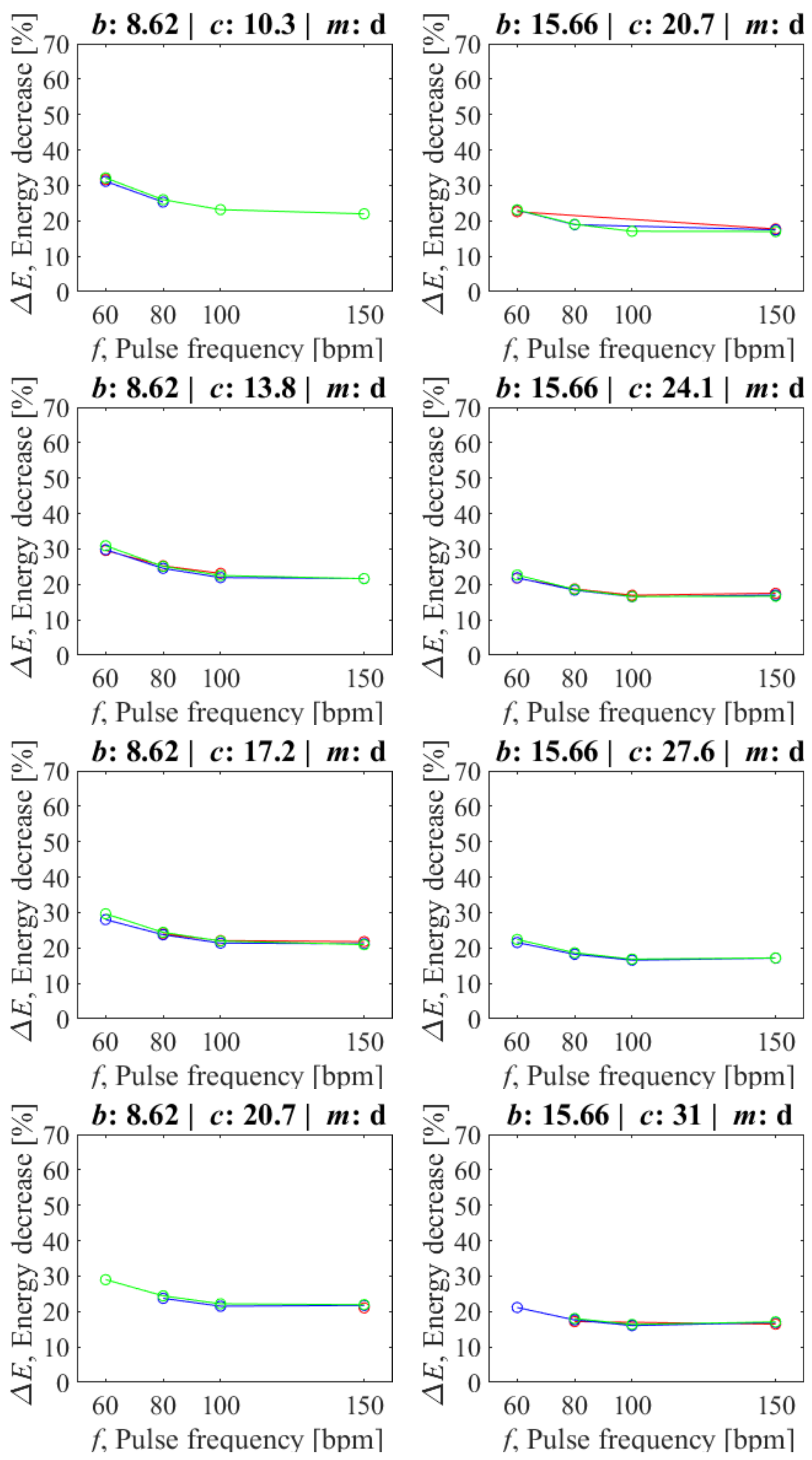


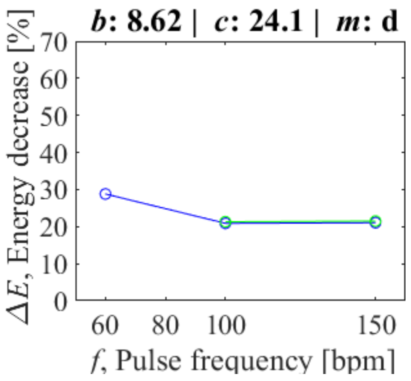


b: 0 | c: 6.9 | m: d
b: 0.88 | c: 6.9 | m: d
b: 3.71 | c: 6.9 | m: d
b: 0 | c: 10.3 | m: d
b: 0.88 | c: 10.3 | m: d
b: 3.71 | c: 10.3 | m: d
b: 0 | c: 13.8 | m: d
b: 0.88 | c: 13.8 | m: d
b: 3.71 | c: 13.8 | m: d
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]

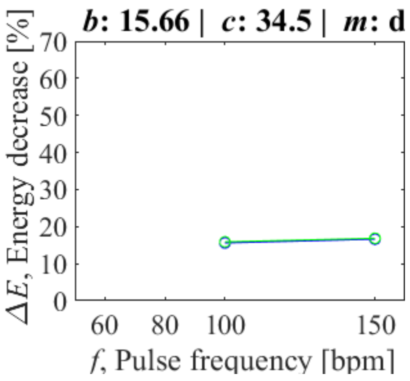


b: 8.62 | c: 6.9 | m: d
b: 15.66 | c: 17.2 | m: d
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]

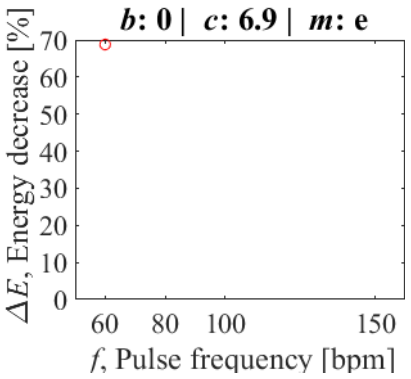

b: 8.62 | c: 10.3 | m: d
b: 15.66 | c: 20.7 | m: d
b: 8.62 | c: 13.8 | m: d
b: 15.66 | c: 24.1 | m: d
b: 8.62 | c: 17.2 | m: d
b: 15.66 | c: 27.6 | m: d
b: 8.62 | c: 20.7 | m: d
b: 15.66 | c: 31 | m: d
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]
0
10
20
30
40
50
60
70
60
80
100
150



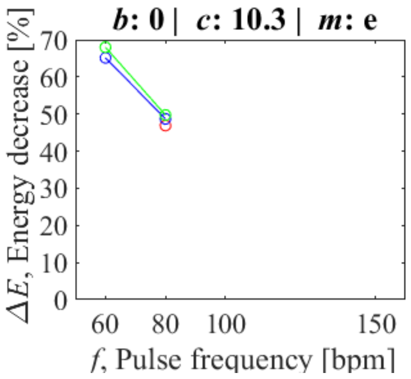
b: 8.62 | c: 24.1 | m: d
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]



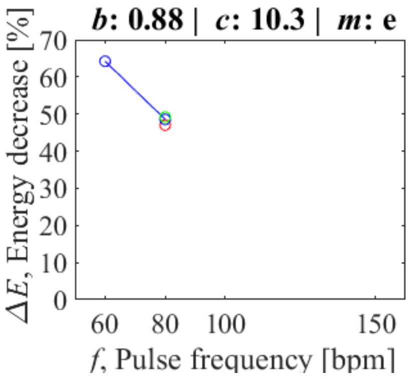
b: 15.66 | c: 34.5 | m: d
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]



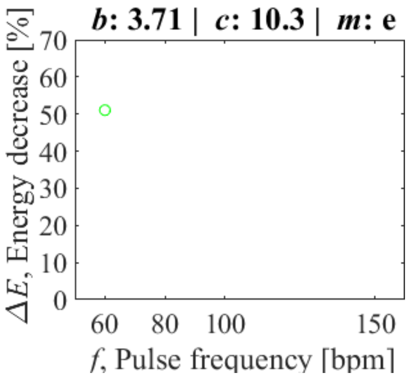
b: 0 | c: 6.9 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]



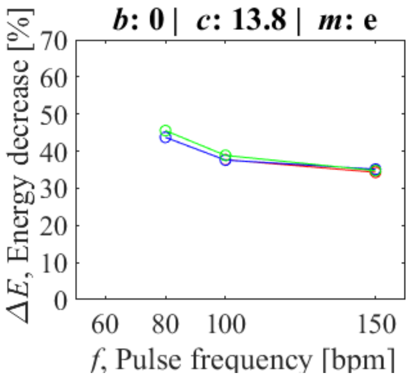
b: 0 | c: 10.3 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]



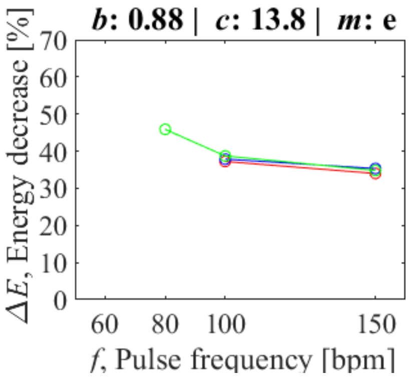
b: 0.88 | c: 10.3 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]



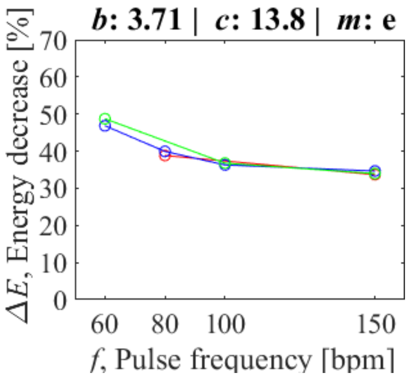
b: 3.71 | c: 10.3 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]



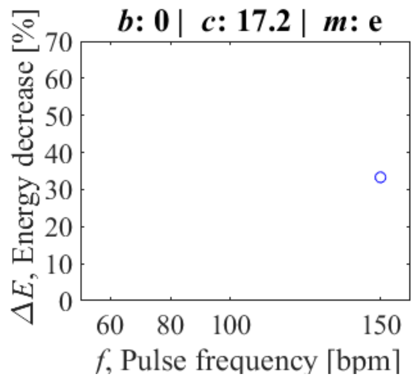
b: 0 | c: 13.8 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]

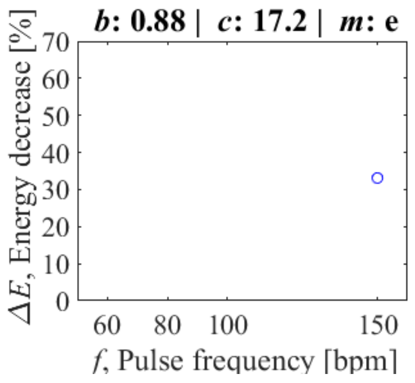


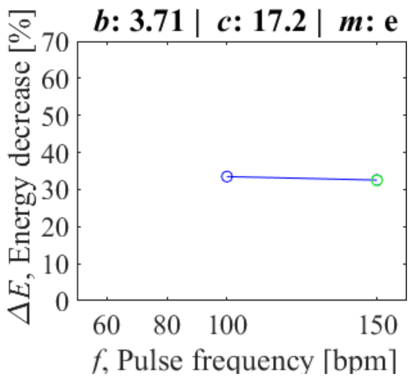
b: 0.88 | c: 13.8 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]

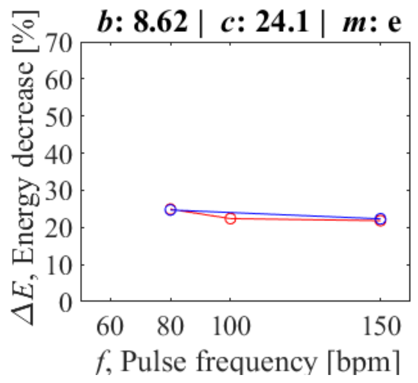


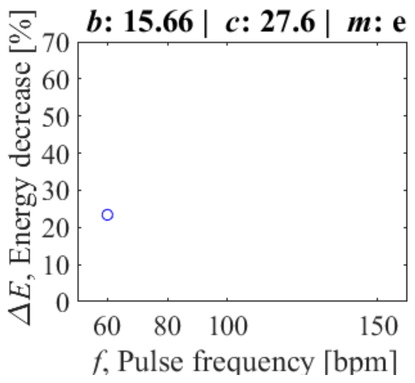
b: 3.71 | c: 13.8 | m: e
ΔE, Energy decrease [%]
f, Pulse frequency [bpm]

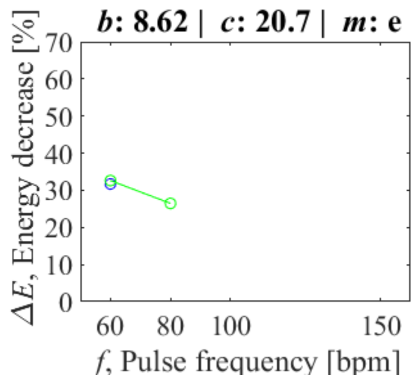

b: 0 | *c*: 17.2 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]

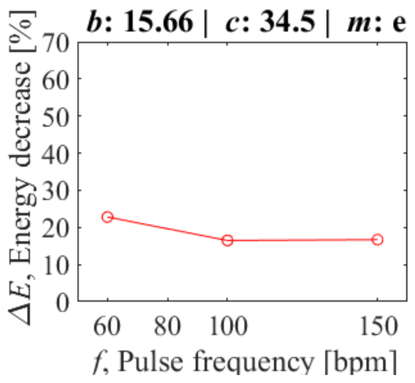

b: 0.88 | *c*: 17.2 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]

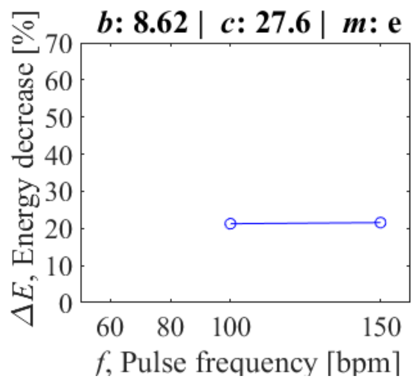

b: 3.71 | *c*: 17.2 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]

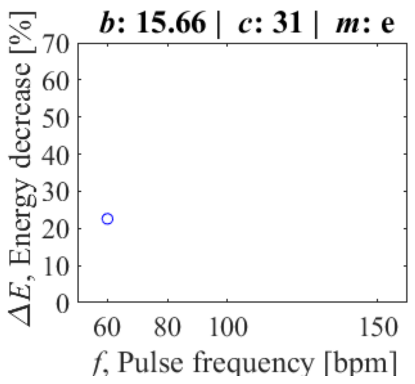

b: 8.62 | *c*: 24.1 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]


b: 15.66 | *c*: 27.6 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]


b: 8.62 | *c*: 20.7 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]


b: 15.66 | *c*: 34.5 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]


b: 8.62 | *c*: 27.6 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]


b: 15.66 | *c*: 31 | *m*: e
Δ*E*, Energy decrease [%]
f, Pulse frequency [bpm]

Appendix O. $\frac{\Delta D}{D}$, Maximum Percent Diameter Distension Vs Backpressure Plots Separated by Aorta Length

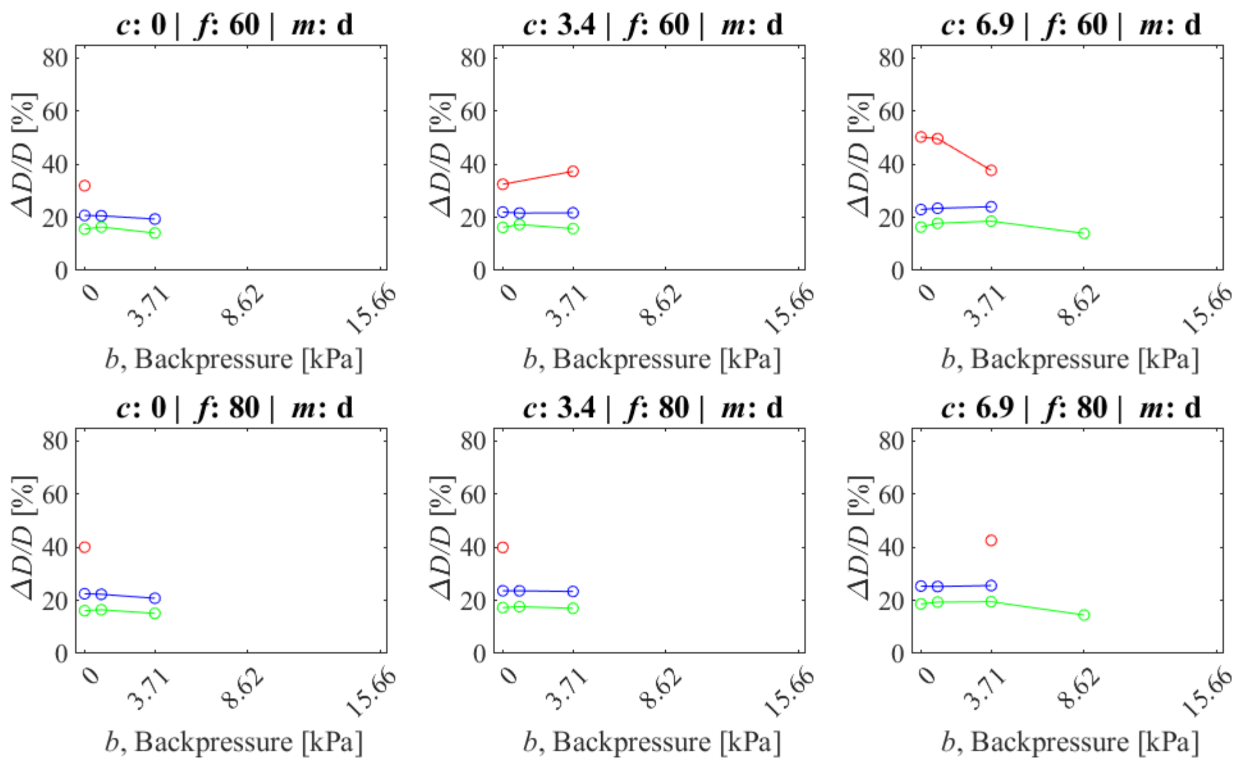
b: backpressure [kPa]

c: chamber pressure [kPa]

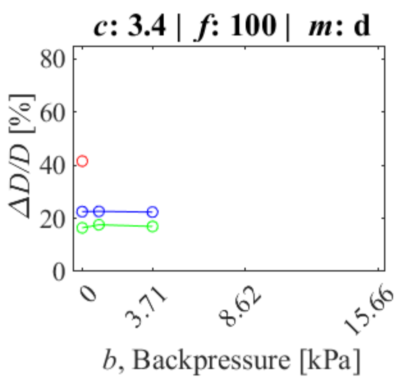
m: material, e for Ecoflex, d for Dragon Skin

l: aorta lengths [mm] of red: 60; blue: 100; green: 140.

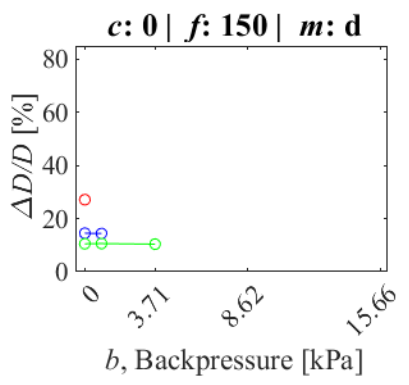
All empty spaces signify insufficient data points to generate plot for that case.

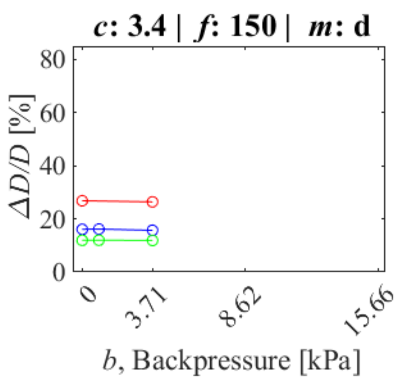


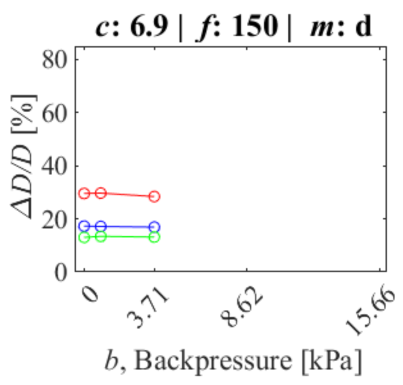
Insufficient data points for
c:0 | *f*:100 | *m*: d

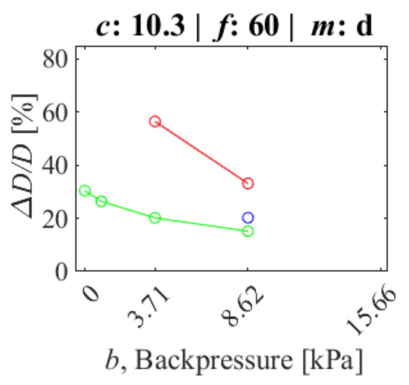


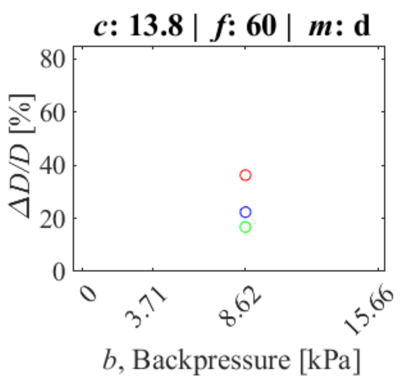
Insufficient data points for
c:100 | *f*:100 | *m*: d

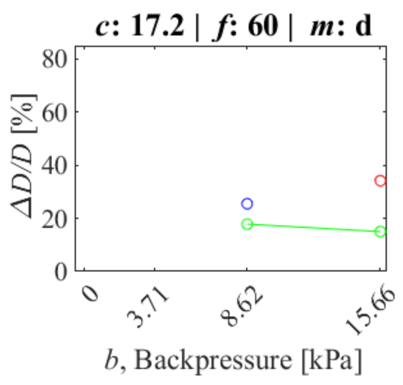


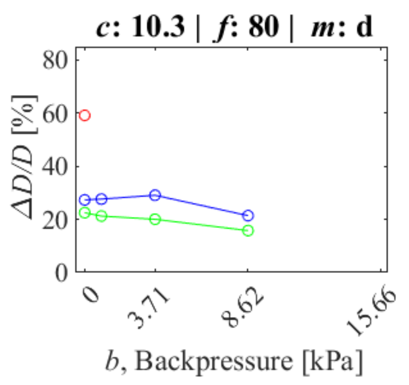


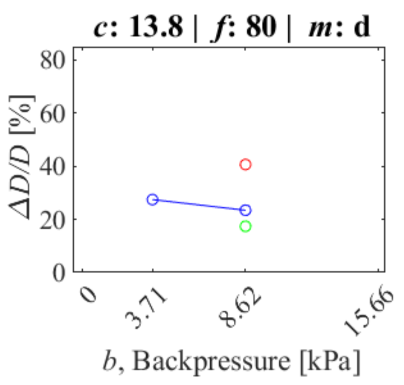


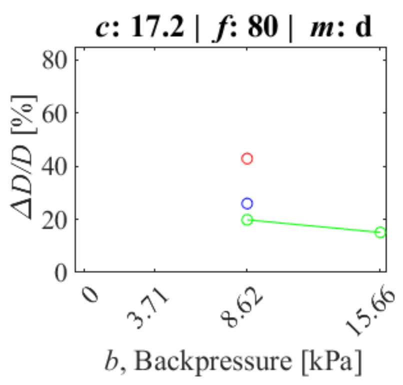


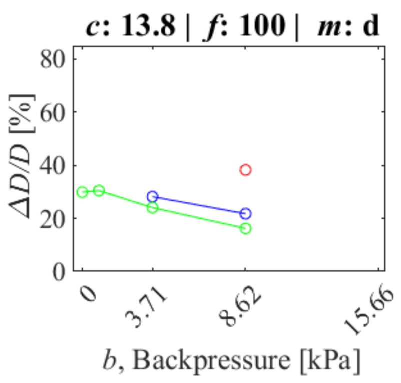


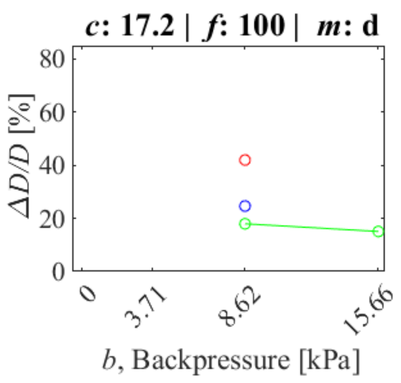


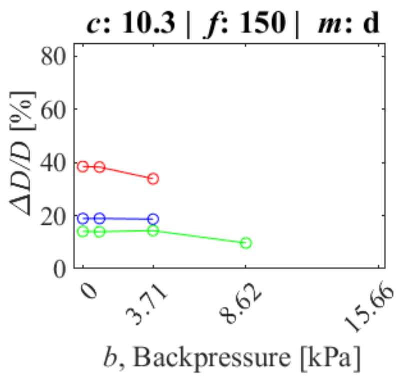


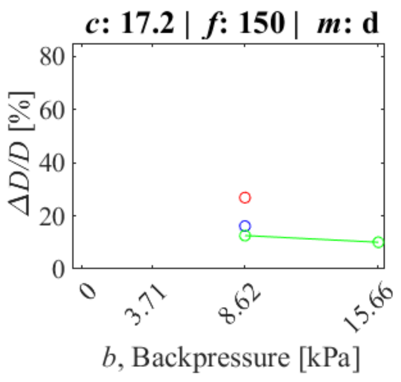


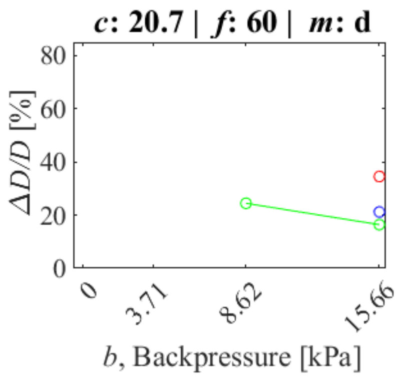


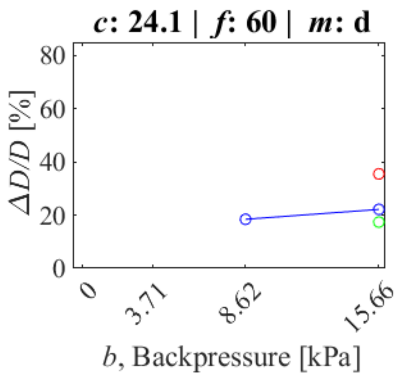

c: 13.8 | f: 100 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

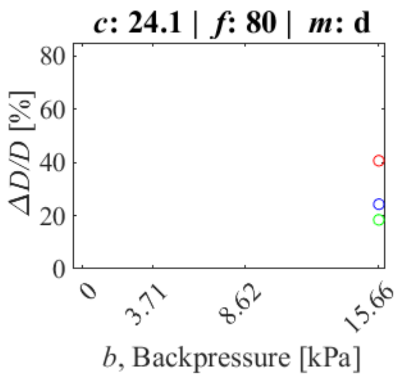

c: 17.2 | f: 100 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

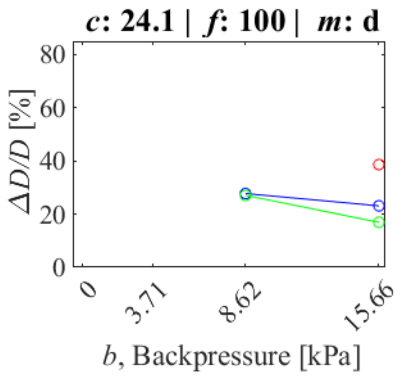

c: 10.3 | f: 150 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

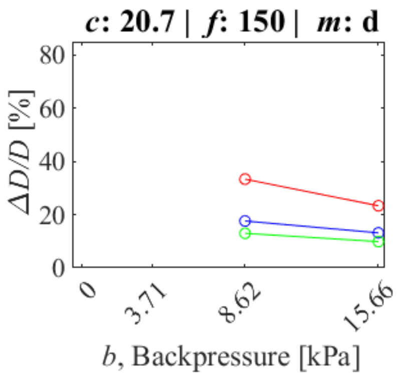

c: 17.2 | f: 150 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

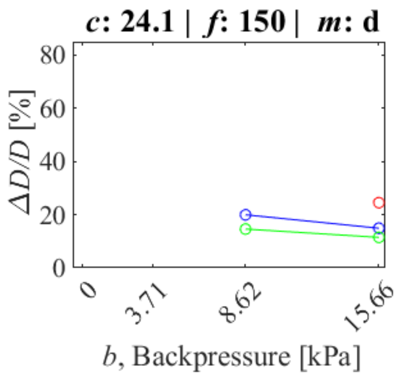

c: 20.7 | f: 60 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

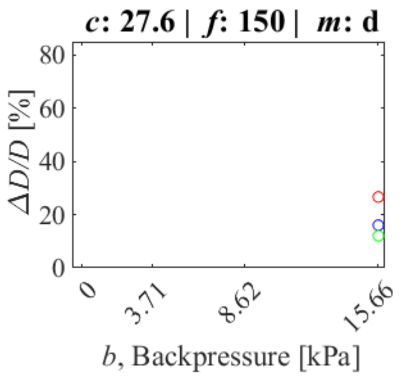

c: 24.1 | f: 60 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

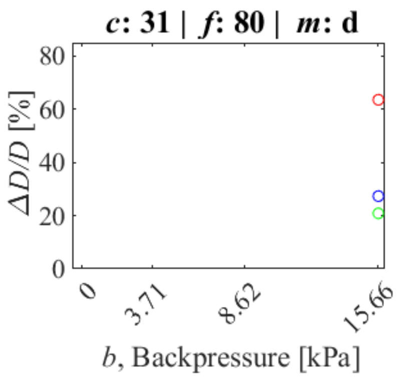

c: 24.1 | f: 80 | m: d
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

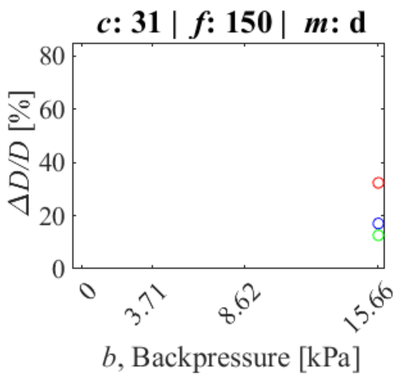

c: 24.1 | f: 100 | m: d
ΔD/D [%]
b, Backpressure [kPa]
0
3.71
8.62
15.66

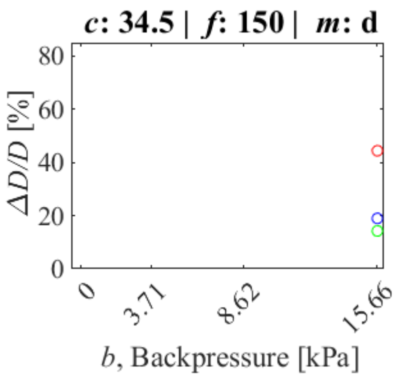

c: 20.7 | f: 150 | m: d
ΔD/D [%]
b, Backpressure [kPa]

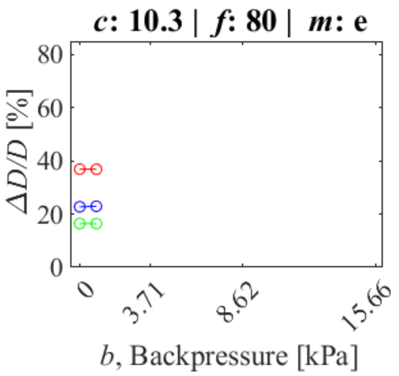

c: 24.1 | f: 150 | m: d
ΔD/D [%]
b, Backpressure [kPa]

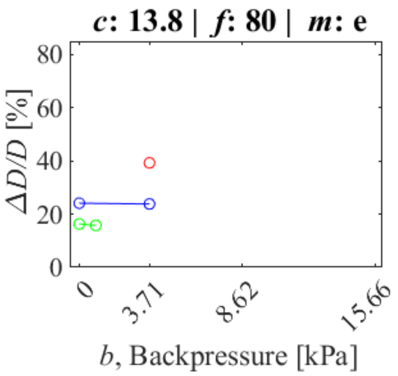

c: 27.6 | f: 150 | m: d
ΔD/D [%]
b, Backpressure [kPa]

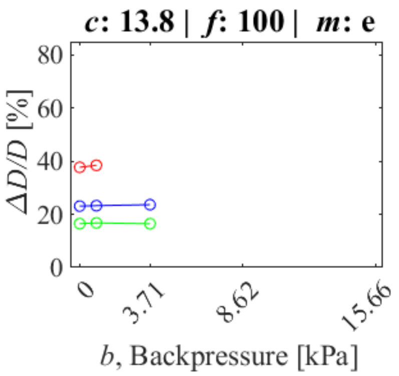

c: 31 | f: 80 | m: d
ΔD/D [%]
b, Backpressure [kPa]

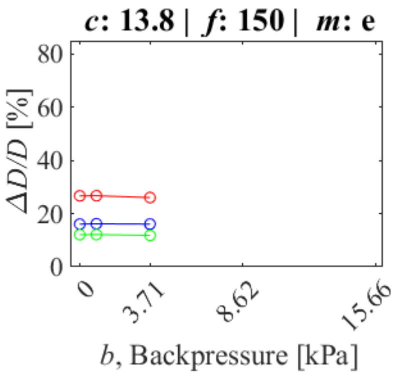

c: 31 | f: 150 | m: d
ΔD/D [%]
b, Backpressure [kPa]


c: 34.5 | f: 150 | m: d
ΔD/D [%]
b, Backpressure [kPa]


c: 10.3 | f: 80 | m: e
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 13.8 | f: 80 | m: e
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 13.8 | f: 100 | m: e
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]


c: 13.8 | f: 150 | m: e
ΔD/D [%]
0
20
40
60
80
0
3.71
8.62
15.66
b, Backpressure [kPa]

Appendix P. $\frac{\Delta D}{D}$, Maximum Percent Diameter Distension Vs Aorta Length Separated by Stroke Volume

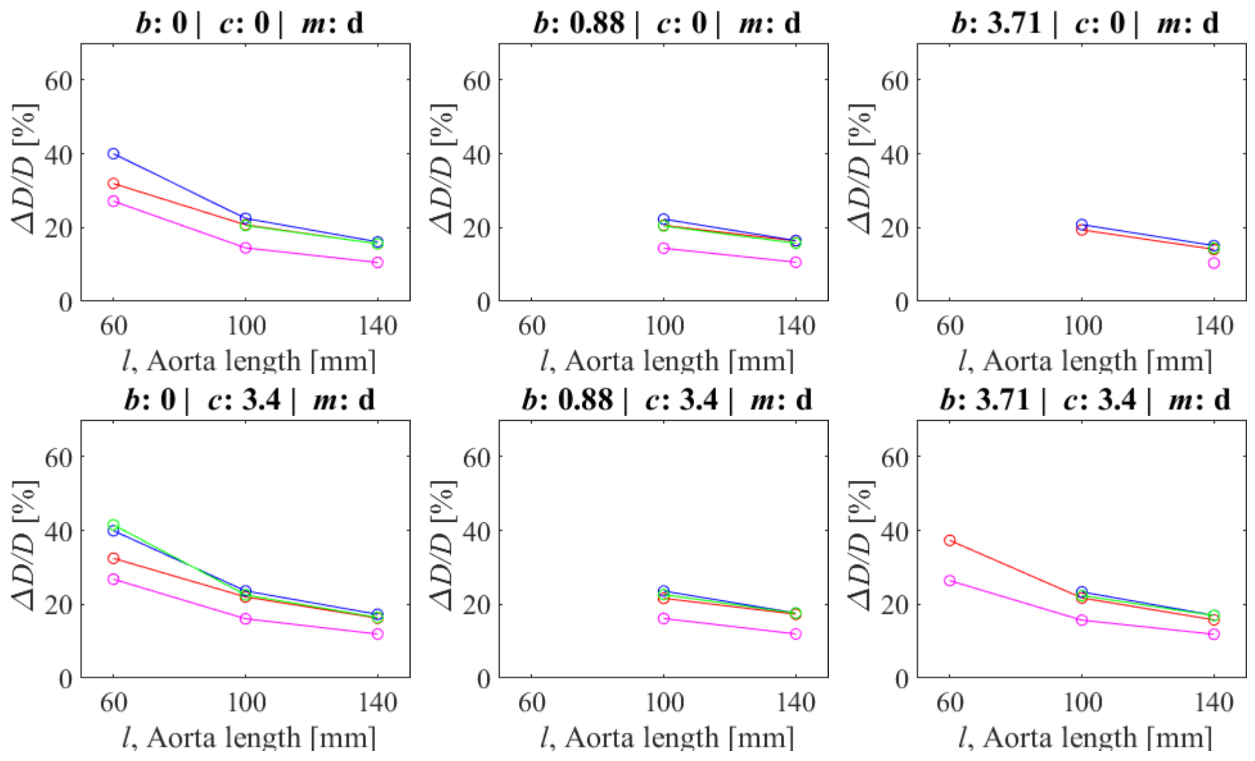
b: backpressure [kPa]

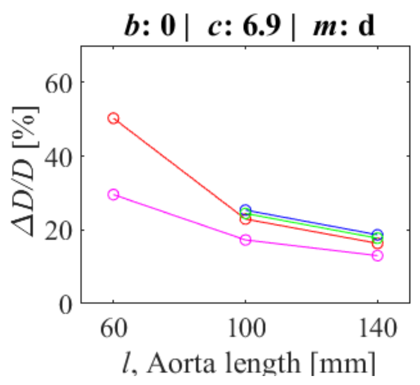
c: chamber pressure [kPa]

m: material, e for Ecoflex, d for Dragon Skin

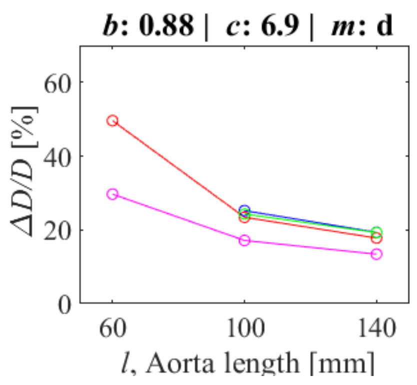
l: aorta lengths [mm] of red: 60; blue: 100; green: 140.

All empty spaces signify insufficient data points to generate plot for that case.

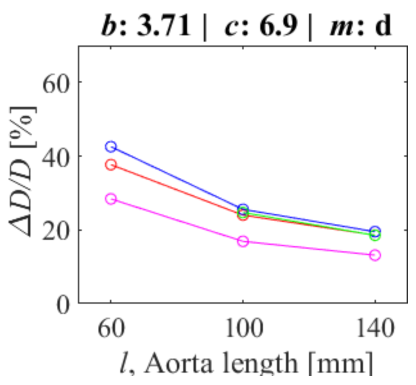




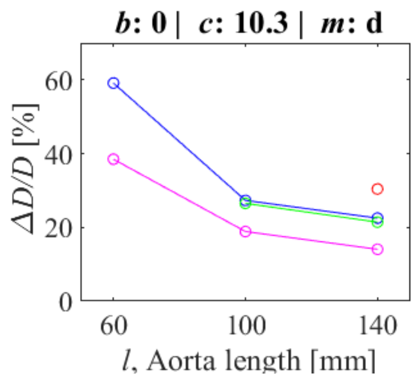
b: 0 | *c*: 6.9 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



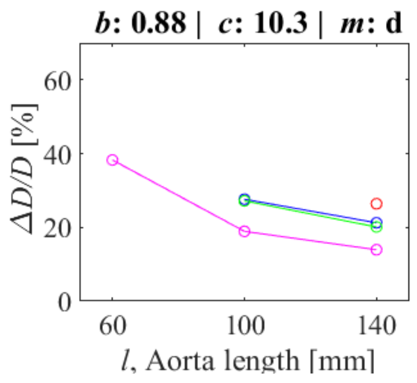
b: 0.88 | *c*: 6.9 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



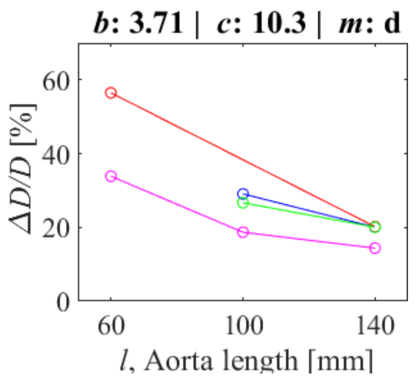
b: 3.71 | *c*: 6.9 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



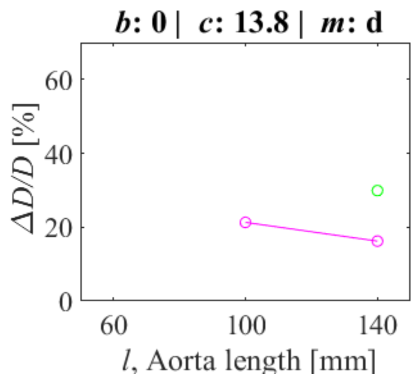
b: 0 | *c*: 10.3 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



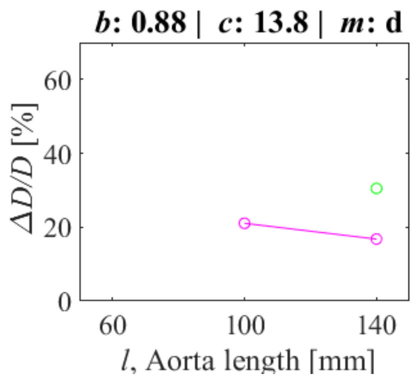
b: 0.88 | *c*: 10.3 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



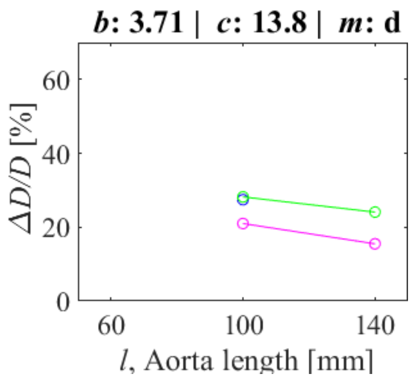
b: 3.71 | *c*: 10.3 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



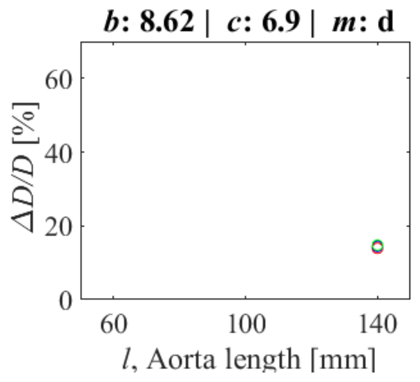
b: 0 | *c*: 13.8 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



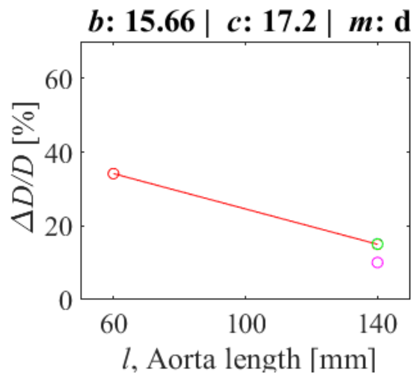
b: 0.88 | *c*: 13.8 | *m*: d
ΔD/D [%]
l, Aorta length [mm]



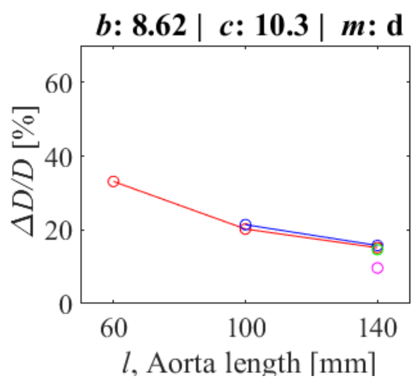
b: 3.71 | *c*: 13.8 | *m*: d
ΔD/D [%]
l, Aorta length [mm]

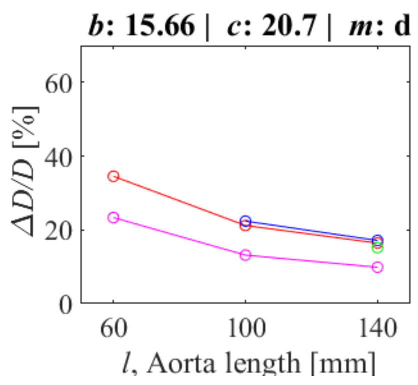


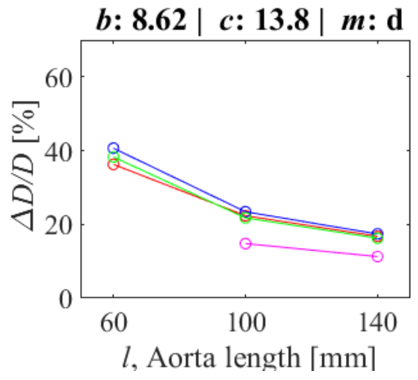
b: 8.62 | *c*: 6.9 | *m*: d
ΔD/D [%]
l, Aorta length [mm]

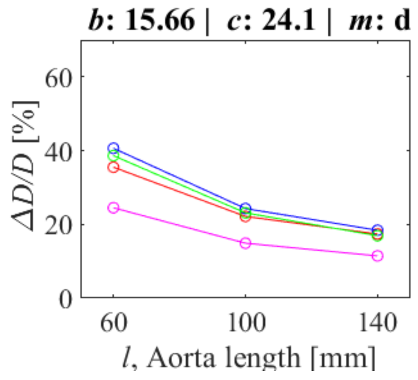


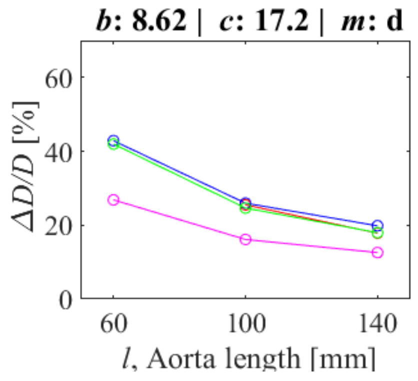
b: 15.66 | *c*: 17.2 | *m*: d
ΔD/D [%]
l, Aorta length [mm]

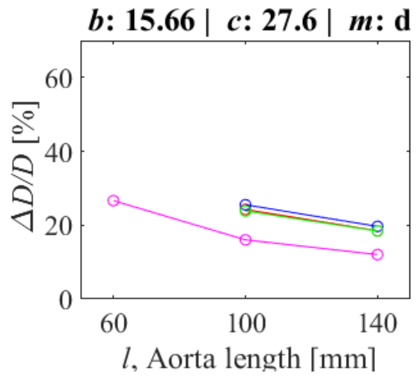

b: 8.62 | c: 10.3 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

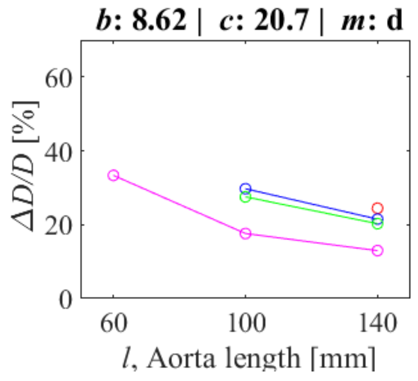

b: 15.66 | c: 20.7 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

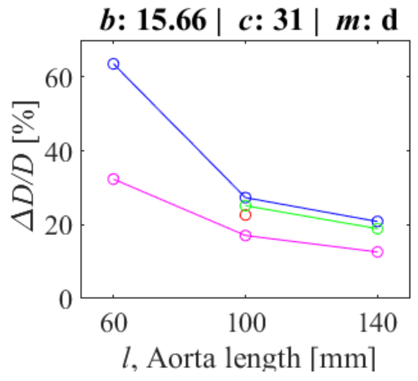

b: 8.62 | c: 13.8 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

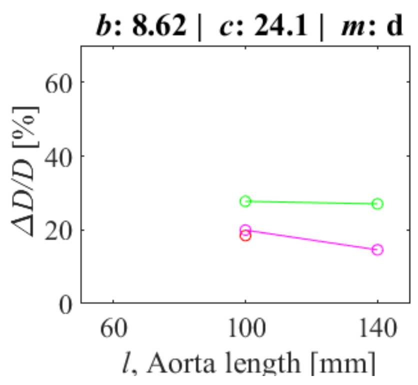

b: 15.66 | c: 24.1 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

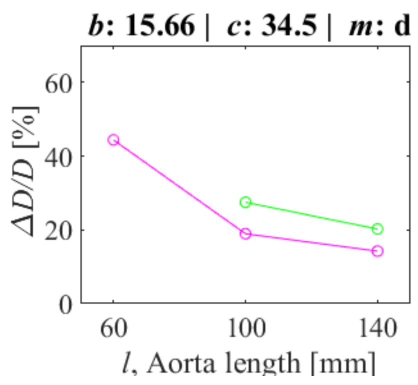

b: 8.62 | c: 17.2 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

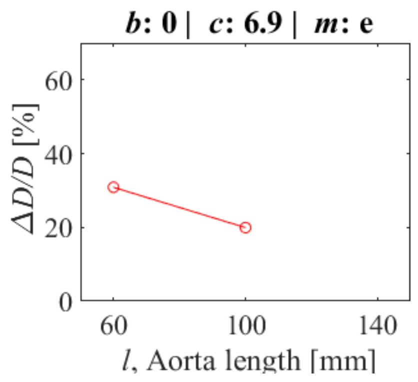

b: 15.66 | c: 27.6 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

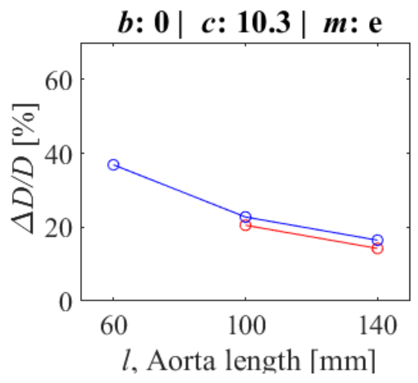

b: 8.62 | c: 20.7 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

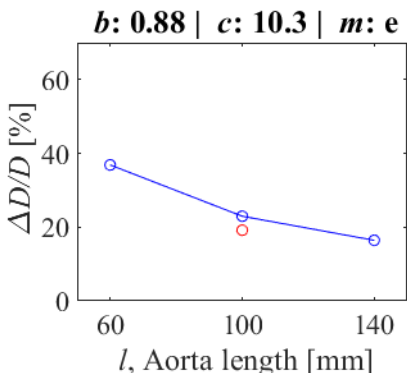

b: 15.66 | c: 31 | m: d
$\Delta D/D$ [%]
0
20
40
60
60
100
140
l, Aorta length [mm]

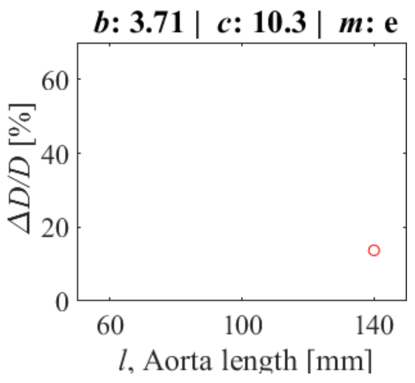

b: 8.62 | c: 24.1 | m: d
ΔD/D [%]
l, Aorta length [mm]

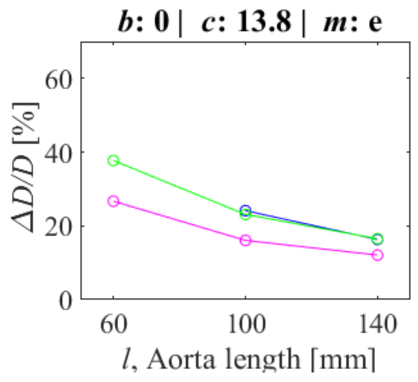

b: 15.66 | c: 34.5 | m: d
ΔD/D [%]
l, Aorta length [mm]

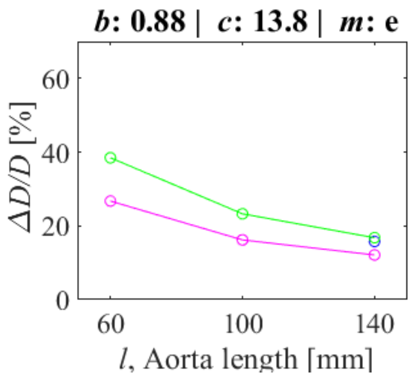

b: 0 | c: 6.9 | m: e
ΔD/D [%]
l, Aorta length [mm]

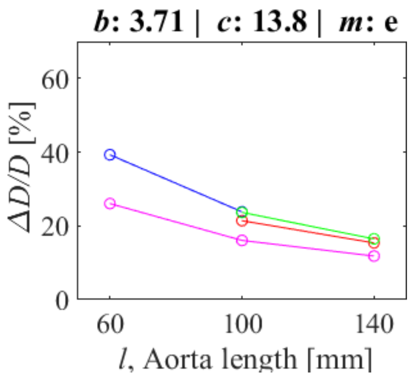

b: 0 | c: 10.3 | m: e
ΔD/D [%]
l, Aorta length [mm]

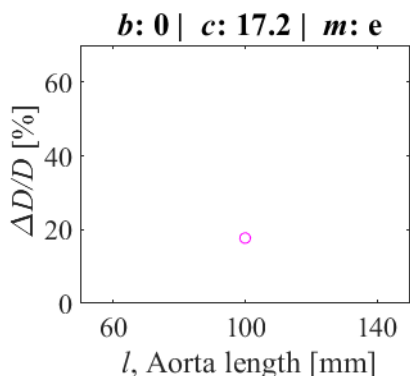

b: 0.88 | c: 10.3 | m: e
ΔD/D [%]
l, Aorta length [mm]

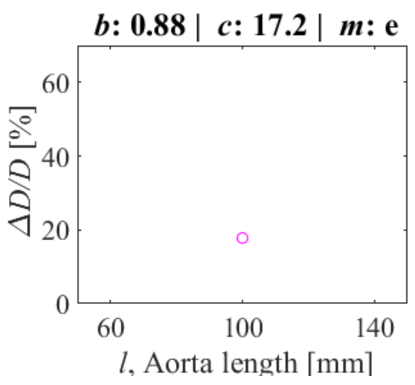

b: 3.71 | c: 10.3 | m: e
ΔD/D [%]
l, Aorta length [mm]

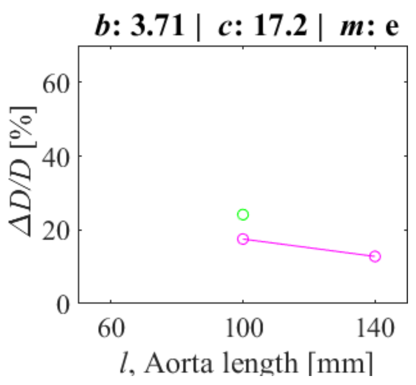

b: 0 | c: 13.8 | m: e
ΔD/D [%]
l, Aorta length [mm]

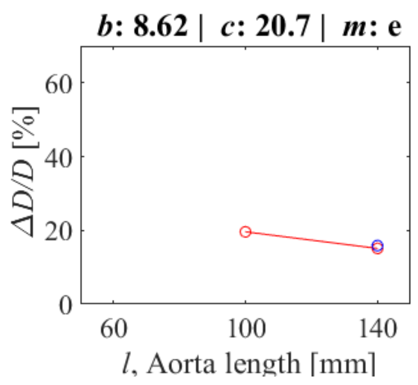

b: 0.88 | c: 13.8 | m: e
ΔD/D [%]
l, Aorta length [mm]

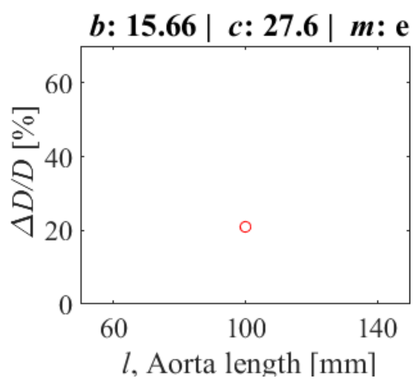

b: 3.71 | c: 13.8 | m: e
ΔD/D [%]
l, Aorta length [mm]

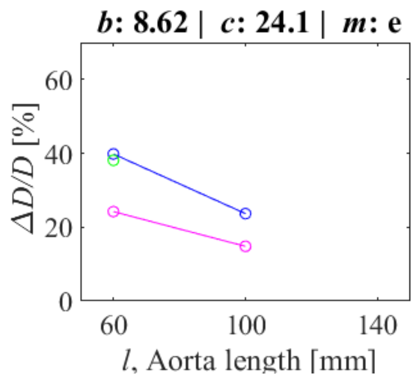

b: 0 | *c*: 17.2 | *m*: e
ΔD/D [%]
l, Aorta length [mm]

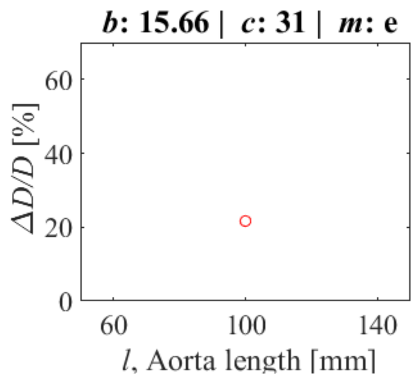

b: 0.88 | *c*: 17.2 | *m*: e
ΔD/D [%]
l, Aorta length [mm]

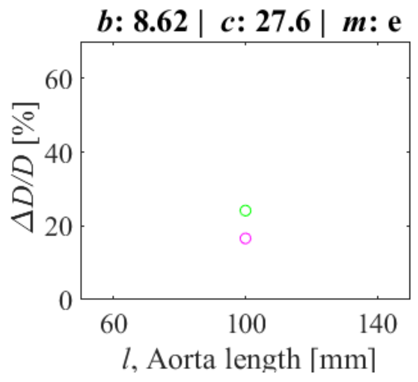

b: 3.71 | *c*: 17.2 | *m*: e
ΔD/D [%]
l, Aorta length [mm]

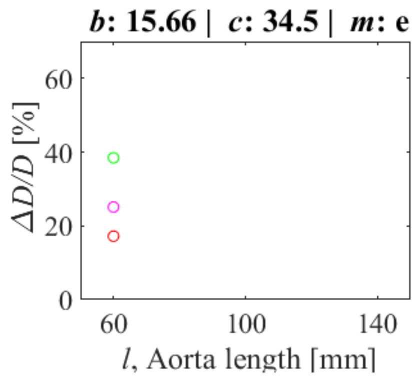

b: 8.62 | *c*: 20.7 | *m*: e
ΔD/D [%]
l, Aorta length [mm]


b: 15.66 | *c*: 27.6 | *m*: e
ΔD/D [%]
l, Aorta length [mm]


b: 8.62 | *c*: 24.1 | *m*: e
ΔD/D [%]
l, Aorta length [mm]


b: 15.66 | *c*: 31 | *m*: e
ΔD/D [%]
l, Aorta length [mm]


b: 8.62 | *c*: 27.6 | *m*: e
ΔD/D [%]
l, Aorta length [mm]


b: 15.66 | *c*: 34.5 | *m*: e
ΔD/D [%]
l, Aorta length [mm]

Appendix Q. *ΔE*, Percent Energy Change Vs $\frac{\Delta D}{D}$, Maximum Percent Diameter Distension Plots Separated by Stroke Volume

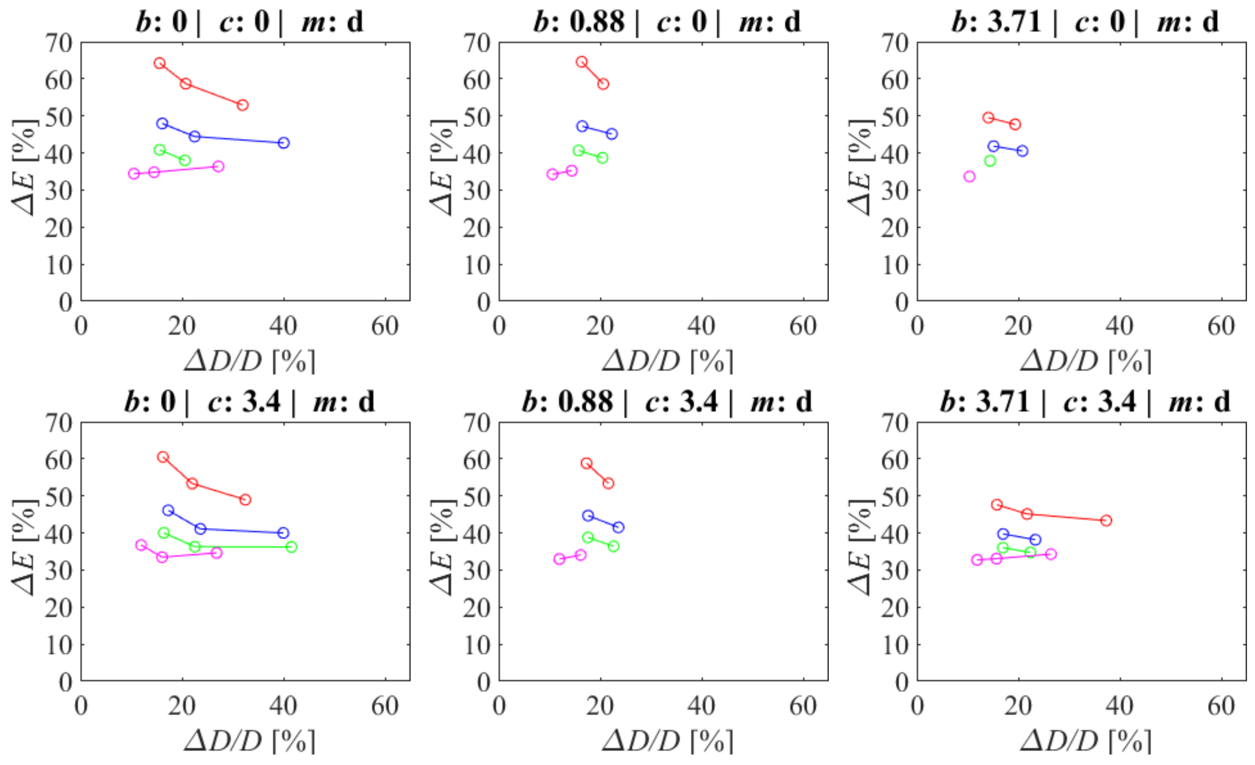
b: backpressure [rpm]

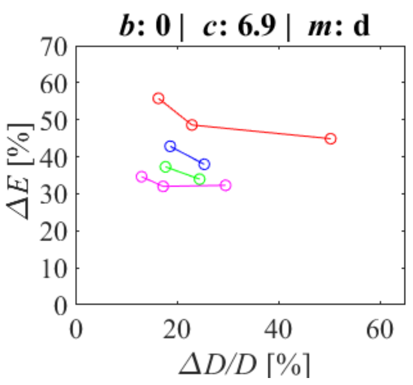
c: chamber pressure [kPa]

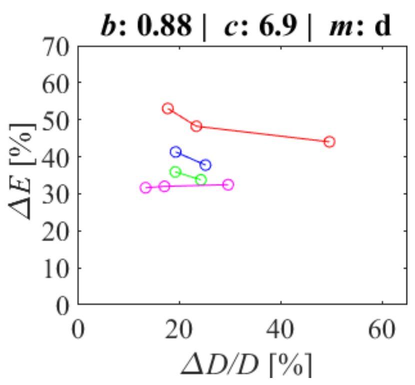
m: material, e for Ecoflex, d for Dragon Skin

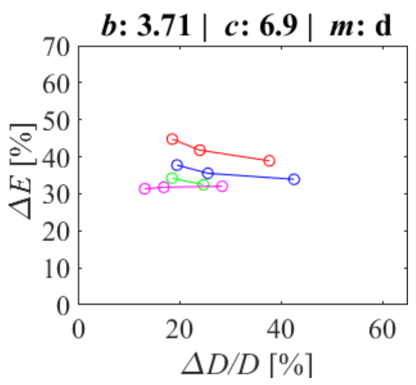
SV: stroke volumes [mL/cycle] of red: 73.6; blue: 72.5; green: 66.5; magenta: 45.4

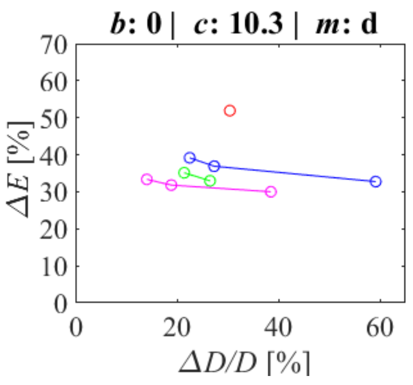
All empty spaces signify insufficient data points to generate plot for that case.

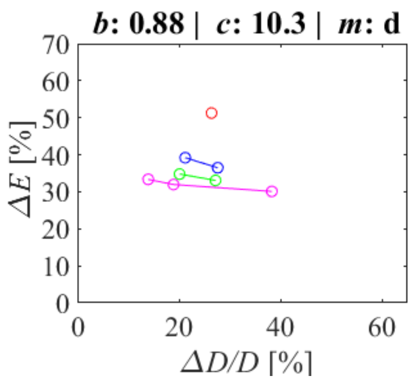


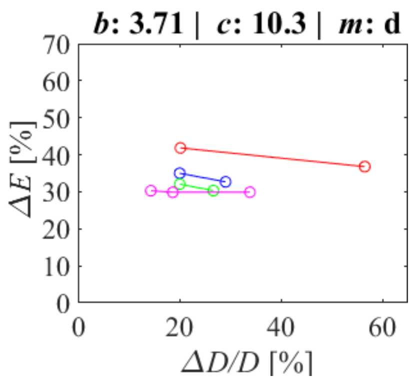

b: 0 | *c*: 6.9 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

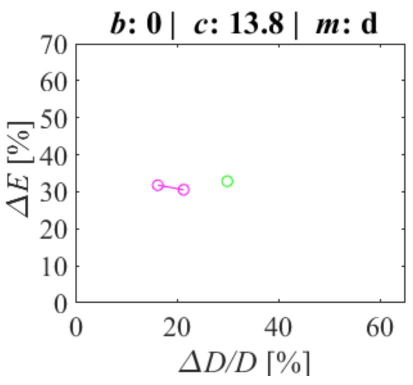

b: 0.88 | *c*: 6.9 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

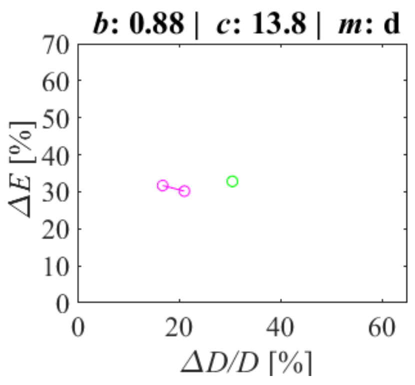

b: 3.71 | *c*: 6.9 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

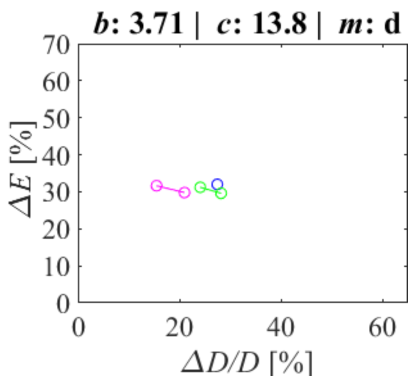

b: 0 | *c*: 10.3 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

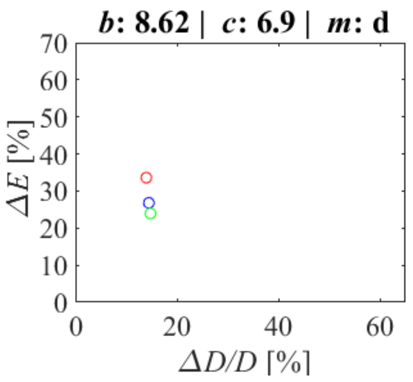

b: 0.88 | *c*: 10.3 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

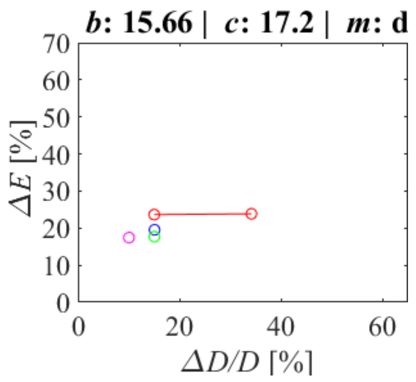

b: 3.71 | *c*: 10.3 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

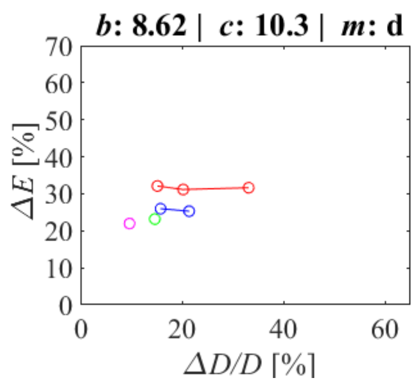

b: 0 | *c*: 13.8 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

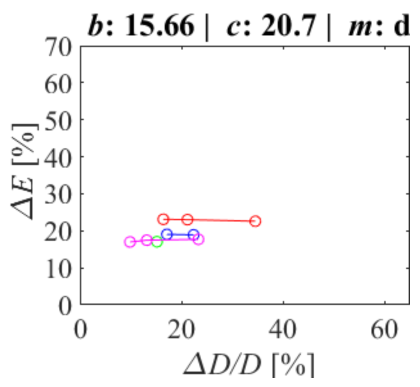

b: 0.88 | *c*: 13.8 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

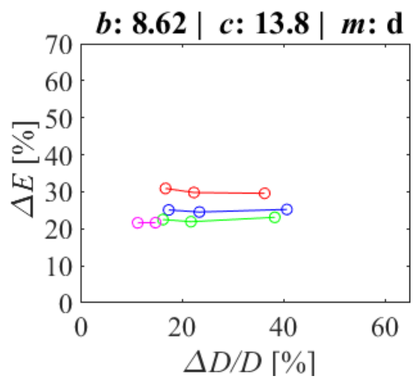

b: 3.71 | *c*: 13.8 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

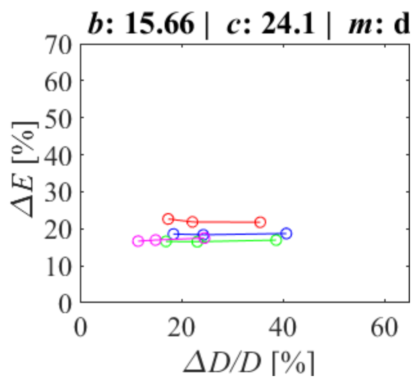

b: 8.62 | *c*: 6.9 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

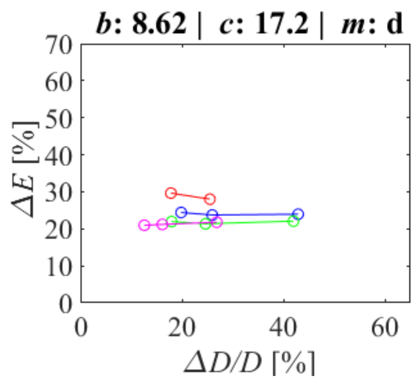

b: 15.66 | *c*: 17.2 | *m*: d
Δ*E* [%]
Δ*D*/*D* [%]
0 10 20 30 40 50 60 70
0 20 40 60

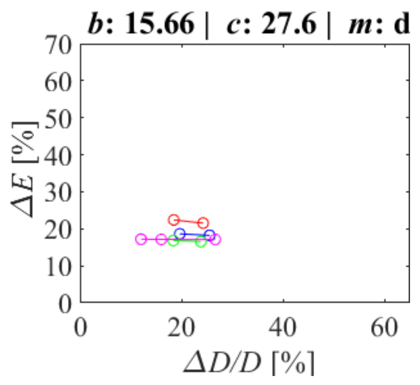

b: 8.62 | *c*: 10.3 | *m*: d
ΔE [%]
ΔD/D [%]

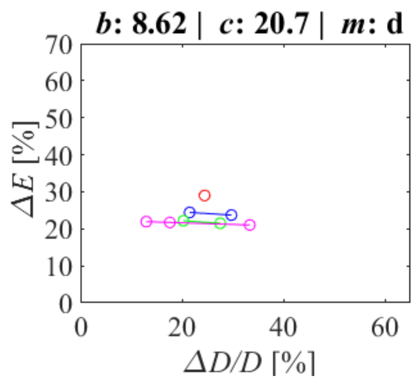

b: 15.66 | *c*: 20.7 | *m*: d
ΔE [%]
ΔD/D [%]

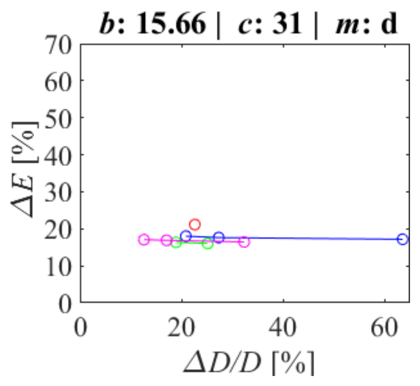

b: 8.62 | *c*: 13.8 | *m*: d
ΔE [%]
ΔD/D [%]

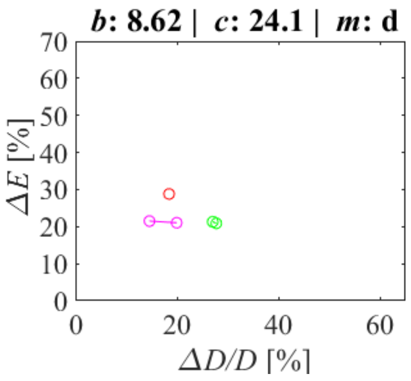

b: 15.66 | *c*: 24.1 | *m*: d
ΔE [%]
ΔD/D [%]

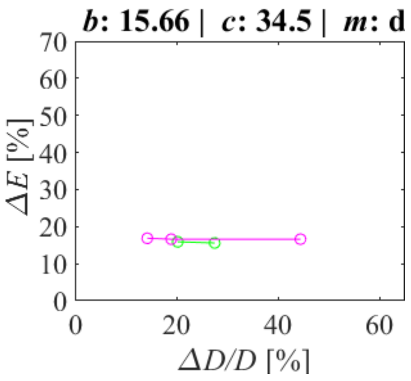

b: 8.62 | *c*: 17.2 | *m*: d
ΔE [%]
ΔD/D [%]

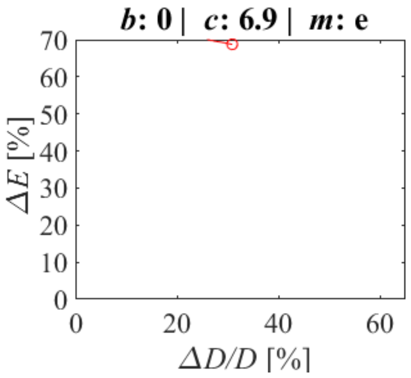

b: 15.66 | *c*: 27.6 | *m*: d
ΔE [%]
ΔD/D [%]

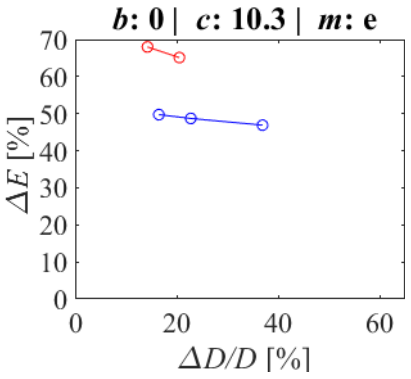

b: 8.62 | *c*: 20.7 | *m*: d
ΔE [%]
ΔD/D [%]

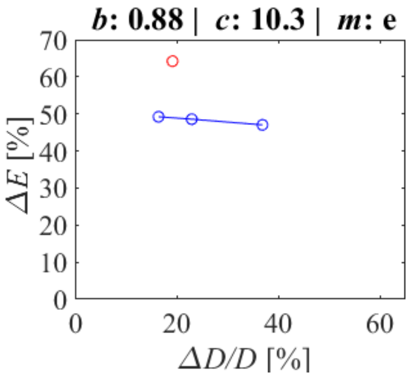

b: 15.66 | *c*: 31 | *m*: d
ΔE [%]
ΔD/D [%]

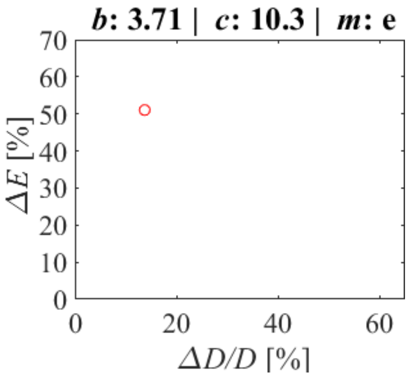

b: 8.62 | c: 24.1 | m: d
ΔE [%]
ΔD/D [%]

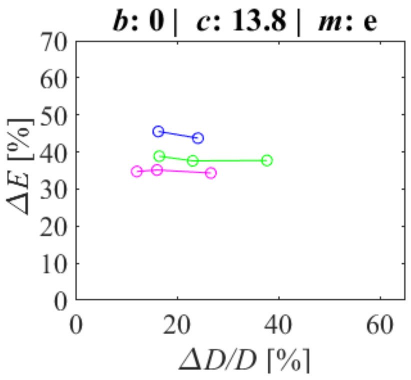

b: 15.66 | c: 34.5 | m: d
ΔE [%]
ΔD/D [%]

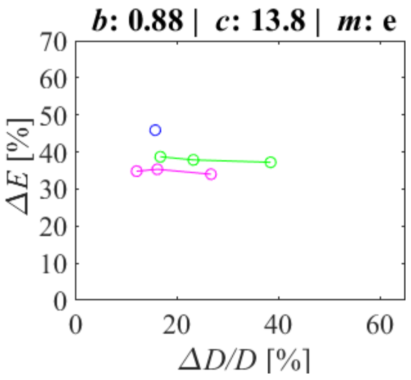

b: 0 | c: 6.9 | m: e
ΔE [%]
ΔD/D [%]

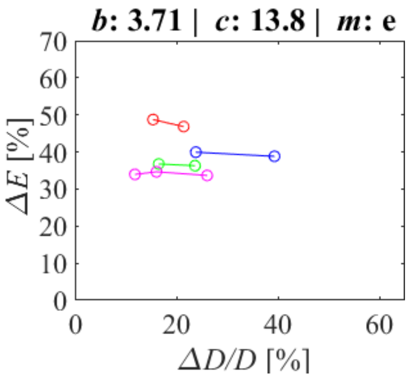

b: 0 | c: 10.3 | m: e
ΔE [%]
ΔD/D [%]

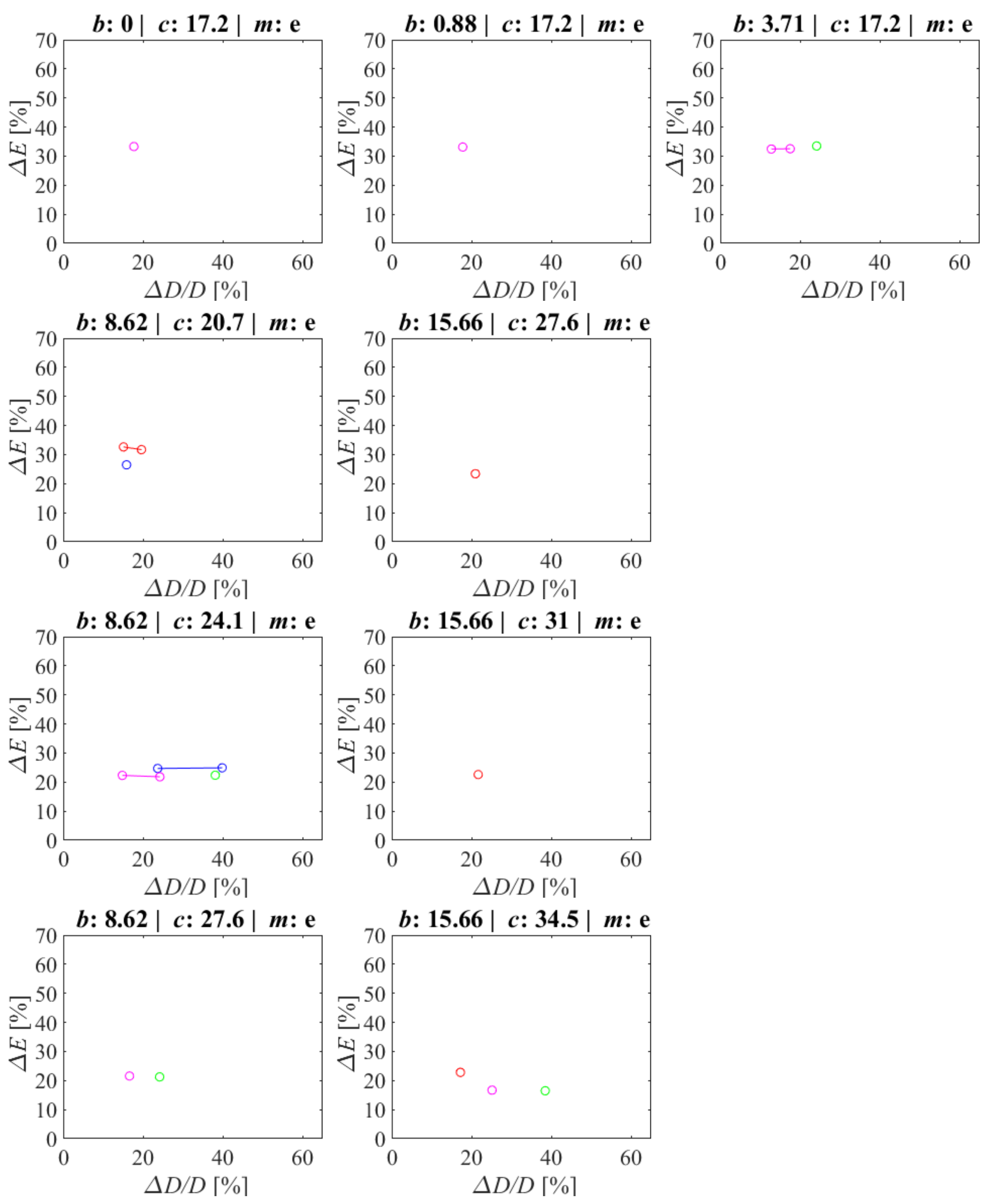

b: 0.88 | c: 10.3 | m: e
ΔE [%]
ΔD/D [%]


b: 3.71 | c: 10.3 | m: e
ΔE [%]
ΔD/D [%]


b: 0 | c: 13.8 | m: e
ΔE [%]
ΔD/D [%]


b: 0.88 | c: 13.8 | m: e
ΔE [%]
ΔD/D [%]


b: 3.71 | c: 13.8 | m: e
ΔE [%]
ΔD/D [%]


b: 0 | c: 17.2 | m: e
b: 0.88 | c: 17.2 | m: e
b: 3.71 | c: 17.2 | m: e
b: 8.62 | c: 20.7 | m: e
b: 15.66 | c: 27.6 | m: e
b: 8.62 | c: 24.1 | m: e
b: 15.66 | c: 31 | m: e
b: 8.62 | c: 27.6 | m: e
b: 15.66 | c: 34.5 | m: e
ΔE [%]
ΔD/D [%]
0
10
20
30
40
50
60
70

Appendix R. Measurement Uncertainty

Table Q.1: DAQ specifications [74]

Analogue Input (AI) resolution	14-bits differential
Maximum AI sample rate for multiple channel	48 kS/s
AI channels available	8
Timing resolution	41.67 ns (24 MHz)
Input range	± 20 V
Absolute accuracy at full scale	0.0147 V
Anti-aliasing filter	Custom designed[3]

Table Q.2: Pressure Calibrator Druck DPI 603 specifications [75]

Input Range	± 50 V
Accuracy	± 0.15% of reading ± 0.02% F.S.
Absolute accuracy at full scale	0.02 V

Table Q.3: LVDT specifications

Input Range	± 10 V
Accuracy	The LVDT calibration uncertainty is half the distance measurement device's lowest decimal value, ±0.005 mm. This corresponds to an uncertainty of approximately ±0.0026 V for measurements between -10 to 10 V

[3] Acknowledgement: Rick Conrad, Department of Mechanical Engineering