FLAT OVAL SPIRAL DUCT DEFLECTION

A Thesis

by

MATTHEW TAYLOR DAUGHERTY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Michael Pate
Committee Members,	Karl Hartwig
	Edward White
Head of Department,	Andreas Polycarpou

May 2014

Major Subject: Mechanical Engineering

ABSTRACT

The deflection of a spiral duct depends on its flat span, gauge, pressure, and reinforcement. Furthermore, the duct weight determines the cost of installation, not only in material costs but also labor, which is bid by the duct weight. The current duct deflection standard, the rectangular-duct construction standard, does not accurately represent flat-oval spiral ducts. Therefore, the objective of this project was to perform those experiments necessary to support the development of a standard for the installation of flat-oval spiral ducting. Also in support of standard development, the project developed empirical models to represent unreinforced ducts for positive and negative pressures along with T-25 ducts for positive pressures. In conclusion, the development of a flat-oval spiral duct deflection standard is important to the HVAC industry, because it would allow engineers to perform reliable duct design with a focus on reductions in material and installation cost. The collected data, empirical modeling, and data analysis that has resulted from this project will form the foundation for development of a flat-oval duct deflection standard. Specifically, the project reported herein resulted in deflection-pressure data for a broad range of unreinforced and reinforced ducts. The test matrix was designed to provide sufficient experimental data to develop empirical deflection models for unreinforced ducts and T-25 positive-pressure ducts. In addition, a more limited test matrix for other reinforced ducts allows for qualitative analysis and empirical observations. The contour plots, constructed with the aid of empirical models, can be used to identify duct sizes and types that satisfy deflection criteria at specific pressures, which is a major step in developing duct standards.

DEDICATION

To my Lord, my family, and my friends, for they are my foundation.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to those who made this project possible:

Dr. Micheal Pate - my advisor, who's guidance was priceless.

Mr. Bob Ried - SPIDA's technical director Mr. Bob Ried for arranging the project and obtaining the samples.

Eastern Sheet Metal and Spiral Pipe of Texas - for donating samples.

Mr. Joshua Kadding - the laboratory manager for supplying the resources to make this project possible.

I express my gratitude for my committee - Dr. Karl Hartwig and Dr. Edward White.

I would also like to thank the Open Source community for supplying the tools needed to complete this project.

I would also like to thank Mr. Richard Zimmer from the Bryan amateur radio club for helping with the RF interference.

A special thanks goes to the employees at the Riverside Energy Efficiency Laboratory for all their help and support with the project, especially Cameron Faucet, Jennifer Rees, James Sweeney, Kristin Vorderkunz, and Kathryn Wadle.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

NOMENCLATURE

Δh Change in Sensor Height (inches)

D Deflection (inches)

D_m Actual Displacement (inches)

E Horizontal Error (inches)

Fs Flat Span (inches)

Ga Gauge (ga)

h Height of Plumb-Bob (inches)

P Pressure (inWG)

R_m Measured Resistance ($k\Omega$)

R_T Potentiometer Resistance ($k\Omega$)

T Plumb-Bob Error (inches)

V_m Measured Voltage (Volts)

V_n Negative Voltage (Volts)

V_p Positive Voltage (Volts)

1. INTRODUCTION

The current standard used for spiral ducts is based on the rectangular duct standard and negative pressure is not allowed. Furthermore, rectangular ducts do not accurately represent flat-oval spiral duct deflections. To support the development of a spiral duct standard that is based on actual spiral duct deflections as a function of pressure a project was conceived, and the results are reported herein. The development of a standard for flat-oval spiral ducting is important for economical design because the pounds of duct required in a building determines the ducting cost for both materials and installation. Therefore, a spiral ducts standard would help reduce the pounds of metal, and hence cost, required for a building.

As a first step, a test facility consisting of a truss support system, instrumentation, a blowers system, and data acquisition and control systems was designed and built. The ducts tested in this facility were the same length, but with different flat spans and gauges. Each duct was placed in the truss support system, then their collar and end caps were mounted. Linear potentiometers were installed to measure the duct deflection. The data was collected by pressurizing or evacuating the test specimen, and this data was then used to generate pressure verses deflection graphs. Empirical models were also developed from this data to predict deflection. Of special importance for this study and standard development the experimental analysis, consisting of deflection verses flat span plots for the 0 to 2 inch deflection range as a function of gauge and pressure.

2. LITERATURE REVIEW

Many aspects were researched to complete this project. Most importantly, the standards the test was going to follow. Other information pertained to previous work, HVAC ducting standards, patents, and equipment used in the project. Each piece provides critical information for the project.

2.1 ANSI 126: Method of Testing HVAC Air Ducts and Fittings

The purpose of this document is to provide test procedures for determining pass-fail of a duct for certain criteria. The duct assembly will have end caps. The overhang of the duct on the support can be no greater than a fifth its length. This is to reduce influence on the deflection measurements. Oval ducts are to be positioned horizontal, meaning the flat side should be parallel with the ground. Dial indicators shall have jeweled bearings and 0.01 mm graduations. Dial indicator support frame may not touch the ducts. Deflection measurement points are to be in between the reinforcements on the duct. They are to be placed at 90 degree angles, located one on the top and the other on the side. [4]

2.2 RP-732

In 1991 ASHRAE funded a research project, RP-732, "Flat Oval Duct Modeling by Finite Element Analysis". The purpose of this project was to develop instillation standards by using Finite Element Analysis, backed with a few experiments. A very small amount of samples were tested, but this lacked the information needed to write a usable standard. The ducts are made by creating a round spiral duct then slipping it on mandrels, which are an inclined plane pushed together to stretch the duct into a flat oval shape. Currently the standard for installing flat oval ducts

is to follow the SMACNA Duct Construction standards. These standards do not accurately represent flat oval ducts. The purpose of RP-732 was to develop a finite element model to generate tables for the flat oval, similar to that of SMACNA's square duct construction standards. In this experiment three duct sizes and two sheet metal thicknesses were tested. This project also had two reinforcing types. Both long seam and spiral seam ducts were tested in this project. The ducts were tested both with positive and negative pressure. This project used 0.75 inch steel plate end caps, even though 20 gauge is all that is needed. However, the attachments in this experiment are not suppose to interfere with the deflection of the duct. The vertical members were spot welded to the duct, which keeps the 2nd moment of inertia deflection from being measured. Dial indicators were used to measure the defection. The test was stopped after the deflection limit reached 0.75. It is believed the spiral seams add strength to the duct. Repeatability was not tested. [25]

2.3 HVAC Duct Construction Standards

The reinforcement for flat oval duct shall be the same size and spacing as specified for a rectangular duct of the same flat span. Supports should be the same as specified for rectangular duct. Deflections shall not exceed 1/4 in on widths of 36 in and less than 1/2 in on greater widths. Ducts should be able to withstand 50% over the design pressure without plasticly deforming or failing. Duct deflections are more a product of pressure than air flow. Using a reinforcement that exceeds the minimum requirements is acceptable, but this does not allow undersizing other elements. Flat oval ducts have advantages of both round and square ducts. This means it can fit in places like a square duct, but also be joined like a round duct. Flat oval duct is for positive pressures only.

Deflection testing pressure readings should be $\pm 0.5\%$ or less. Dial indicators must

be accurate to 1% with graduations of 0.001 inches. The piping between the fan and the duct must be 2 inch diameter minimum. These test procedures are acceptable for both positive and negative pressure qualifications. Positive tests should be done first, because failure is more easily observed. The same specimen is acceptable to use for both positive and negative tests. Testers must make sure the end caps are leak-free enough to complete the test, otherwise more testing will be required. [22]

2.4 Spiral Duct Manufacturers Association (SPIDA)

Consistency in a spiral seam is critical for quality ducts; this requires well trained operators and well maintained machinery. Tolerances are required for the lockseam, material thickness, allowable deflection, metal types, and duct cleanliness. [27] Round spiral and flat oval ducts are faster to install, have longer lengths between joints, lighter weight, and are more energy efficient. [26]

2.5 Sheet Metal and Air Conditioning Contractors National Association (SMACNA)

For low pressure rectangular ducts, maximum allowed deflection for ducts over 24 inches wide was "...arbitrarily established as 0.75 inches...". Reinforcement maximum allowable deflection was "...arbitrarily established as 0.25..." inches. Seams and joints must withstand 1.5 times the operating pressure without deformation. [16] Flat oval duct combines the best of rectangular with round duct. The flat oval can fit in spaces round is unable to fit in, just like rectangular duct. Flat oval has the advantages of the joint assemblies being similar to round duct. A spiral flat oval duct is machine formed from round spiral duct. This duct can be joined with slip couplings or flanges. Flat oval ducts have less flat surfaces than rectangular ducts, reducing the need for reinforcements, because it is less susceptible to vibration. Additional reinforcing is supplied by the spiral seam. The maximum allowable deflection is 3/4

of an inch. Reinforcement helps control this deflection; however, the reinforcing is only allowed to deflect 1/4 of an inch. [15] An important factor to a duct system is dimensional stability. [20] [21] The maximum allowable deflection for Tie-Rod reinforced fibrous ducts is less than 1/100 of the span. Unpressurized ducts experience a natural sag; an anti-sag assembly can be used to prevent this. [18] Ducts can be used in many applications some; applications might be high temperature or even corrosive or abrasive. Ducts are generally used in industrial ventilation, air pollution, and dust collecting systems. Metals can corrode from chemicals and moisture, especially at higher temperatures. [17] [19] Flat oval duct's flat sides should be of the same configuration as specified for rectangular ducts; limiting the wall deflections to 3/4 inches and reinforcements to 1/4 inches. Joints and seams should be similar to round duct. Ducts shall not deform when pressurized to 50% greater than the assigned pressure class. Flat oval is a cross between a rectangular and a round duct, pulling advantages from both. Flat oval is less susceptible to vibration, because it has less flat surfaces than rectangular duct. Flat oval can be fit in tight places because of its similar aspect ratio to rectangular duct. [23]

2.6 Duct Selection and Application

All ducts are required to be sealed, but leakage testing is not required if the ducts are in the conditioned space. [1] [2] Poor duct design can cause lots of problems including added expenses, discomfort, and even adverse health effects. [3] Galvanized steel ducts are coated with zinc, which gives them resistance to the environment. The coating should not crack or flake when forming the Pittsburgh lock seam. [28]

2.7 Patents and Research

Spiral Ovalizers can employ a hydraulic boom to form the duct from a round shape to an oval shape. [11] This is a new design for a duct flange, allowing the

flange to have similar rigidity with less sheet metal and making the flange safer for the worker. The 30 inch wide specimen deflected 30% less and the 60 inch wide specimen deflected 20% less. [13] This patent develops a method and device for pressure testing pipeline. This must form a fluid tight seal for pressure testing the line. [7] The testing equipment was designed so the housing could be moved to a section of pipe being tested. This formed an annular test chamber round the pipe. [12] A method to justify structural integrity of rectangular duct during tornado-induced depressurization. Due to hand calculations not being sufficient finite element analysis (FEA) was performed. [9]

2.8 Equipment

The linear potentiometers are durably built for trouble free use and still have a high resolution. [14] The differential pressure transmitter is accurate to $\pm 0.25\%$ and outputs 4-20mA using a 4wire current loop. [10] The fan is a 1 1/2 horse power fan, which can produce 11 inches of water pressure. [6] ER70S-6 electrode wires contain high amounts of silicon and manganese making them ideal for welding steels with some rust or mill scale. This electrode can produce excellent welds with CO_2 shielding gas. [8]

2.9 Summary

Many aspects were researched to complete this project. The project was based on ANSI 126. RP-732 was a previous work on flat oval spiral duct deflection. Additionally HVAC ducting standards, patents, and equipment used in the project were researched. Information gained from this research provides critical information for the project.

3. TEST FACILITY

A test facility for testing the deflection of spiral flat-oval ducts involved the following major components and systems.

1. A Truss Structure
2. Instrumentation
3. A Blower System
4. A Data Acquisition and Control System
5. Fork Extenders

3.1 Truss Structure

A 16x16 inch truss was constructed for the backbone of the structure. The structure was built from 2x2x1/8 inch angle iron and 3/8 inch rod A36 steel. These were welded using a wire welder with ER70S-6C wire with a diameter of 0.035 inches and a 75% argon, 25% carbon dioxide shielding gas. The entire structure was 27 feet long, 12 feet wide, and 6 feet 8 inches tall. The main beam, a 16 inch warren like truss 24 feet 4 inches long, theoretically deflects only 5 hundredths of an inch with the heaviest duct. The program, written based on the second moment of inertia equations out of Roark's Formulas for Stress and Strain [29], can be seen in Appendix B.3 on pages 132-133. This structure was designed to be disassembled for relocation. The main truss was supported by 16x16 inch trusses, which can be seen in Figure 3.1. The structure's size made it possible to load ducts from either the side or the front.

Sensor frames and Unistrut were designed to be bolted to the truss. The frames supporting the linear potentiometers were built from 2x2x1/8 inch square tubing. Square tubing limited rotational freedom, while allowing the change in size of the

Figure 3.1.: Photograph of Truss System

frames. Powerful magnets held the sensors on the square tubing for adjustment during the experiment.

3.2 Instrumentation

Tape measures and potentiometers recorded deflections of the duct, while pressure transducers recorded the pressure inside the duct. The 10 inch potentiometers were setup in a voltage divider circuit and used until they hit their limit, then tape measures took over. [5] The standard required 1% accuracy, which meant a tape measure with 1/32 inch markings was acceptable for measurements over 10 inches. The tape measure was manually recorded. The pressure transducers served two purposes; first recording the pressure for the data point, and second feeding information to the blower controller. A manometer was used to manually check the pressure sensors and doubled as a pressure relief valve in emergencies. Seventy inch long plumb bobs measured the verticalness of the sensor frames. These sensors can

be seen in Figure 3.2.

When the sensors were first installed, radio frequency interference was a problem. All the sensor wires had to be removed and replaced with twisted pair, which was problematic due to having three wires run to each sensor. The linear potentiometer wiper was given two wires, so negative had a pair and positive also had a pair. The three wire portion was then braided. This modification greatly reduced the radio frequency interference, but the pressure transducers still had problems. Capacitors placed across the terminals filtered the interference even more. The capacitor's reactance shorts the Radio Frequency (RF) to ground without shorting the DC sensor signal to ground.

3.3 Blower System

The blower system pressurized the duct during positive pressure tests. The blower system evacuated the duct during negative pressure tests. Figure 3.3b and 3.3c show a schematic of each configuration, while Figure 3.3a shows a picture of the actual setup. At full shutoff the blower could reach 10 inches of positive or negative pressure. A Variable Frequency Drive (VFD) controlled the blower system to reach the desired pressures.

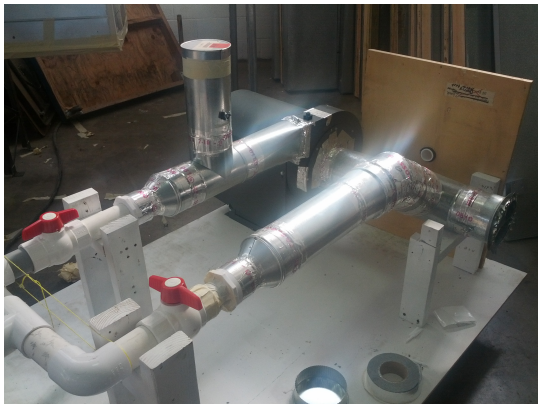
(a) 10 inch Linear Potentiometers

(b) Pressure Transducers

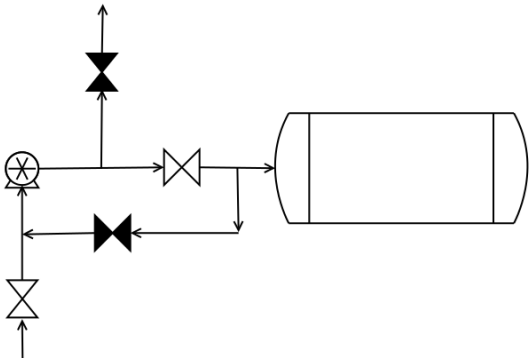
(c) Manometer

(d) Plumb Bob

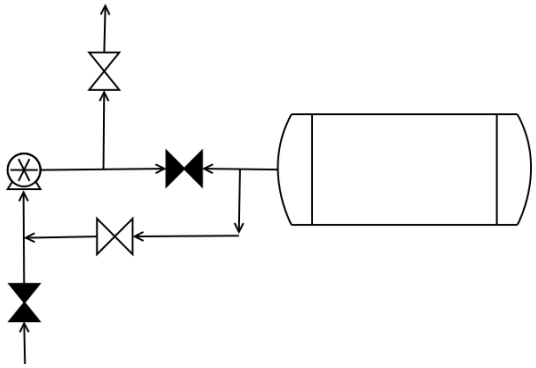
Figure 3.2.: Photographs of Sensors



(a) Blower System Photograph



(b) Blower Schematic for Positive Pressure



(c) Blower Schematic for Negative Pressure

Figure 3.3.: Blower Setup

3.4 Data Acquisition and Control System

The program used was written in LabView and can be seen in Appendix B.3 on page 129. It provided two functions, recording the data and controlling the pressure. At each pressure, sensor recordings were taken every second for two minutes and exported to a Comma Seperated Value (CSV) file. Any programming language could parse a CSV file, which gave many options for data analysis. A Porporitonal Intragal Derivative (PID) controller controlled the pressure using the setup in 3.3. The derivative component was set to zero making this actually a PI controller. The difference in duct sizes required the control system to be slow for stability.

3.5 Fork Extenders

The duct pieces were 12 feet long, which was cumbersome to handle by hand. Fork lift extensions were manufactured from 1/2 inch plywood and 2 inch by 6 inch by 10 foot boards. The 2x6 boards were ripped on a table saw to 4 1/2 inches wide to fit over 4 inch wide forks. Polyurethane adhesive and drywall screws bonded the wood together. These were light enough for one person to carry, yet strong enough to hold even the heaviest duct.

3.6 Summary

The test facility for testing deflection of flat-oval spiral ducts was constructed. This consisted of a structure to hold the duct, sensors to measure the deflection, and a system to pressurize the duct. The structure was large enough and stout enough to accept the required ducts. The sensor frames bolted solidly to this structure, so the tape measures and linear potentiometers could accurately measure deflection. Pressure transducers were used to measure the pressure inside the duct, while the blower system pressurized the duct. A data acquisition program recorded the data

and controlled the pressure using the blower system.

4. UNCERTAINTY ANALYSIS

Figure 4.3a shows the degrees of freedom causing error on the side potentiometer sensors. The linear potentiometers were calibrated using a dial indicator mounted on a steel adjustable platform for a CNC milling machine, as shown in Figure 4.1. Table 4.1 lists the variables, values, and errors of each parameter needed to derive the uncertainty of the linear potentiometers and pressure transducer. The assumption was made that no amperage flowed through the volt meter, because the meter's resistance was greater than $10G\Omega$. This assumption allowed $I_T = I_m$. Ohm's law was used across R_T and across R_m in Figure 4.2. The amperage flow was equal through equations, due to no flow to the meter. This result shows in Equation 4.20 a deflection greater than 1 in was required to stay within the SMACNA standard of 1%. The pressure transducers output 4 to 20 mA over a range of 0 to 10 inWG. These were powered by 20 V with an error of 6.22 mV. A 200 Ω resistor with an error of 4 Ω converted the sensor output to voltage. Equation 4.27 used Ohms law and Kline-McClintock to determine the pressure transducer error was 0.025 inWG.

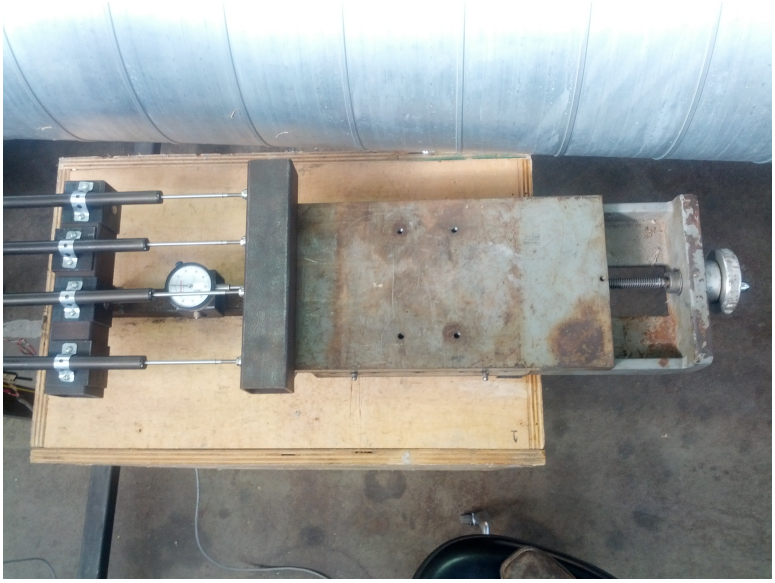


Figure 4.1.: Linear Potentiometer Calibration

Table 4.1.: Variables and Values for Displacement Sensors

Description	Variable	Value	Error
Measured Voltage	V_m	$10V$	$0.00622V$
Positive Voltage	V_p	$10V$	$0.00622V$
Negative Voltage	V_n	$-10V$	$0.00622V$
Potentiometer Resistance	R_T	$10k\Omega$	$0.008k\Omega$
Change in Sensor Height	Δh	$10in$	$1in$
Height of Plumb Bob	h	$66in$	$1in$
Plumb Bob Error	T	$0in$	$0.03in$
Horizontal Error	E	$0in$	$0.01in$
Measured Resistance	R_m	$10k\Omega$	
Actual Displacement	D_m	10 in	

4.1 Equation Derivation

$$V_{T\prime} = IR_T \qquad V_{m\prime} = IR_m$$
$$\frac{V_{m\prime}}{R_m} = \frac{V_{T\prime}}{R_T} \rightarrow R_m = \frac{V_{m\prime}R_T}{V_{T\prime}}$$
$$\boxed{R_m = \frac{R_T(V_m + V_n)}{V_p - V_n}[k\Omega]} \tag{4.1}$$

Equation 4.2 came from similar triangles in Figure 4.3b.

$$\frac{T}{h} = \frac{\Delta T}{\Delta h} \tag{4.2}$$
$$\boxed{\Delta T = \frac{T\Delta h}{h}[\text{in}]} \tag{4.3}$$

Equation 4.4 came from Pythagorean theorem of the triangle in Figure 4.3c.

$$R_m^2 = M^2 + E^2 \tag{4.4}$$
$$\boxed{M = \sqrt{R_m^2 - E^2}} \tag{4.5}$$

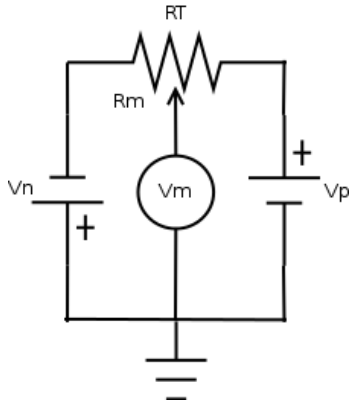
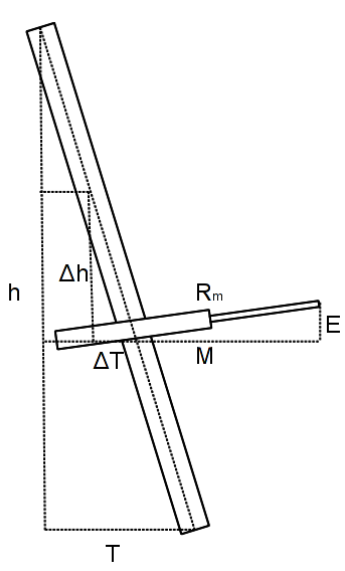
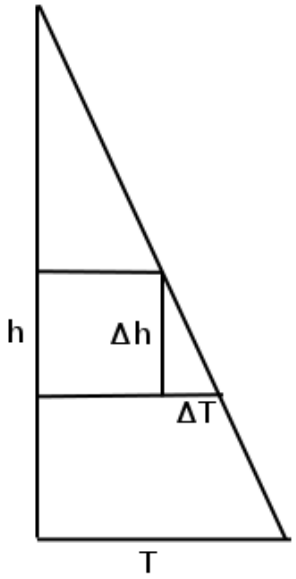


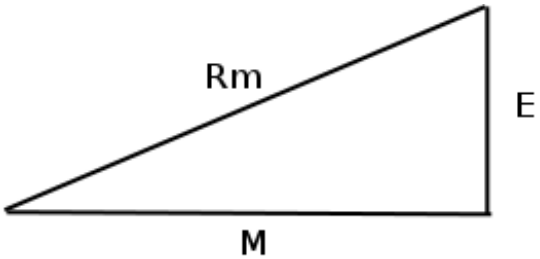
Figure 4.2.: Linear Potentiometer Schematic



(a) Sensor Frame Error



(b) Sensor Frame Vertical Error



(c) Sensor Frame Horizontal Error

Figure 4.3.: Sensor Frame Uncertainty

Equations (4.1) and (4.3) were combined to find the total deflection measured by the potentiometer (D_m).

$$\begin{aligned} D_m &= M \pm \Delta T \\ &\boxed{= \sqrt{R_m^2 - E^2} \pm \frac{T\Delta h}{h}[\text{in}]} \end{aligned} \tag{4.6}$$

E=0 and T=0

$$\begin{aligned} D_m &= \sqrt{R_m^2 - \cancel{E^2}^{\,0}} \pm \frac{T\cancel{\Delta h}^{\,0}}{\cancel{h}} \\ &\boxed{= R_m - \frac{R_T(V_m + V_n)}{V_p - V_n}} \end{aligned} \tag{4.7}$$

Kline-McClintock was performed on Equation 4.7.

$$w_{R_m} = \sqrt{\left(w_{V_m}\frac{\partial R_m}{\partial V_m}\right)^2 + \left(w_{R_T}\frac{\partial R_m}{\partial R_T}\right)^2 + \left(w_{V_p}\frac{\partial R_m}{\partial V_p}\right)^2 + \left(w_{V_n}\frac{\partial R_m}{\partial V_n}\right)^2} \tag{4.8}$$

Equations 4.9 and 4.10 are partial derivatives for each variable.

$$\frac{\partial R_m}{\partial V_m} = \frac{R_T}{V_p - V_n} \qquad \frac{\partial R_m}{\partial R_T} = \frac{V_m + V_n}{V_p - V_n} \tag{4.9}$$

$$\frac{\partial R_m}{\partial V_p} = -\frac{R_T(V_p + V_m)}{(V_p + V_n)^2} \qquad \frac{\partial R_m}{\partial V_n} = -\frac{R_T(V_m - V_n)}{(V_p + V_n)^2} \tag{4.10}$$

$$w_{R_m} = \sqrt{\begin{array}{l}\left(w_{V_m}\dfrac{R_T}{V_p - V_n}\right)^2 + \left(w_{R_T}\dfrac{V_m + V_n}{V_p - V_n}\right)^2 + \left(-w_{V_p}\dfrac{R_T(V_p + V_m)}{(V_p + V_n)^2}\right)^2 \\ \qquad + \left(-w_{V_n}\dfrac{R_T(V_m - V_n)}{(V_p + V_n)^2}\right)^2\end{array}} \tag{4.11}$$

$$= \sqrt{\begin{array}{l}\left(0.00622\dfrac{10}{10+10}\right)^2 + \left(0.008\dfrac{10+10}{(10+10)}\right)^2 + \left(0.00622\dfrac{10(10+10)}{(10+10)^2}\right)^2 \\ \qquad + \left(0.00622\dfrac{10(10+10)}{(10+10)^2}\right)^2\end{array}} \tag{4.12}$$

$$\boxed{= 0.010[\text{in}]} \tag{4.13}$$

$$w_{D_m} = \sqrt{\left(\frac{\partial D_m}{\partial R_m}\right)^2 + \left(\frac{\partial D_m}{\partial E}\right)^2 + \left(w_{w_T}\frac{\partial D_m}{\partial T}\right)^2 + \left(w_{\Delta h}\frac{\partial D_m}{\partial \Delta h}\right)^2 + \left(w_h\frac{\partial D_m}{\partial h}\right)^2} \tag{4.14}$$

$$\frac{\partial D_m}{\partial R_m} = \frac{R_m}{\sqrt{R_m^2 - E^2}} \qquad \frac{\partial D_m}{\partial E} = \frac{-E}{\sqrt{R_m^2 - E^2}} \tag{4.15}$$

$$\frac{\partial D_m}{\partial h} = -\frac{\Delta h T}{h^2} \qquad \frac{\partial D_m}{\partial T} = \frac{\Delta h}{h} \tag{4.16}$$

$$\frac{\partial D_m}{\partial \Delta h} = \frac{T}{h} \tag{4.17}$$

$$w_{D_m} = \sqrt{\left(w_{R_m}\frac{R_m}{\sqrt{R_m^2-E^2}}\right)^2+\left(w_E\frac{-E}{\sqrt{R_m^2-E^2}}\right)^2+\left(w_T\frac{\Delta h}{h}\right)^2 + \left(w_{\Delta h}\frac{T}{h}\right)^2+\left(w_h\frac{T\Delta h}{h^2}\right)^2} \tag{4.18}$$

$$= \sqrt{\left(0.010\frac{10}{\sqrt{10^2-0^2}}\right)^2+\left(0.01\frac{-0}{\sqrt{10^2-0^2}}\right)^2+\left(0.03\frac{10}{66}\right)^2 + \left(1\frac{0}{66}\right)^2+\left(1\frac{10*0}{66^2}\right)^2} \tag{4.19}$$

$$\boxed{D_m = 0.011[\text{in}]} \tag{4.20}$$

$$I = 1.6P + 4 \tag{4.21}$$

$$V = IR = 1.6PR + 4R \tag{4.22}$$

$$P = \left(\frac{V}{R} - 4\right)\frac{1}{1.6} \tag{4.23}$$

$$\frac{\partial P}{\partial V} = \frac{1}{1.6R} \qquad \frac{\partial P}{\partial R} = -\frac{V}{1.6R^2} \qquad \frac{\partial P}{\partial P} = 1 \tag{4.24}$$

$$w_P = \sqrt{\left(w_v\frac{\partial P}{\partial V}\right)^2+\left(w_R\frac{\partial P}{\partial R}\right)^2+\left(w_I\frac{\partial P}{\partial P}\right)^2} \tag{4.25}$$

$$= \sqrt{\left(w_v\frac{1}{1.6R}\right)^2+\left(w_r\frac{I}{1.6R}\right)^2+(0.025(1))^2} \tag{4.26}$$

$$\boxed{= 0.025\ [\text{inWG}]} \tag{4.27}$$

4.2 Summary

The uncertainty for the deflection measurements came from the degrees of freedom of the sensor frames and the sensor error. The linear potentiometers were calibrated using a steel adjustable platform for a CNC milling machine and a dial

indicator. The potentiometers were shown to be accurate to 0.011 inch. The pressure transducer had an uncertainty of 0.025 inWG The uncertainty fit within the test requirements.

5. DUCT TEST SPECIMENS AND TEST MATRIX

A test matrix of the 64 ducts can be seen in Table 5.1 and 5.2 on page 24. These ducts varied in sizes and reinforcement. The ducts were split between positive and negative tests.

5.1 Duct Sizes

All the ducts were 24 feet long with varying flat spans and heights. There were three heights 4, 16, and 30 inches. The 4 and 30 inch samples were tested to confirm if the flat span designation used on rectangular duct, worked on spiral flat-oval ducts. The flat spans, gauges, and reinforcements were compared using the 16 in height. The flat span ranged from 6 in to 63 in. Square duct is compared by this designation, but this comparison does not account for the half circles; therefore, this makes the flat-oval ducts actually wider with a larger cross sectional area.

The ducts varied in sheet metal thickness from 18 to 26 gauge. Galvanized sheet metal with a thickness of 18 gauge was 0.516 in thick and 26 gauge was 0.0217 in thick, which showed 18 gauge is twice as thick as 26 gauge. This, by itself, affected the deflection, but the flat-oval duct seams had four layers of sheet metal, which had an even greater affect on deflection.

5.2 Duct Reinforcements

The nine different reinforcement types are listed in Table 5.3. A slip in collar connected the two duct halves at the joint for unreinforced ducts. T-25 reinforcement used a transverse connector at the joint. The remaining reinforcements connected the joint using slip in collars, but had additional support. Attached reinforcement had angle iron welded to the top and bottom of the duct for support. Trapeze

reinforcement was supported by two pieces of Unistrut bolted together on the duct's top and bottom. Tie rod reinforcement had an all thread through the duct with a piece of conduit on the inside to hold the duct at the proper spacing.

5.3 Summary

The test matrix consisted of 64 ducts with all of the ducts being 24 feet long with different flat spans and gauges. Two ducts had different heights than the rest of the samples. These were intended for validation of flat span designation. The ducts had nine different types of reinforcements. Ducts were classified as positive tests or negative tests. The ducts were spread over the test matrix to collect data more effectively.

Table 5.1.: Test Matrix Positive

Gauge	18						20						22						24						26					
Flatspan	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6
Unreinforced	T	T		T				T		T	T				T	T	T	T					T	T				T		
T-25 6ft lengths													T																	
T-25 12ft lengths										T			T			T				T				T				T		
Attached reinforcing 3ft centers													T							T				T						
Attached reinforcing 6ft centers													T											T						
Trapeze 3ft centers																				T				T						
Trapeze 6ft centers				T																T								T		
Tie Rod 3ft centers																				T										
Tie Rod 6ft centers																				T		T		T				T		

Table 5.2.: Test Matrix Negative

Gauge	18						20						22						24						26					
Flatspan	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6
Unreinforced		T		T	T		T			T	T					T		T					T			T				
T-25 6ft lengths																														
T-25 12ft lengths										T						T								T						
Attached reinforcing 3ft centers																								T						
Attached reinforcing 6ft centers																								T						
Trapeze 3ft centers																				T				T						
Trapeze 6ft centers				T																T										
Tie Rod 3ft centers													T																	
Tie Rod 6ft centers																						T		T						

Table 5.3.: Reinforcements

Designation	Reinforcement	Reinforcement Spacing	Description
1	unreinforced	24ft	A slip in collar
2	T-25	6ft	A transverse
3	T-25	12ft	connector
4	attached	3ft	A slip in collar plus
5	attached	6ft	angle iron
6	trapeze	3ft	A slip in collar plus
7	trapeze	6ft	Unistrut
8	tie rod	3ft	A slip in collar plus
9	tie rod	6ft	Tie Rod

6. MOUNTING AND TEST PROCEDURE

Each duct test required a number of steps to complete, which included the ducts had to be assembled and installed properly before the testing could began. After testing, the duct was removed to prepare for the next test.

6.1 Duct Assembly and Installation

Before testing could be performed, the specimen was assembled and installed into the structure. This procedure consisted of placing both duct sections in the structure, then installing the collar and end caps. The duct had to be accurately installed in the structure, so the sensors could be properly positioned for a test.

6.1.1 Duct Insertion into the Structure

The forklift temporarily placed the first duct section in the structure. Next, several people, or a rolling pallet, moved this duct section to its final resting place in the structure. Then, the fork lift placed the second duct section in the structure.

6.1.2 Collar Installation

The collar was first test fit in each duct to determine the smaller duct section. The tacks were then drilled out of one seam in the collar, so it could be resoled. The smaller section of the duct had the collar inserted. The duct was set on Unistrut to make the bottom flat as shown in Figure 6.1a. The collar was clamped, starting with the tacked seam and working towards the drilled seam, so the collar was tight against the duct the entire way around. The screws were placed an 8 inches apart, this was done how lug nuts on a car are tightened to reduce the chances of a fish mouth. The duct halves were worked together using a soil knife. The thinner ducts needed support from the inside when the screws were installed. This was accomplished

with a 2x4 T (Figure 6.1b). Two wraps of 3 inch wide rolled sealant was required to cover all the screw heads and the joint. A tape press tool pressed the rolled sealant firmly to the duct improving sealing. A finished joint can be seen in Appendix A on page 100.

6.1.3 End Cap Installation

Rolled sealant sealed the inside of the T-25 connectors as shown in Figure 6.2a. Figure 6.2b shows placing the gasket tape on the flange. Finally, the end cap was screwed on the flange as seen in 6.2c. Finally, the pressure hoses were placed on the taps.

6.1.4 Duct Positioning in Structure

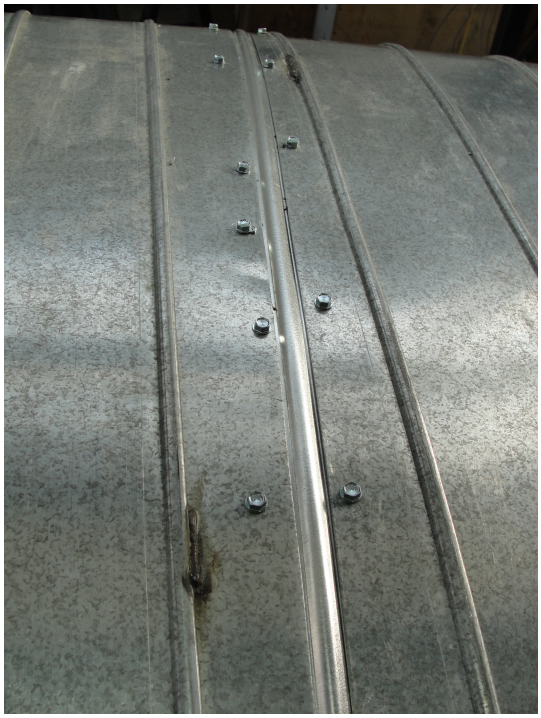
The distance between the duct and the floor was important, because the duct would hit the truss before reaching the max test pressure if positioned too high. During positive pressure tests, the duct rose because of many things. First, the duct tried to be come round causing the flat span to bow. Second, the increased second moment of inertia caused by the duct becoming more round. This caused the duct to slouch less. Third, the duct pushed off the Unistrut supports. During a negative test the second moment of inertia reduces when the pressure sucks the duct flat, so the duct slouches more.

6.1.5 Sensors Installation

The sensors measured 16 points of deflection on the duct. The sensors and tape measures were as close together as possible this can be seen in Appendix A on page 100. Their optimal placing would be on the exact same point on the duct, but far enough apart not to interfere with each other. The distance of the sensors from the duct depends on whether it was a positive or negative test. With a positive test

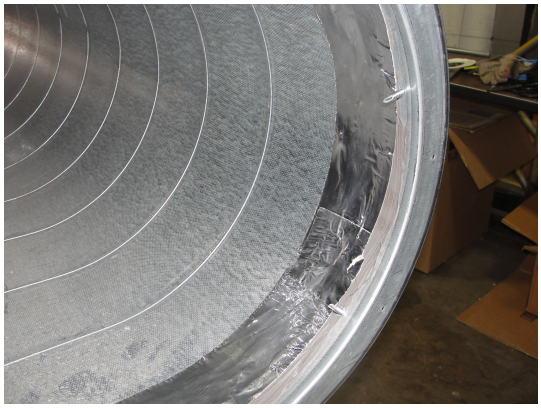
(a) Installing Coupling on First Duct Section

(b) 2x4 T Supporting Thin Duct for Assembly

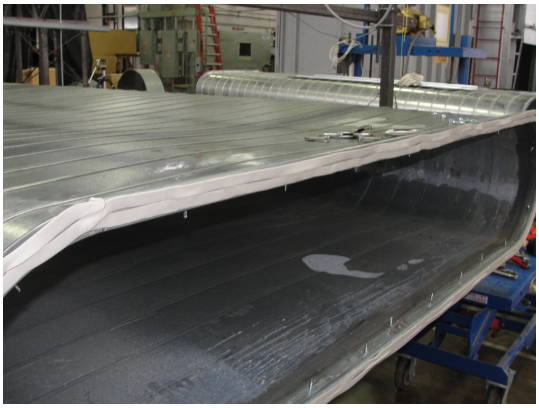


(c) Finished Duct Joint Prior to Sealing

Figure 6.1.: Joint Installation



(a) Rolled Sealant



(b) Gasket Tape

(c) Pressure Taps

Figure 6.2.: End Cap Installation

the sensors had to be extended all the way, meaning the holders had to hold the end of the potentiometer tubes, because once the potentiometers ran out of room they were removed and the test finished with the tape measures. The top swelled out and the sides of the duct sucked in, so the side sensors had to be nearly bottomed out. With a negative pressure test position the vertical sensors where placed with only 1/2 inch of play before they bottomed out and the side where extended all the way. If the sensors bottomed out they could have been damaged.

6.2 Test Procedure

The computer and Variable Frequency Drive (VFD) were first turned on. Then the operator printed off the tape measure template and started the data recording program. This program is in Appendix B.3 on page 129. The operator zeroed the pressure transducers and recorded the frame spacings. The computer recorded the zero pressure reading, while the operator manually recorded the tape measures. The operator enabled the autopilot switch to activate the Porporitonal Intragal Derivative (PID) controller and typed the desired pressure into the program. When the duct reached the desired pressure the operator recorded another data point. The PID controller had overshoot; therefore, incremental steps were taken towards the desired pressure. Once all the data points were recorded, the operator recorded the hysteresis to determine if the duct could be used for another test. Finally, the computer and VFD were off.

6.3 Duct Disassembly

The sensors were removed and placed in a safe place. Next, the end caps were removed and the duct was broke in half at the joint. The forklift removed the first section. The several people or a rolling pallet carried the second duct section to the fork lift for removal.

6.4 Summary

Testing a single specimen required a number of steps, which included the duct had to be placed into the structure, the collar and end caps had to be installed and the duct had to be positioned at the correct height to give room for expansion. After duct placement sixteen linear potentiometers and tape measures were installed to measure deflection. Data was then collected during the deflection test and afterwords the duct was removed from the structure to prepare for the next specimen.

7. EXPERIMENTAL TEST RESULTS

The experimental test results can be split into the following three categories.

1. Raw experimental data, which is the pressure verses deflection results for each individual specimen (gauge and flat span comparison).
2. The effect of reinforcements.
3. The plastic deformations observed during testing.

7.1 Experimental Data

7.1.1 General Observations

Figure 7.1 is an example of a graph of raw deflection data and pressure data for a specific duct. This plot is discussed, because it demonstrates general behavior and characteristics that can be found in all graphs. In addition, it provides an opportunity to show data for multiple sensors at a variety of locations. Afterwords, the plot will show data from only the two most applicable sensors. The blue in the plot dots are the normalized data, which is derived from an average of the two main top sensors. The top sensors were chosen for normalization, because their change was the most prominent. The black triangles in the plot are the two bottom sensors, whose behavior is almost horizontal after 2 inWG. The reason for this behavior is as the bottom pushes out from the pressure then the duct rises on its supports and the second moment of inertia increases causing the duct to stiffen. The raw data graphs can be seen on pages 33 through 46. These graphs also show that for a constant gauge and shorter flat spans result in smaller deflections. Similarly, the thicker the gauges show that these ducts for a constant flat spans result in smaller deflections. Reinforced ducts as compared to unreinforced ducts, when all other duct properties are similar, have substantially smaller deflections.

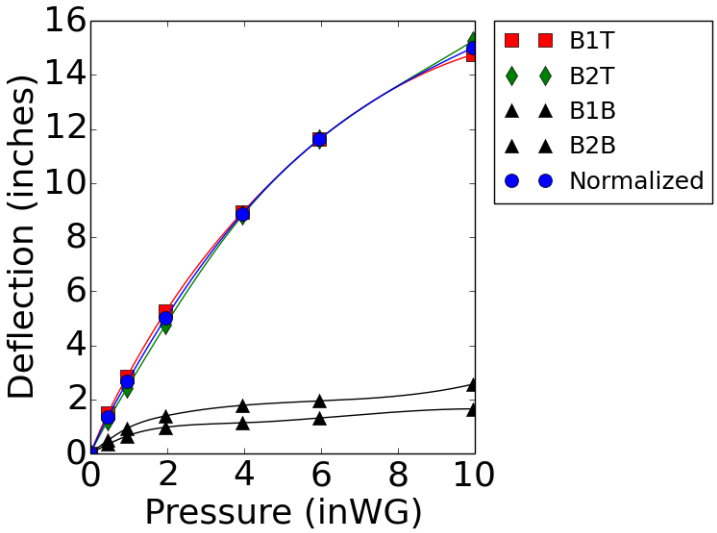
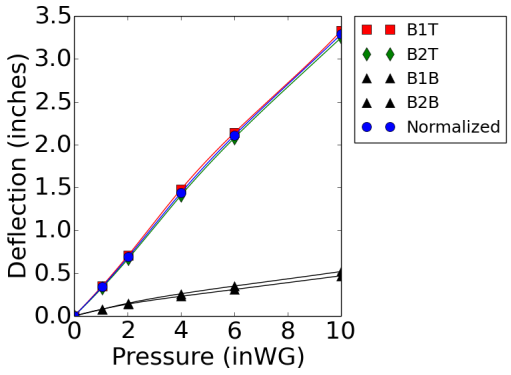


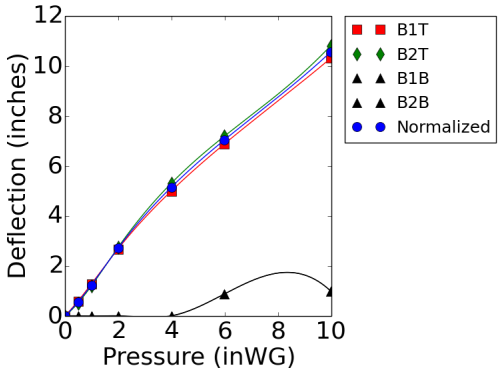
Figure 7.1.: 22 Gauge and 38 inch Flat Span

7.1.2 Unreinforced Test Results

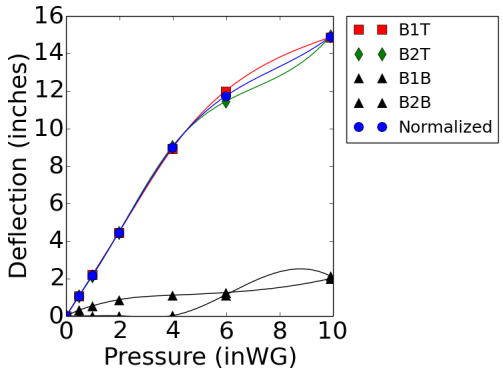
In Figures 7.2 and 7.2 data curves for the top sensors and normalized data show as the duct becomes more round (smaller flat span) then the curve becomes more concave. Also, in Figures 7.3 and 7.3 as the duct reaches its deflection limit then the curve becomes horizontal due to the duct deforming until the top and bottom touch. Figures 7.2 through 7.3 show as gauge thins deflection increases and as flat span increases deflection also increases. Figure 7.4 shows ducts with a 4 and 30 inch height, and features the other graphs can be observed.



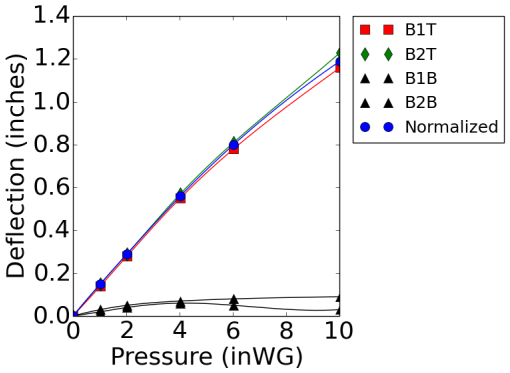
(a) 18 Gauge and 25 inch Flat Span



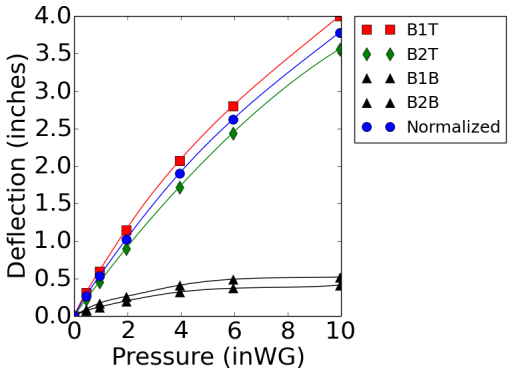
(b) 18 Gauge and 48 inch Flat Span



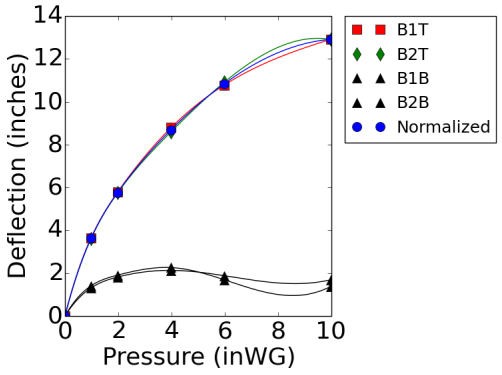
(c) 20 Gauge and 48 inch Flat Span



(d) 24 Gauge and 6 inch Flat Span

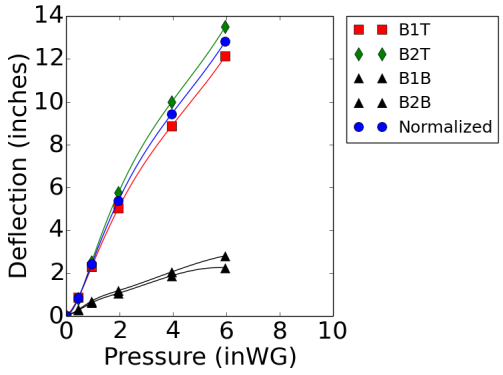


(e) 24 Gauge and 14 inch Flat Span

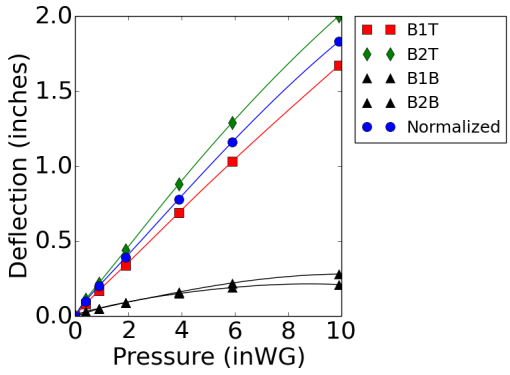


(f) 26 Gauge and 25 inch Flat Span

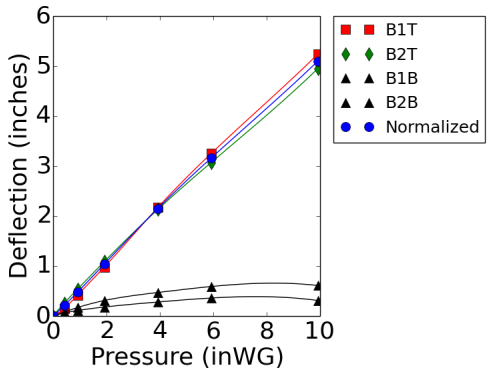
Figure 7.2.: Unreinforced Positive



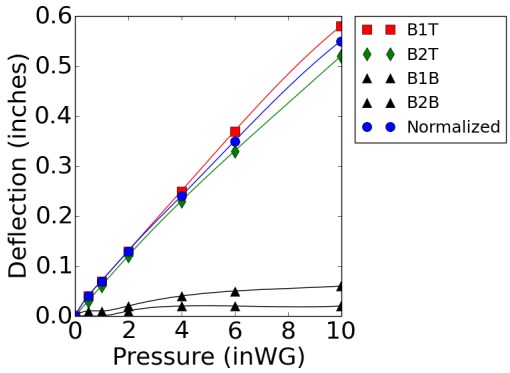
(g) (cont.) 18 Gauge and 63 inch Flat Span



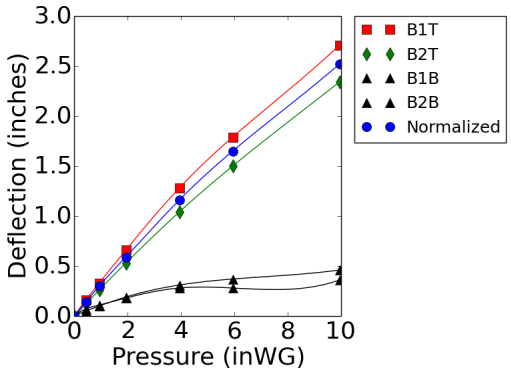
(h) (cont.) 20 Gauge and 14 inch Flat Span



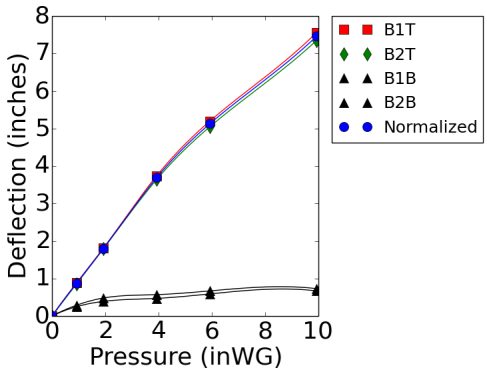
(i) (cont.) 20 Gauge and 25 inch Flat Span



(j) (cont.) 22 Gauge and 6 inch Flat Span

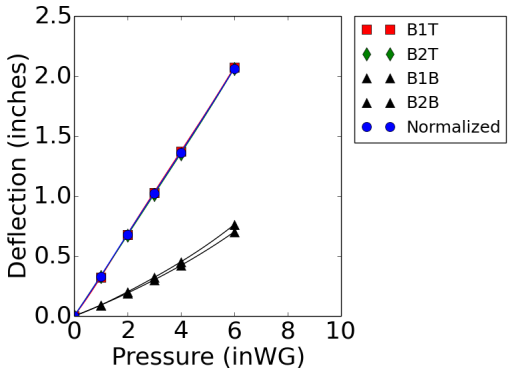


(k) (cont.) 22 Gauge and 14 inch Flat Span

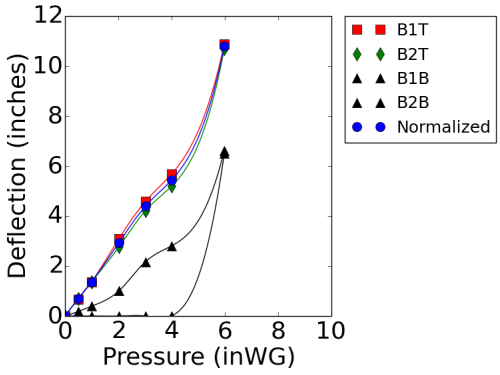


(l) (cont.) 22 Gauge and 25 inch Flat Span

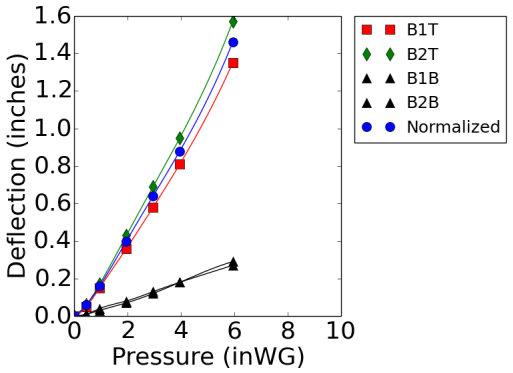
Figure 7.2. (cont.)



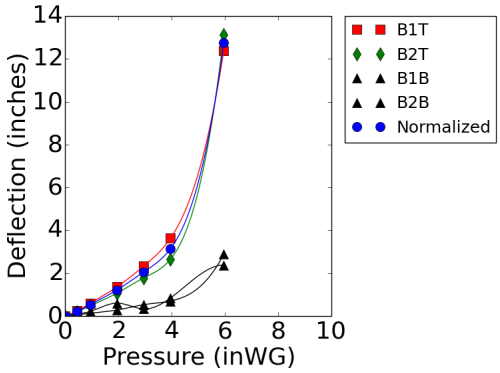
(a) 18 Gauge and 25 inch Flat Span



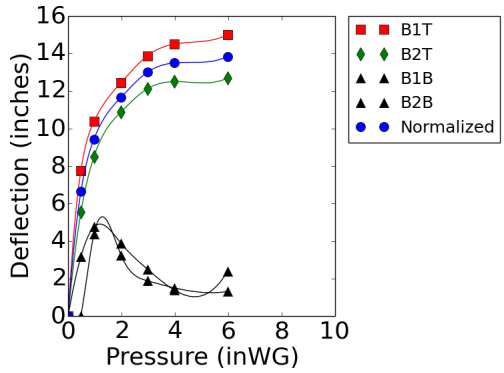
(b) 18 Gauge and 48 inch Flat Span



(c) 20 Gauge and 14 inch Flat Span

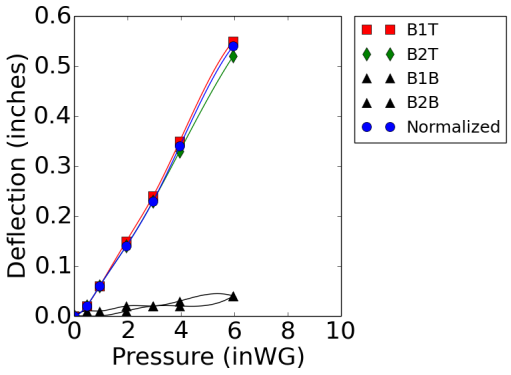


(d) 24 Gauge and 14 inch Flat Span

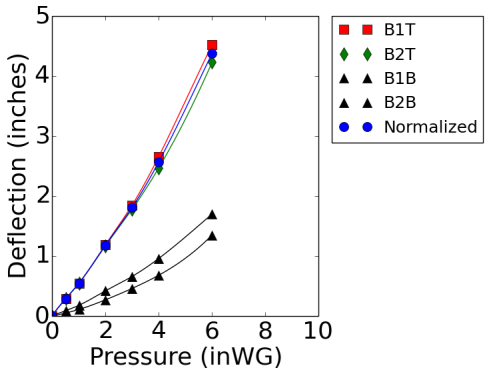


(e) 26 Gauge and 48 inch Flat Span

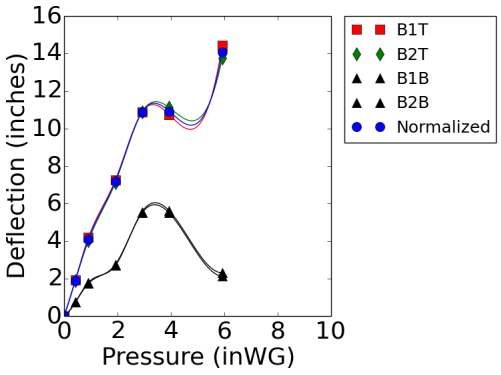
Figure 7.3.: Unreinforced Negative



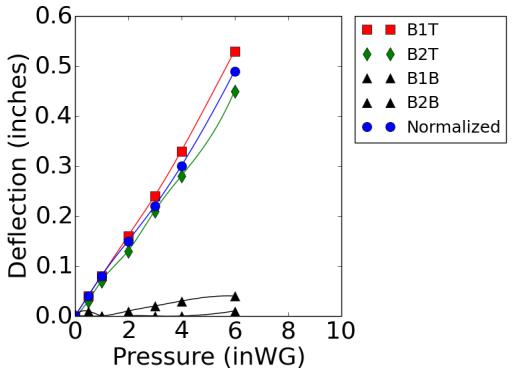
(f) (cont.) 18 Gauge and 14 inch Flat Span



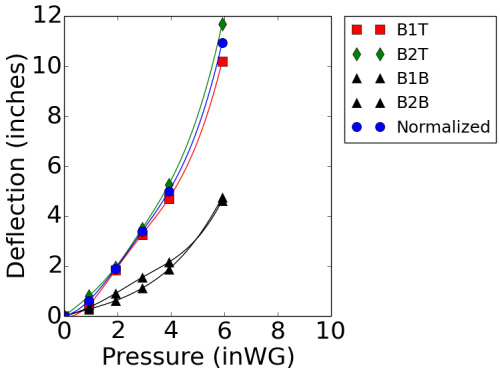
(g) (cont.) 20 Gauge and 25 inch Flat Span



(h) (cont.) 20 Gauge and 63 inch Flat Span

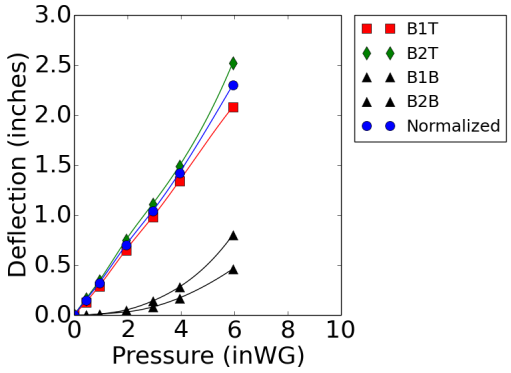


(i) (cont.) 22 Gauge and 6 inch Flat Span

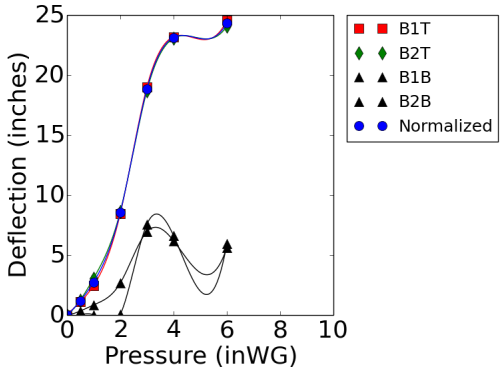


(j) (cont.) 22 Gauge and 25 inch Flat Span

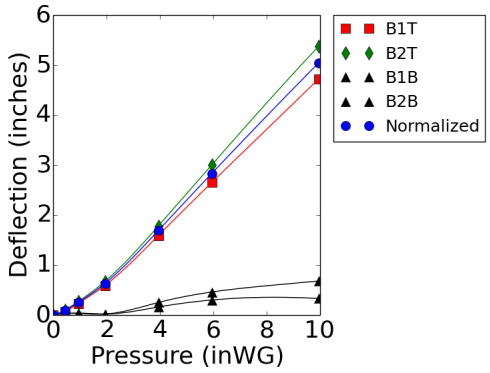
Figure 7.3. (cont.)



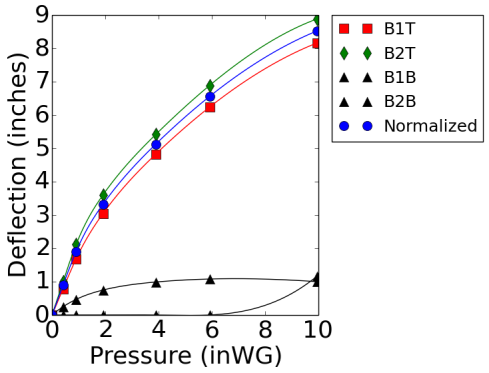
(a) 22 Gauge, 25 inch Flat Span, and 4 inch Height Negative Pressure



(b) 22 Gauge, 25 inch Flat Span, and 30 inch Height Negative Pressure



(c) 22 Gauge, 25 inch Flat Span, and 4 inch Height Positive Pressure



(d) 22 Gauge, 25 inch Flat Span, and 30 inch Height Positive Pressure

Figure 7.4.: Unreinforced 4 inch and 30 inch height

7.1.3 Transverse Reinforcing Test Results

The transverse reinforced duct in Figure 7.5 had four sections, which increased leakage and caused the center transverse connector to buckle before reaching 10 inWG. Even with the problem, the deflection was still less than Figure 7.6c, which shows closer reinforcements reduce deflection.

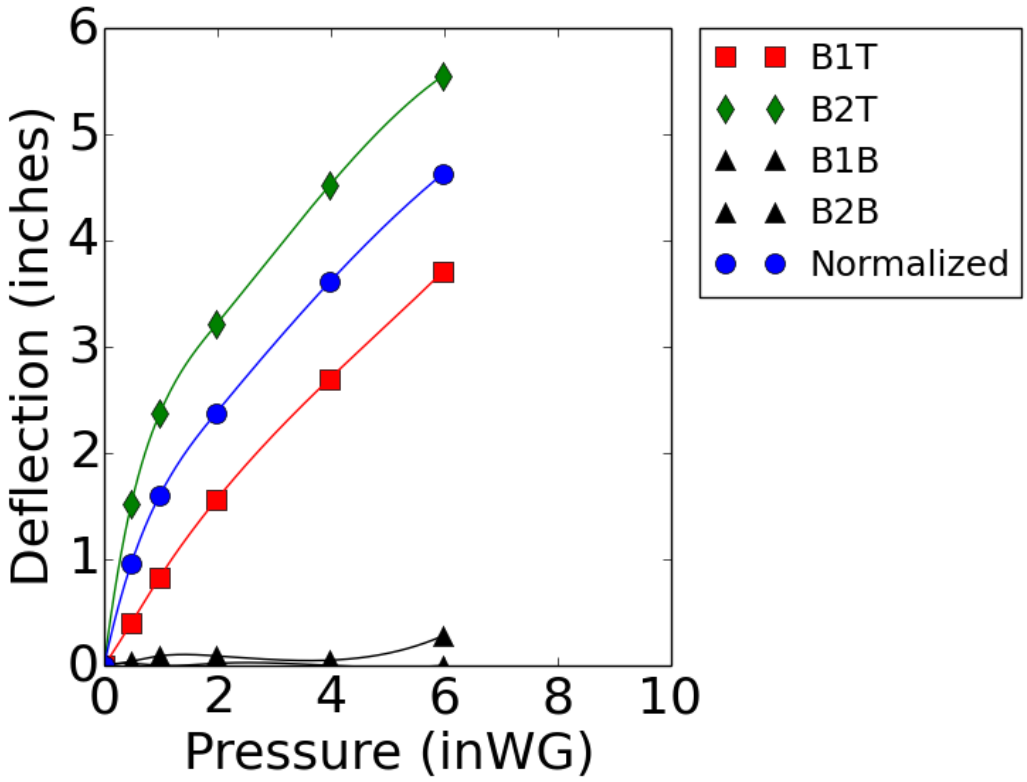
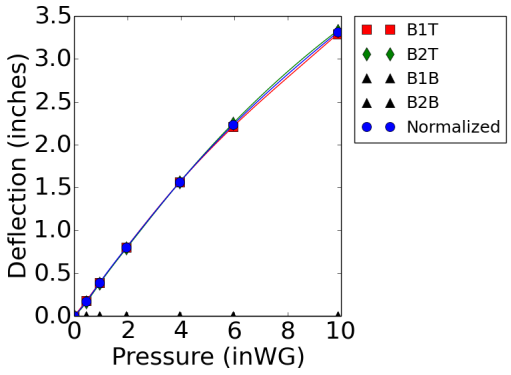
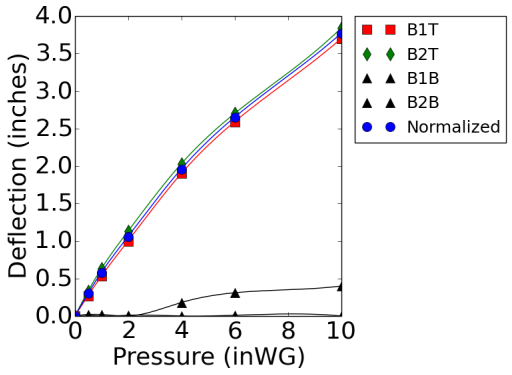


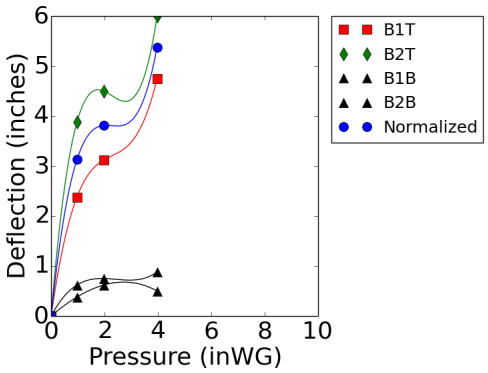
Figure 7.5.: 22 Gauge and 63 inch Flat Span



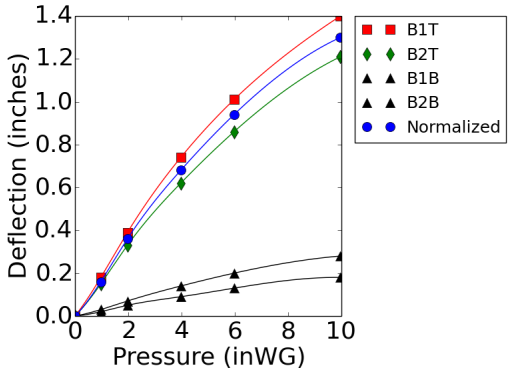
(a) 20 Gauge and 25 inch Flat Span



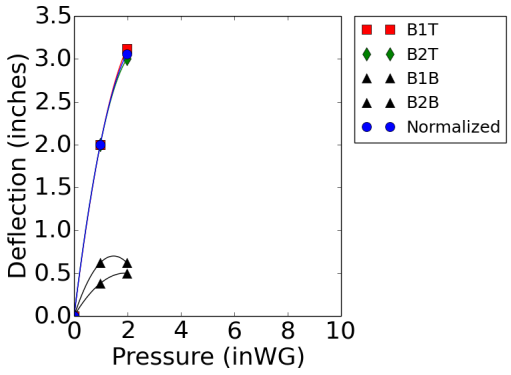
(b) 22 Gauge and 25 inch Flat Span



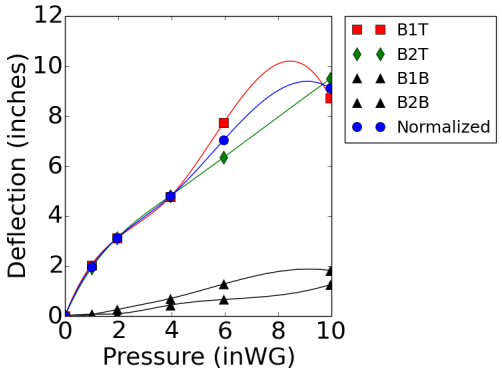
(c) 22 Gauge and 63 inch Flat Span



(d) 24 Gauge and 6 inch Flat Span

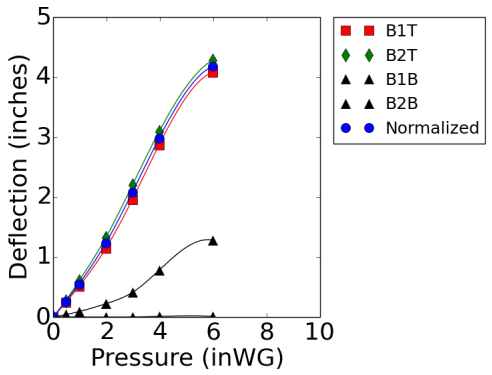


(e) 24 Gauge and 48 inch Flat Span

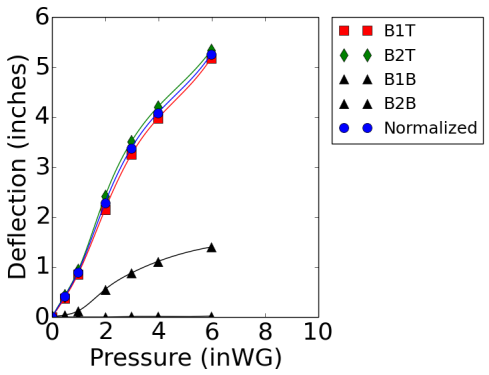


(f) 26 Gauge and 25 inch Flat Span

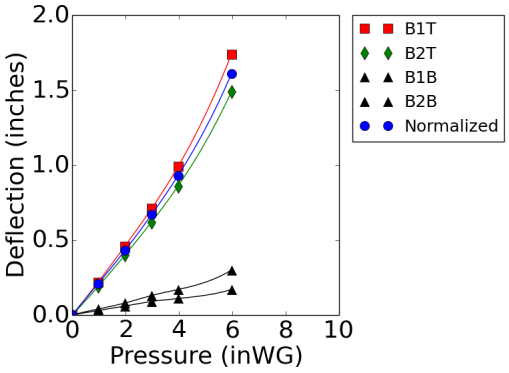
Figure 7.6.: T-25 Positive



(a) 20 Gauge and 25 inch Flat Span



(b) 22 Gauge and 25 inch Flat Span

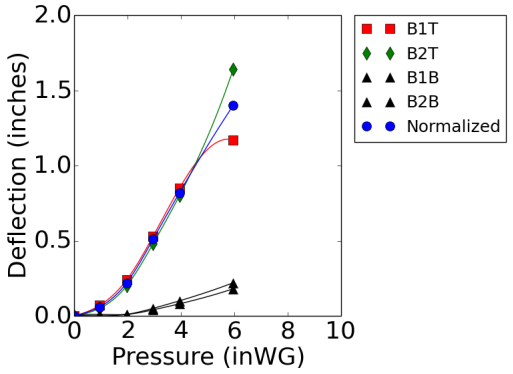


(c) 24 Gauge and 6 inch Flat Span

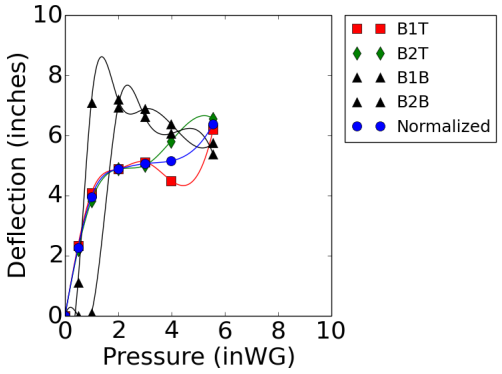
Figure 7.7.: T-25 Negative

7.1.4 Trapezed Reinforcing Test Results

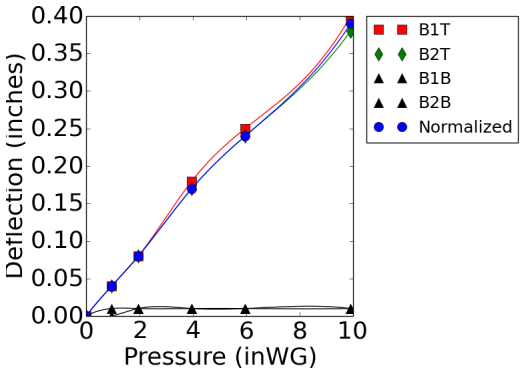
The trapezed reinforced graphs are shown in Figure 7.8 and 7.9. Figure 7.8d and 7.9d shows closer spaced reinforcement reduces deflection.



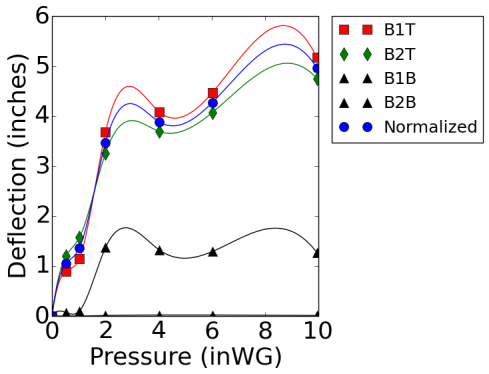
(a) 24 Gauge and 6 inch Flat Span Negative Pressure



(b) 24 Gauge and 48 inch Flat Span Negative Pressure

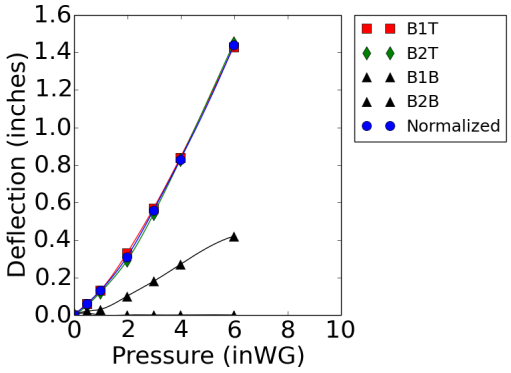


(c) 24 Gauge and 6 inch Flat Span Positive Pressure

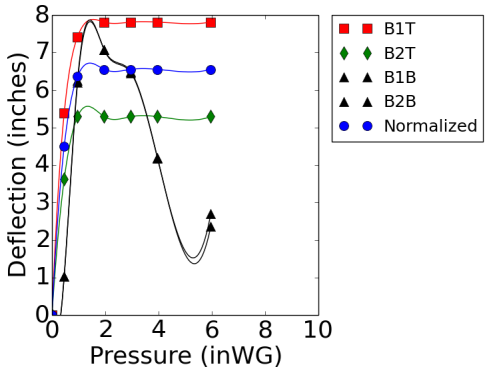


(d) 24 Gauge and 48 inch Flat Span Positive Pressure

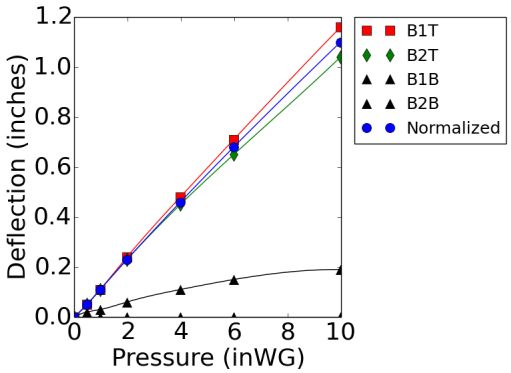
Figure 7.8.: Trapeze 3ft



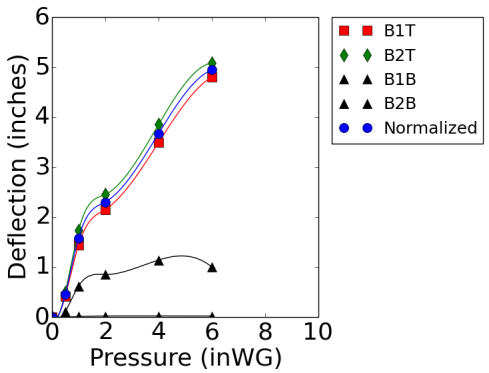
(a) 18 Gauge and 25 inch Flat Span Negative Pressure



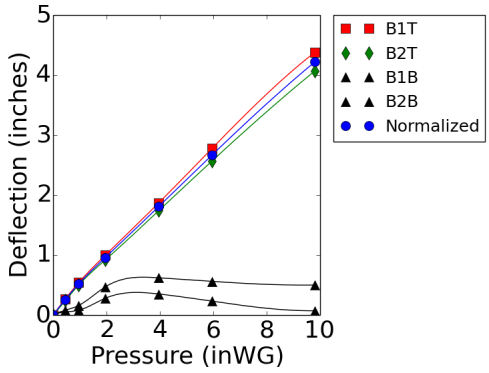
(b) 24 Gauge and 48 inch Flat Span Negative Pressure



(c) 18 Gauge and 25 inch Flat Span Positive Pressure



(d) 24 Gauge and 48 inch Flat Span Positive Pressure

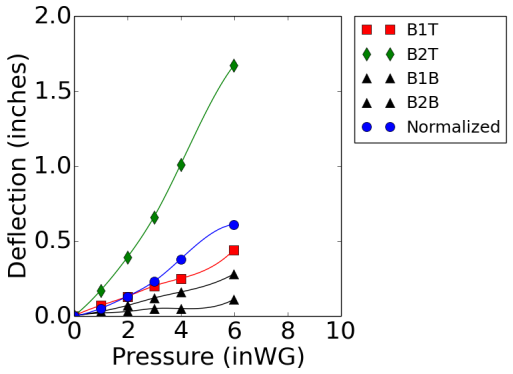


(e) 26 Gauge and 25 inch Flat Span Positive Pressure

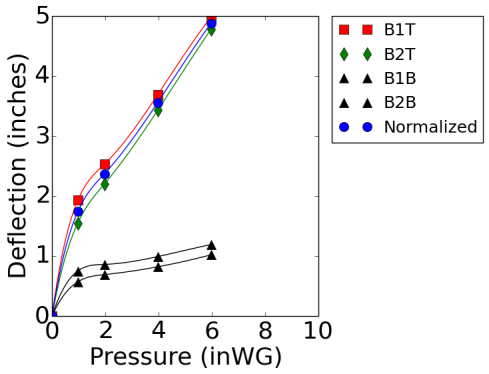
Figure 7.9.: Trapeze 6ft

7.1.5 Attached Reinforcing Test Results

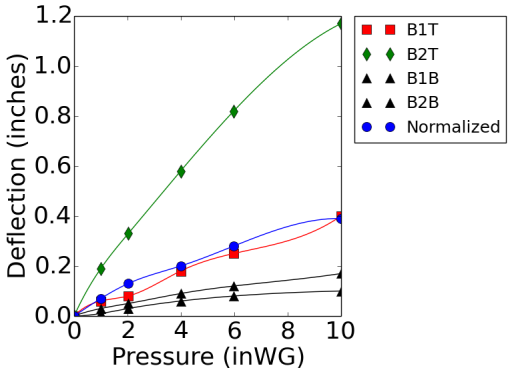
The attached reinforcing raw data graphs are shown in Figure 7.10 and 7.11. Figure 7.10b and 7.11b show that closer reinforcement spacings result in lower deflections.



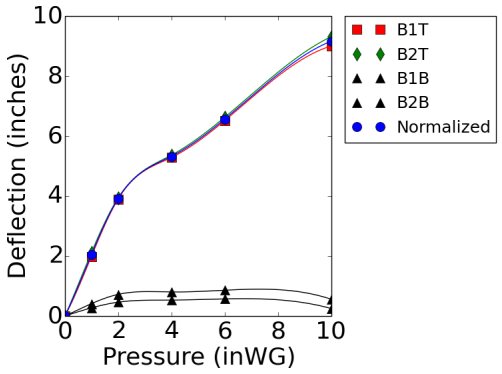
(a) 24 Gauge and 6 inch Flat Span Negative Pressure



(b) 22 Gauge and 63 inch Flat Span Positive Pressure

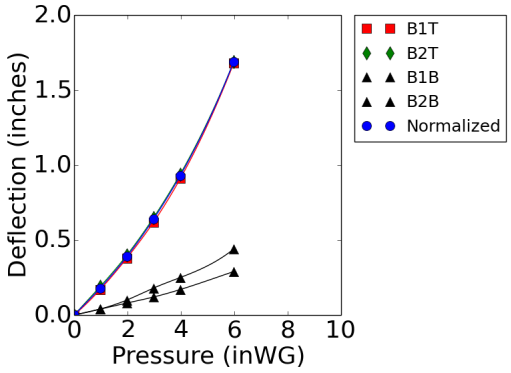


(c) 24 Gauge and 6 inch Flat Span Positive Pressure

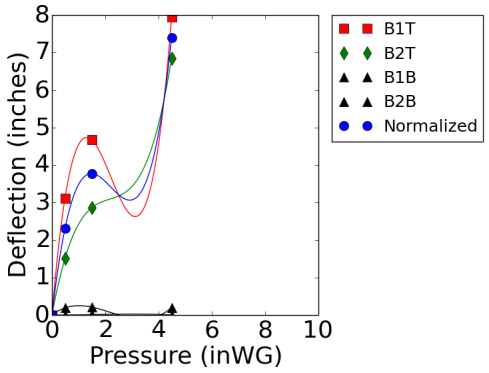


(d) 24 Gauge and 48 inch Flat Span Positive Pressure

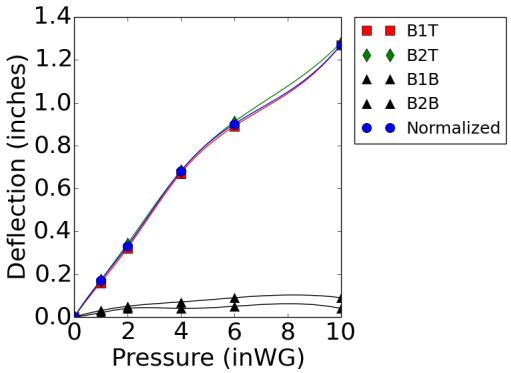
Figure 7.10.: Attached 3ft



(a) 24 Gauge and 6 inch Flat Span Negative Pressure



(b) 22 Gauge and 63 inch Flat Span Positive Pressure

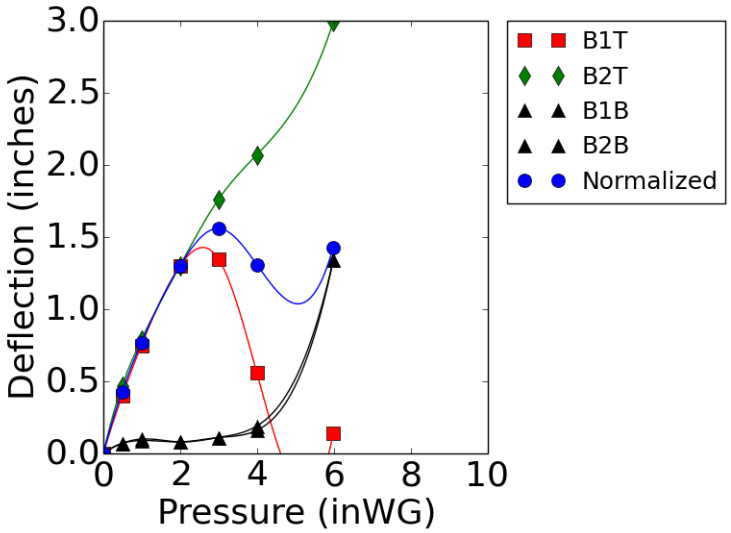


(c) 24 Gauge and 6 inch Flat Span Positive Pressure

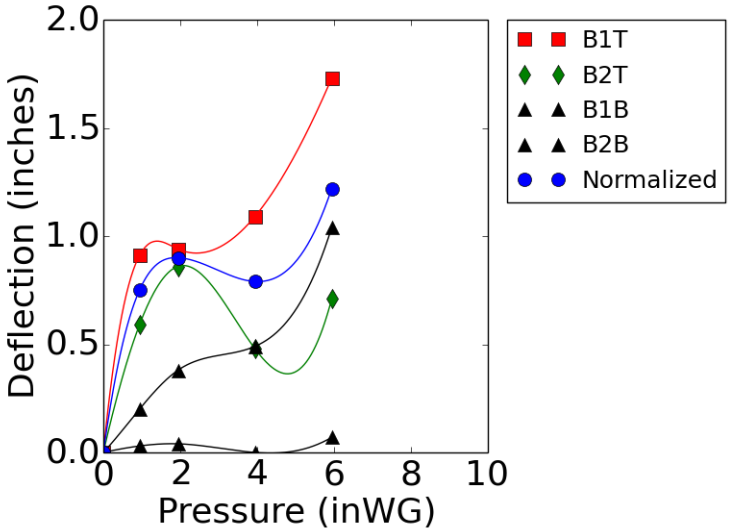
Figure 7.11.: Attached 6ft

7.1.6 Tie Rod Reinforcing Test Results

The tie rod reinforcing raw data graphs are shown in Figure 7.12 and 7.13. These figures also show that the closer the reinforcements the less the deflection.

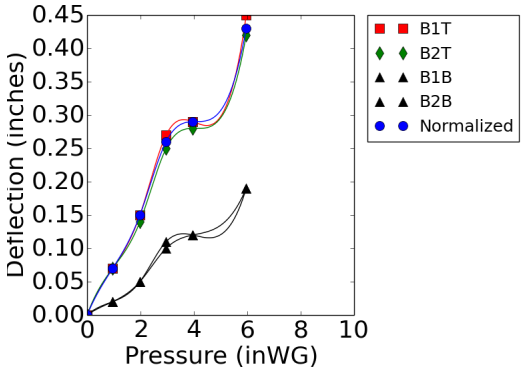


(a) 22 Gauge and 63 inch Flat Span Negative Pressure

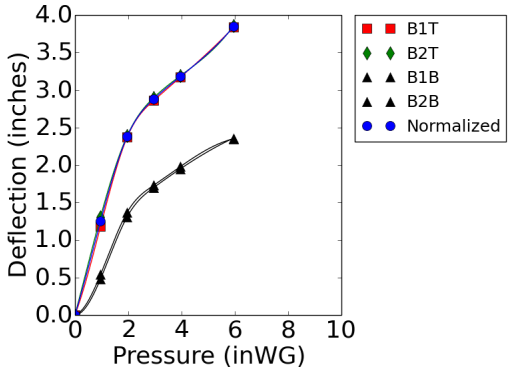


(b) 24 Gauge and 48 inch Flat Span Positive Pressure

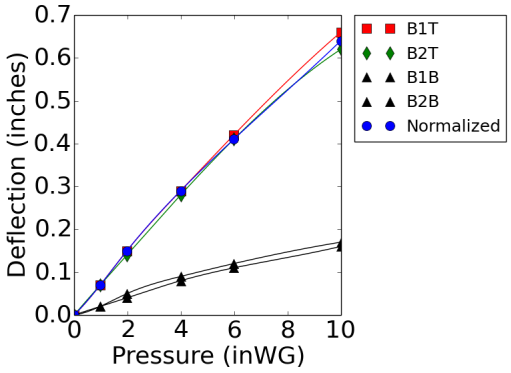
Figure 7.12.: Tie Rod 3ft



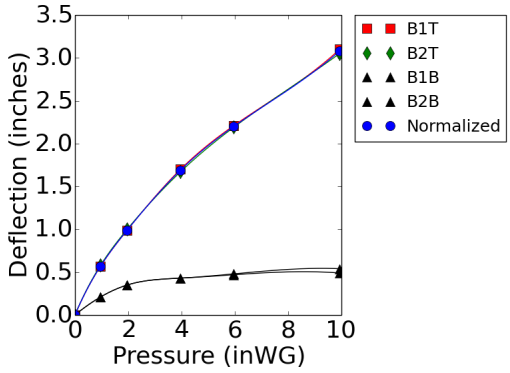
(a) 24 Gauge and 6 inch Flat Span Negative Pressure



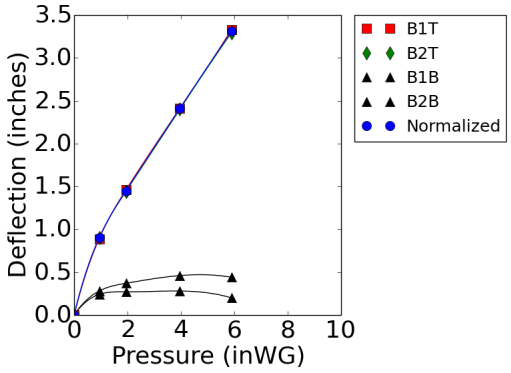
(b) 24 Gauge and 25 inch Flat Span Negative Pressure



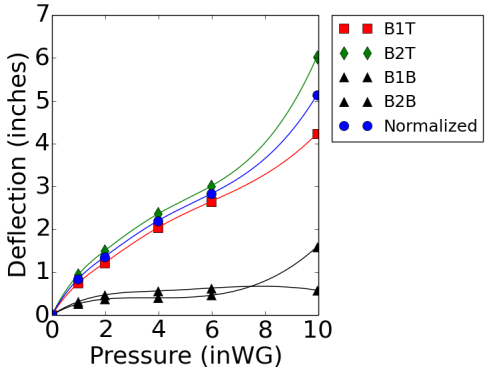
(c) 24 Gauge and 6 inch Flat Span Positive Pressure



(d) 24 Gauge and 25 inch Flat Span Positive Pressure



(e) 24 Gauge and 48 inch Flat Span Positive Pressure



(f) 26 Gauge and 25 inch Flat Span Positive Pressure

Figure 7.13.: Tie Rod 6ft

7.2 Reinforcements

Tables 7.1 and 7.2 present data at pressures of 6 inWG and -6 inWG. These pressure points have a large enough deflection to discern between ducts, without causing failure in the ducts. The results shown in Tables 7.1 and 7.2 the unreinforced positive and unreinforced negative along with the T-25 positive are taken from regression models, while the rest of the results shown are actual experimental data.

Tables 7.1 and 7.2 show that the smaller the reinforcing spacing result in smaller deflections. The unreinforced samples are always going to have the most deflection for a given gauge, flat span, and pressure, because a unreinforced sample could represent any reinforcement type when the reinforcement spacing is greater than the length of the duct.

When comparing attached, trapeze, and tie rod reinforcements, an equal reinforcement spacing is needed. The 24 gauge and 48 inch flat span duct concludes the order from least deflection to most deflection: Tie Rod, Trapeze, and Attached Reinforcing, also shown in Table 7.3.

The T-25 reinforcement was better than both the unreinforced and attached reinforcement as seen in Table 7.1. It is hard to draw conclusions between the T-25 and the other reinforcements, due to the reinforcement spacing being different and the attached reinforcement not bounding the T-25 reinforcement.

7.3 Deformation

Physical deformations were observed during testing. A seam would unravel on the thin gauge ducts as seen in Figure 7.14a. A T-25 flange failure causes the center of the flat span and the center of the radius to bend as shown in Figure 7.14c and Figure 7.14d. Figure 7.14b shows the angle iron tearing small holes at the weld on the end of the angle iron for the attached reinforcing. The tie rod reinforcing would

pull the nut out of the rod or tear through the sheet metal around the washer. High pressure caused many deformations. This was rare; therefore, data was collected up to the deformation pressure.

7.4 Summary

The preliminary interpretation of the raw data, concluded the order from least deflection to most deflection: tie rod, trapeze, and attached reinforcing. The unreinforced always deflects more than any of the reinforcing. The closer the reinforcement is spaced the less the deflection. Physical deformations were also observed during testing. Data was collected up until the physical deformation or 10 inWG.

Table 7.1.: Matrix Positive 6 inWG

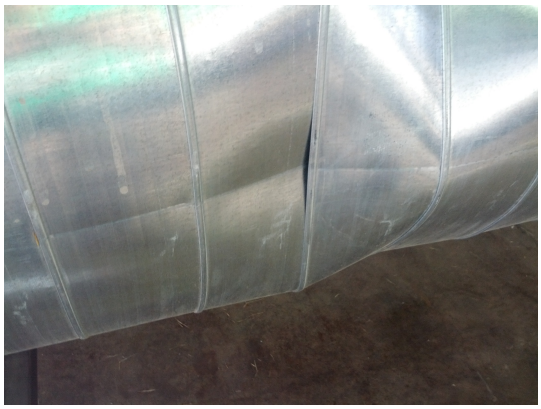
Gauge	18						20						22						24						26					
Flatspan	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6
Unreinforced	P:6.0 D:12.9	P:6.0 D:7.4	P:6.0 D:4.5	P:6.0 D:2.1			P:6.0 D:19.1	P:6.0 D:11.9	P:6.0 D:7.8	P:6.0 D:3.4	P:5.9 D:1.2			P:6.0 D:16.7	P:6.0 D:11.4	P:6.0 D:5.6	P:5.9 D:1.6	P:6.0 D:0.3		P:6.0 D:21.9	P:6.0 D:15.5	P:6.0 D:8.1	P:5.9 D:2.6	P:6.0 D:0.8			P:6.0 D:19.9	P:6.0 D:11.0	P:6.0 D:4.3	
T-25 6ft lengths													P:6.0 D:4.6																	
T-25 12ft lengths	P:6.0 D:11.8	P:6.0 D:9.6	P:6.0 D:7.3	P:6.0 D:3.2			P:6.0 D:8.7	P:6.0 D:7.4	P:6.0 D:5.7	P:6.0 D:2.4			P:6.0 D:6.6	P:6.0 D:6.3	P:6.0 D:5.2	P:6.0 D:2.8			P:6.0 D:5.8	P:6.0 D:6.3	P:6.0 D:5.9	P:6.0 D:4.2	P:6.0 D:1.9	P:6.0 D:0.9	P:6.0 D:6.0	P:6.0 D:7.5	P:6.0 D:7.6	P:6.0 D:6.8	P:6.0 D:5.1	P:6.0 D:3.4
Attached reinforcing 3ft centers													P:6.0 D:4.9							P:6.0 D:6.6				P:6.0 D:0.3						
Attached reinforcing 6ft centers													P:4.5 D:7.4											P:6.0 D:0.9						
Trapeze 3ft centers																				P:6.0 D:4.3				P:6.0 D:0.2						
Trapeze 6ft centers				P:6.0 D:0.7																P:6.0 D:4.9								P:6.0 D:2.7		
Tie Rod 3ft centers																				P:5.9 D:1.2										
Tie Rod 6ft centers																				P:5.9 D:3.3		P:6.0 D:2.2		P:6.0 D:0.4				P:6.0 D:2.8		

Table 7.2.: Matrix Negative 6 inWG

Gauge	18						20						22						24						26					
Flatspan	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6	63	48	38	25	14	6
Unreinforced		P:-6.0 D:-10.8	P:-6.0 D:-7.2	P:-6.0 D:-4.6	P:-6.0 D:-2.1		P:-5.9 D:-14.1		P:-6.0 D:-8.8	P:-6.0 D:-6.8	P:-6.0 D:-4.8	P:-6.0 D:-3.2				P:-6.0 D:-8.7	P:-6.0 D:-7.3	P:-6.0 D:-6.1					P:-6.0 D:-12.8	P:-6.0 D:-8.7		P:-6.0 D:-13.8				
T-25 6ft lengths																														
T-25 12ft lengths										P:-6.0 D:-4.2						P:-6.0 D:-5.3								P:-6.0 D:-1.6						
Attached reinforcing 3ft centers																								P:-6.0 D:-0.6						
Attached reinforcing 6ft centers																								P:-6.0 D:-1.7						
Trapeze 3ft centers																				P:-5.6 D:-6.4				P:-6.0 D:-1.4						
Trapeze 6ft centers				P:-6.0 D:-1.4																P:-6.0 D:-6.5										
Tie Rod 3ft centers													P:-6.0 D:-1.4																	
Tie Rod 6ft centers																						P:-6.0 D:-3.9		P:-6.0 D:-0.4						

Table 7.3.: Reinforcement Comparison

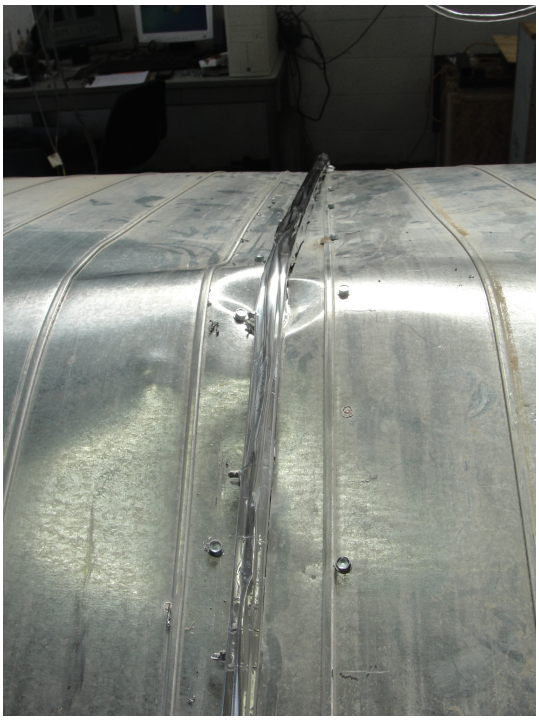
Reinforcement	Deflection
Unreinforced	Worst
Attached Reinforcing	Poor
Trapeze	Better
Tie Rod	Best



(a) Seam Failure

(b) Failed Attached Reinforcing

(c) T-25 Failure



(d) Failed T-25

Figure 7.14.: Deformations

8. EXPERIMENTAL ANALYSIS FOR STANDARD DEVELOPMENT

This experimental analysis provides the low deflection information required for constructing a standard. Deflection under 2 inches is important, because in a ceiling the space is confined, and this means that even small deflections could damage the surroundings.

8.1 Experimental Analysis for Standard Development

Figure 8.1 through 8.14 show flat span verses deflection, over a 0 to 2 inch range, for each gauge at a constant pressure. These figures show as the flat span increases deflection increases, as gauge thins deflection increases, and as pressure increases deflection increases. The graphs show from a 0 to 70 inch flat span range, even though the range tested was from 6 to 63 inch flat spans. The deflection is from 0 to 2 inches, because these graphs can be used to determine the real-world acceptable deflection for a duct; a 2 inch deflection is an unacceptably large deflection, but small enough for the figures to catch all the needed features. The points on these figures are experimental data, whereas the curves are quadratic regressions with the right intercept pinned at the origin. Equation 8.1 shows the form used for the regressions.

$$d(Fs) = |C_1|Fs^2 + |C_2|Fs \tag{8.1}$$

Fs represents the flat span, d represents the deflection, and C represents the constants. The 26 gauge has no data point in the 0 to 2 inch deflection range causing the regression curve to be less reliable, but this curve is left for comparison to the 24 gauge curve.

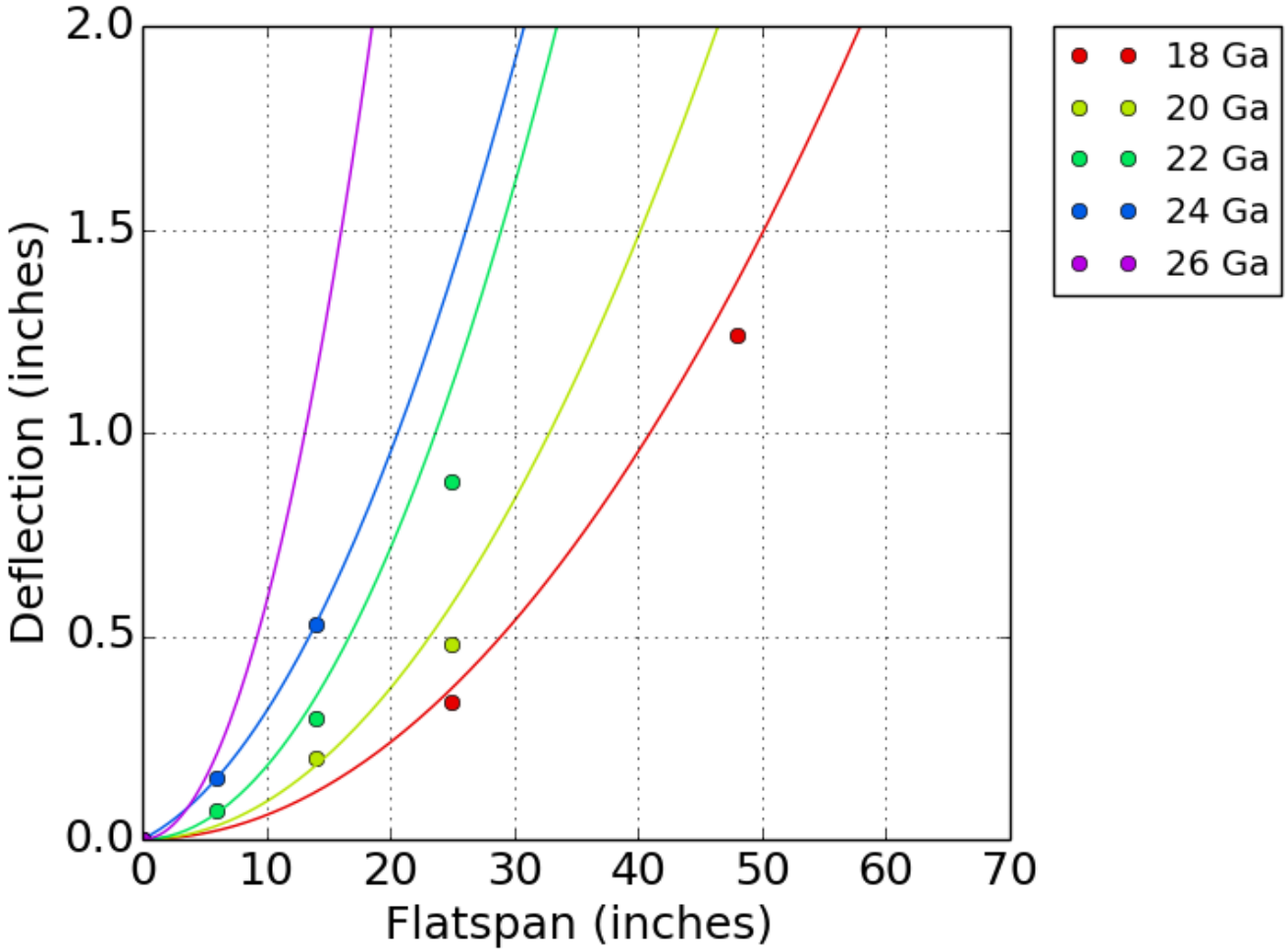


Figure 8.1.: Unreinforced Flat Span vs Deflection for 1 inWG

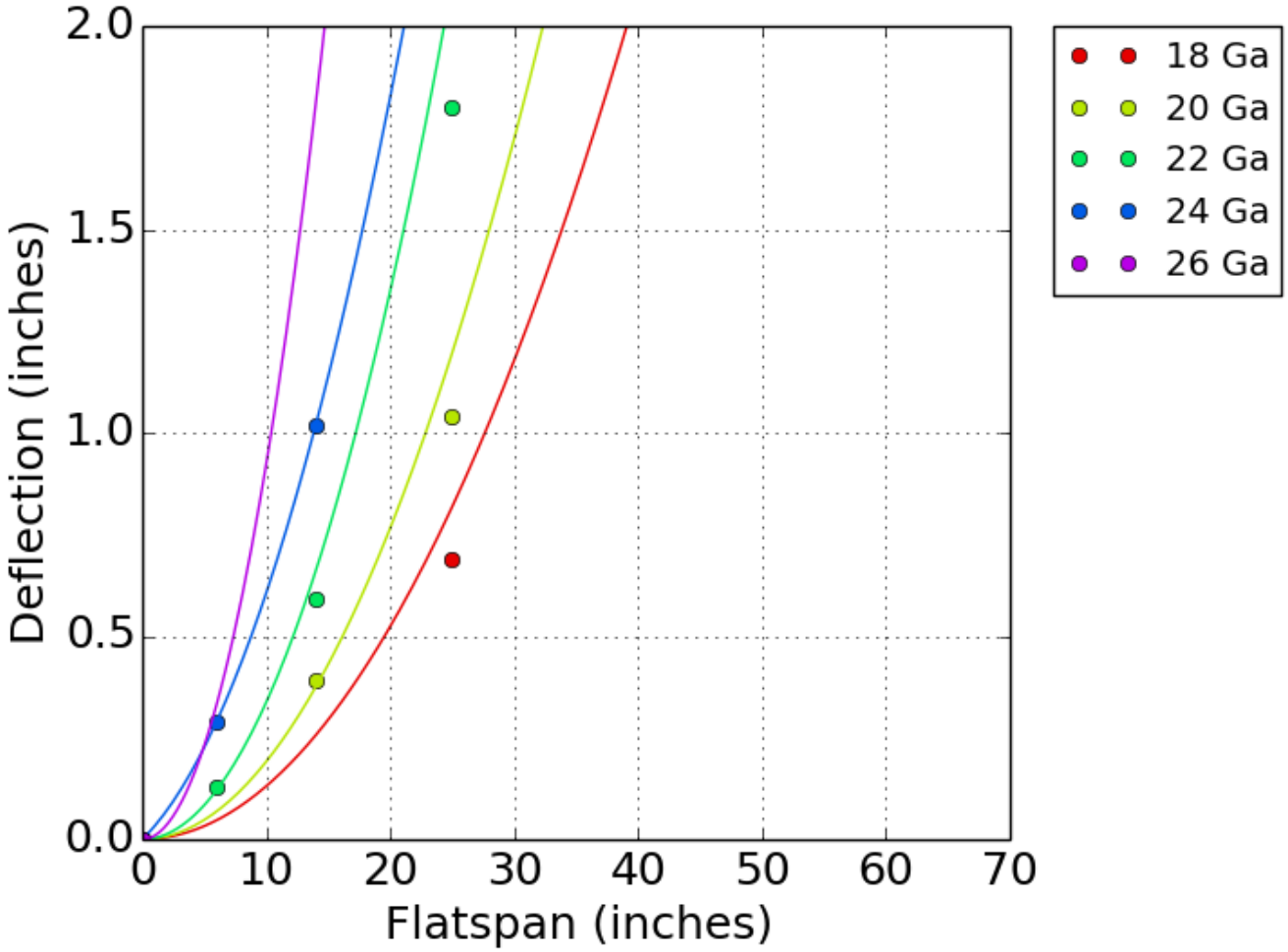


Figure 8.2.: Unreinforced Flat Span vs Deflection for 2 inWG

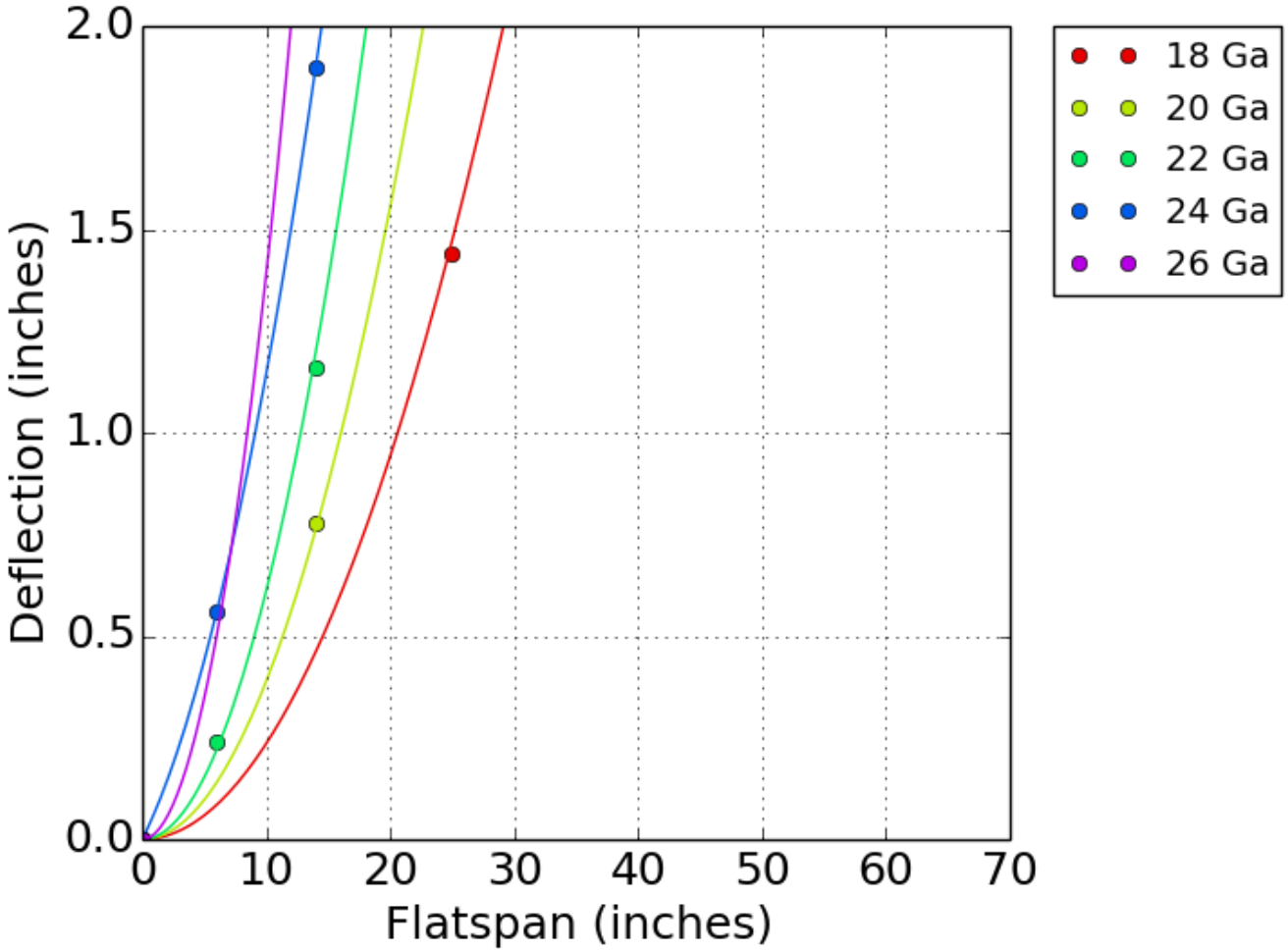


Figure 8.3.: Unreinforced Flat Span vs Deflection for 4 inWG

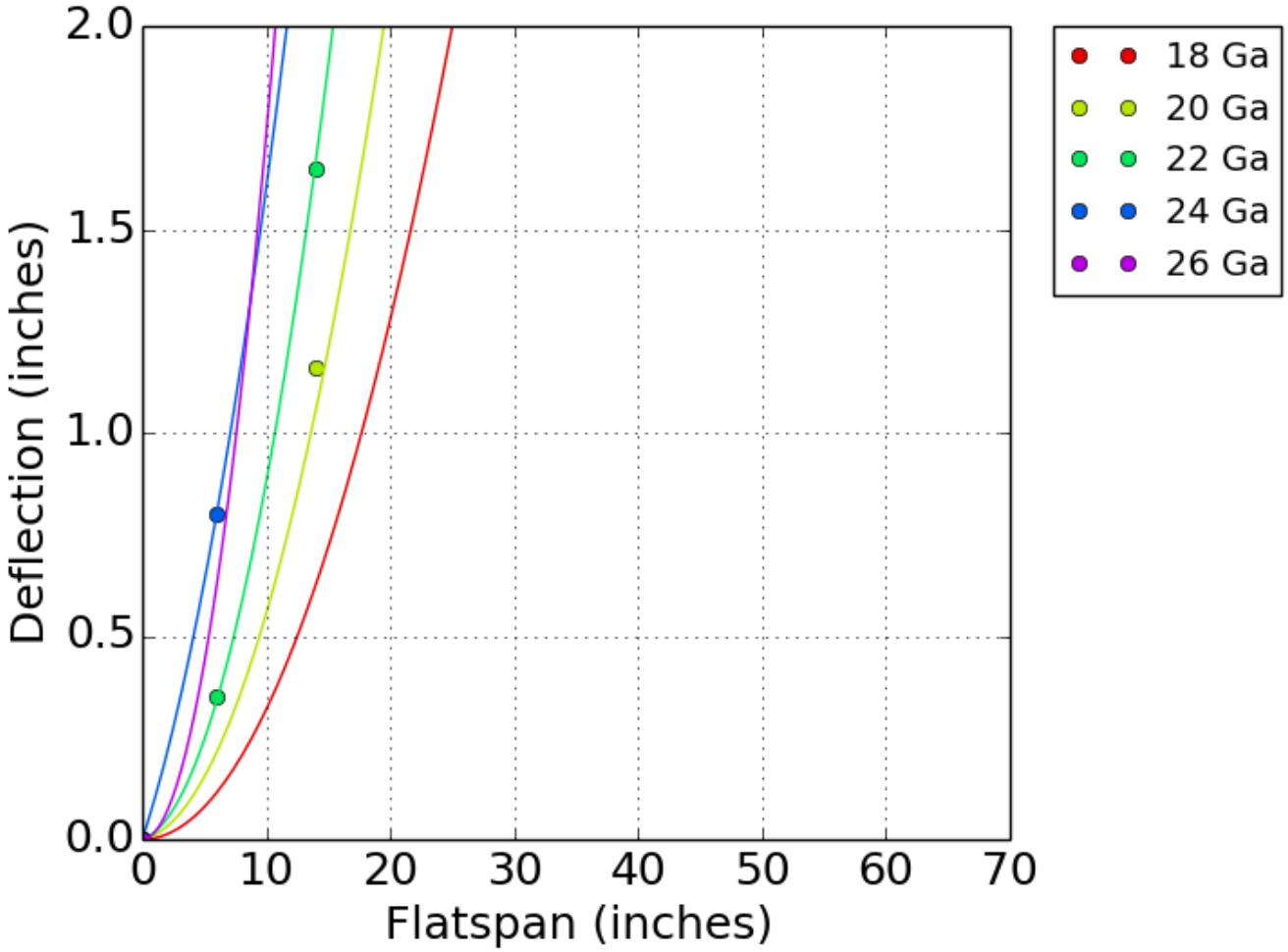


Figure 8.4.: Unreinforced Flat Span vs Deflection for 6 inWG

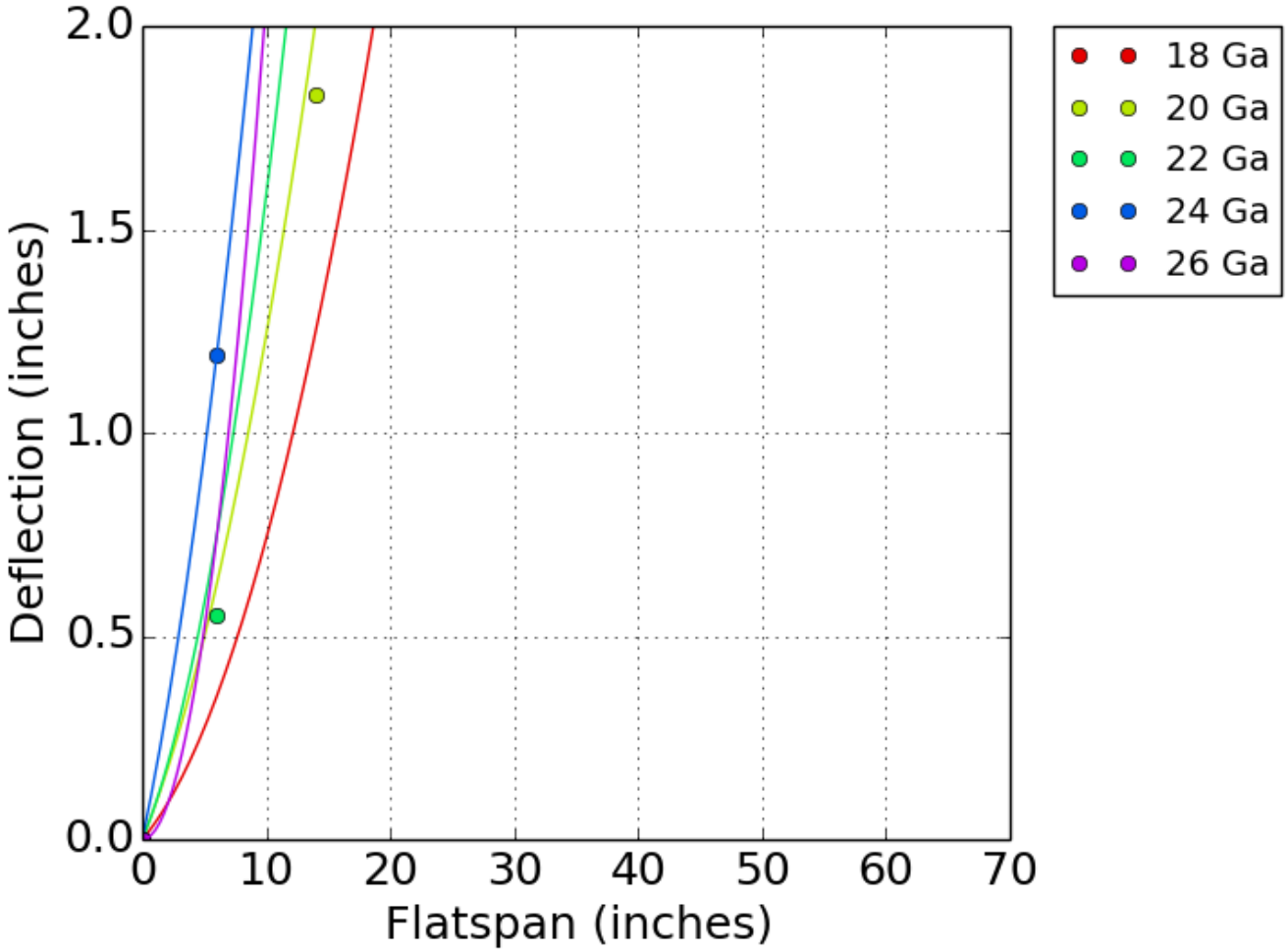


Figure 8.5.: Unreinforced Flat Span vs Deflection for 10 inWG

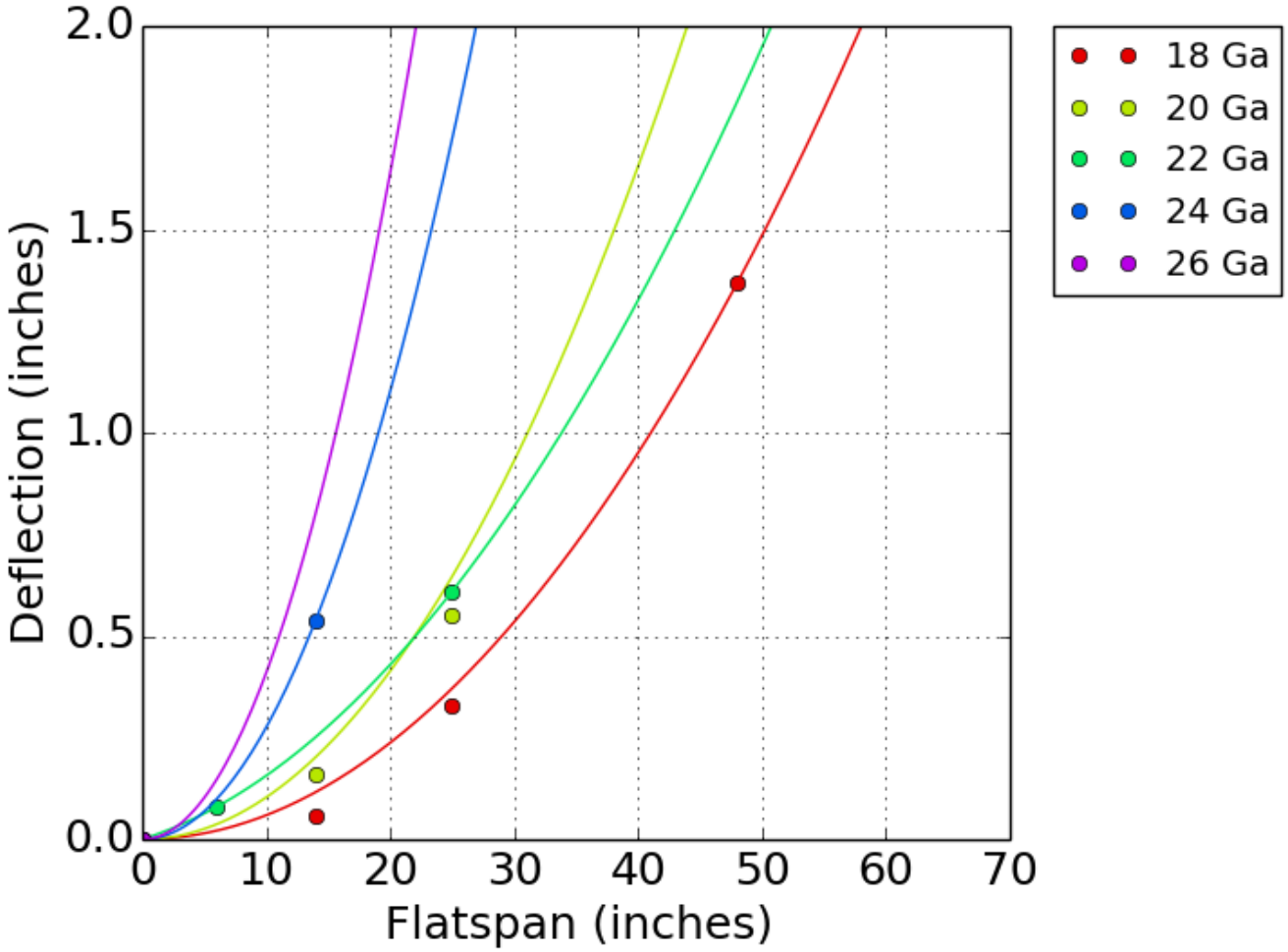


Figure 8.6.: Unreinforced Flat Span vs Deflection for -1 inWG

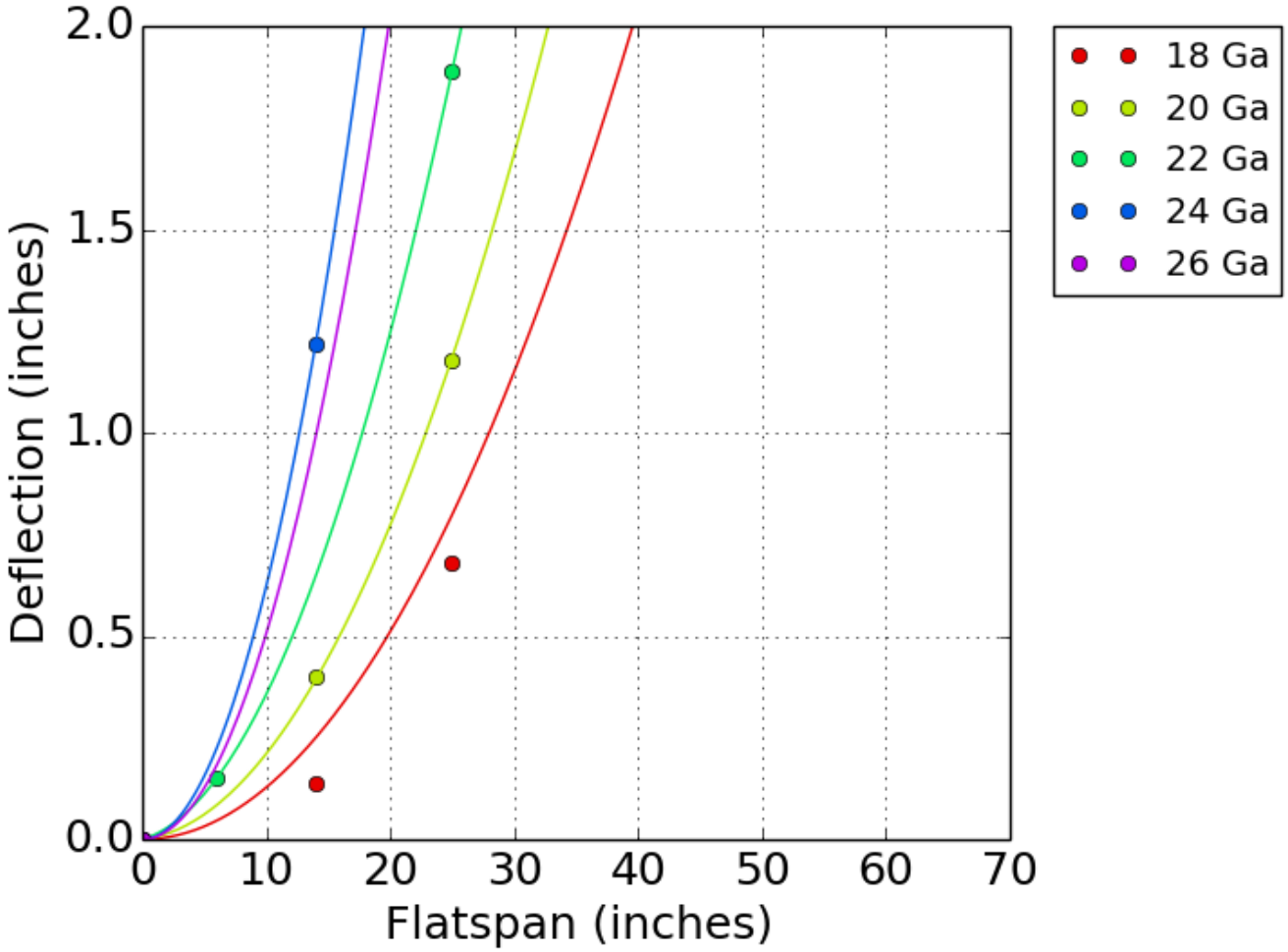


Figure 8.7.: Unreinforced Flat Span vs Deflection for -2 inWG

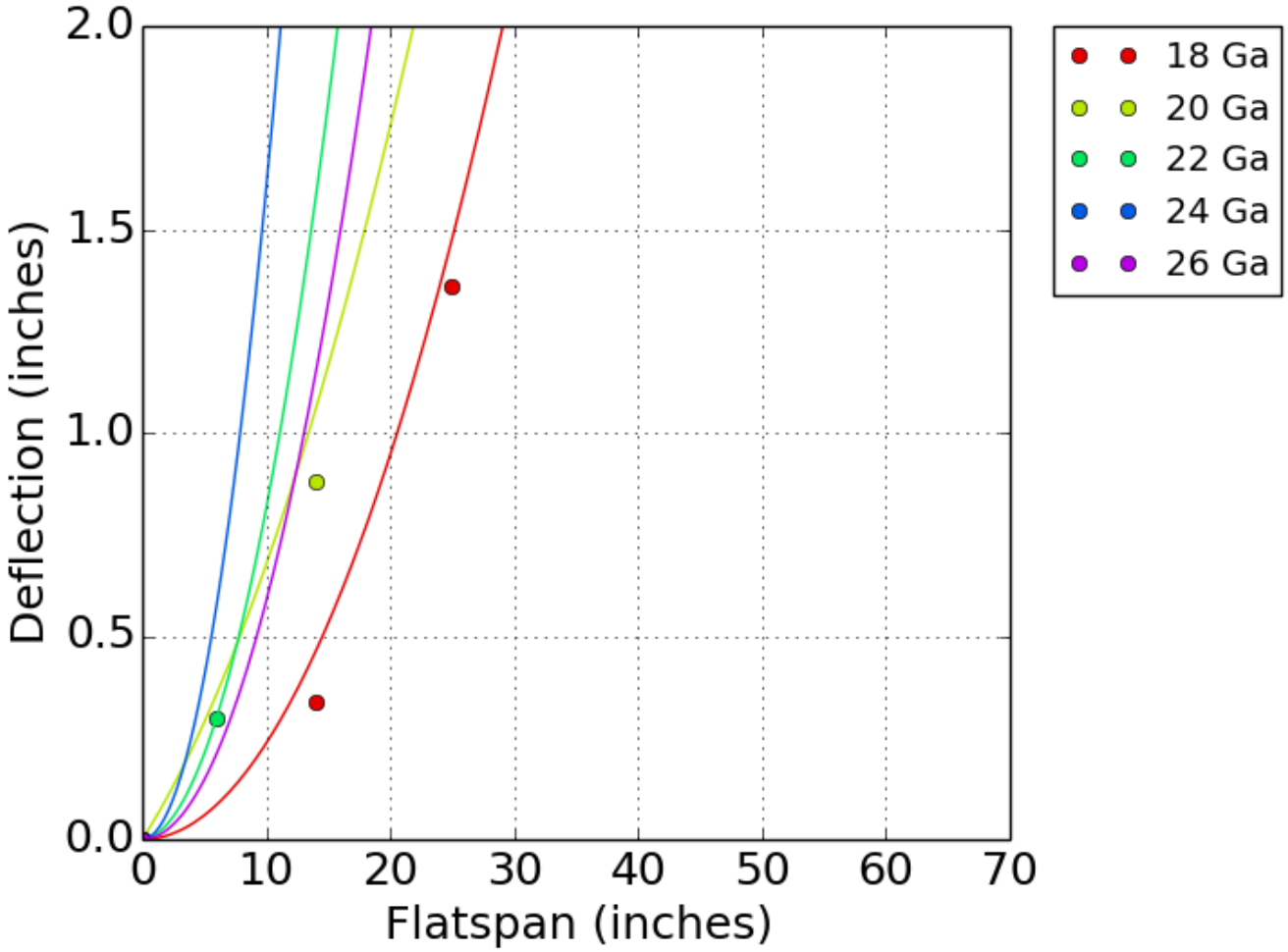


Figure 8.8.: Unreinforced Flat Span vs Deflection for -4 inWG

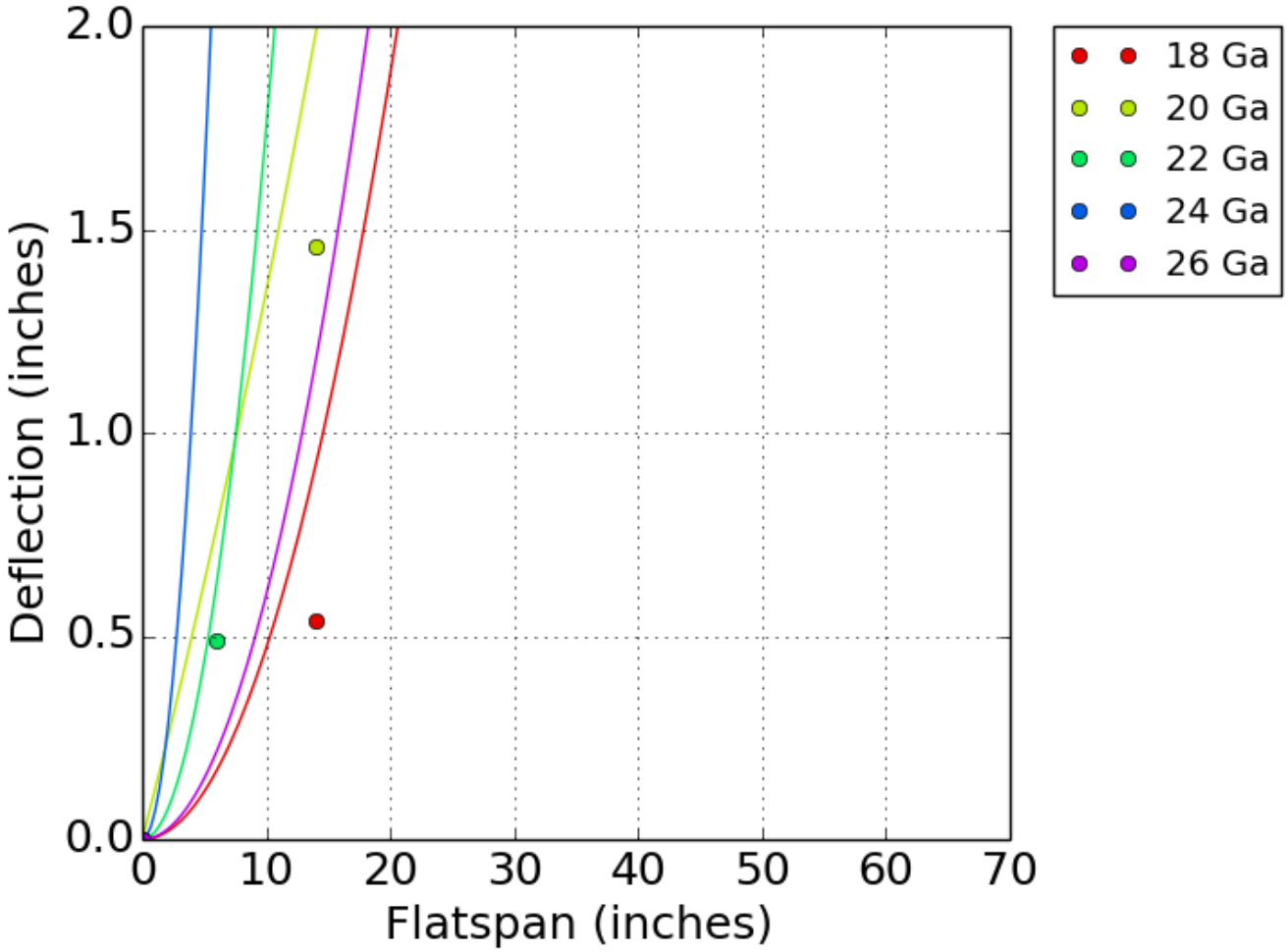


Figure 8.9.: Unreinforced Flat Span vs Deflection for -6 inWG

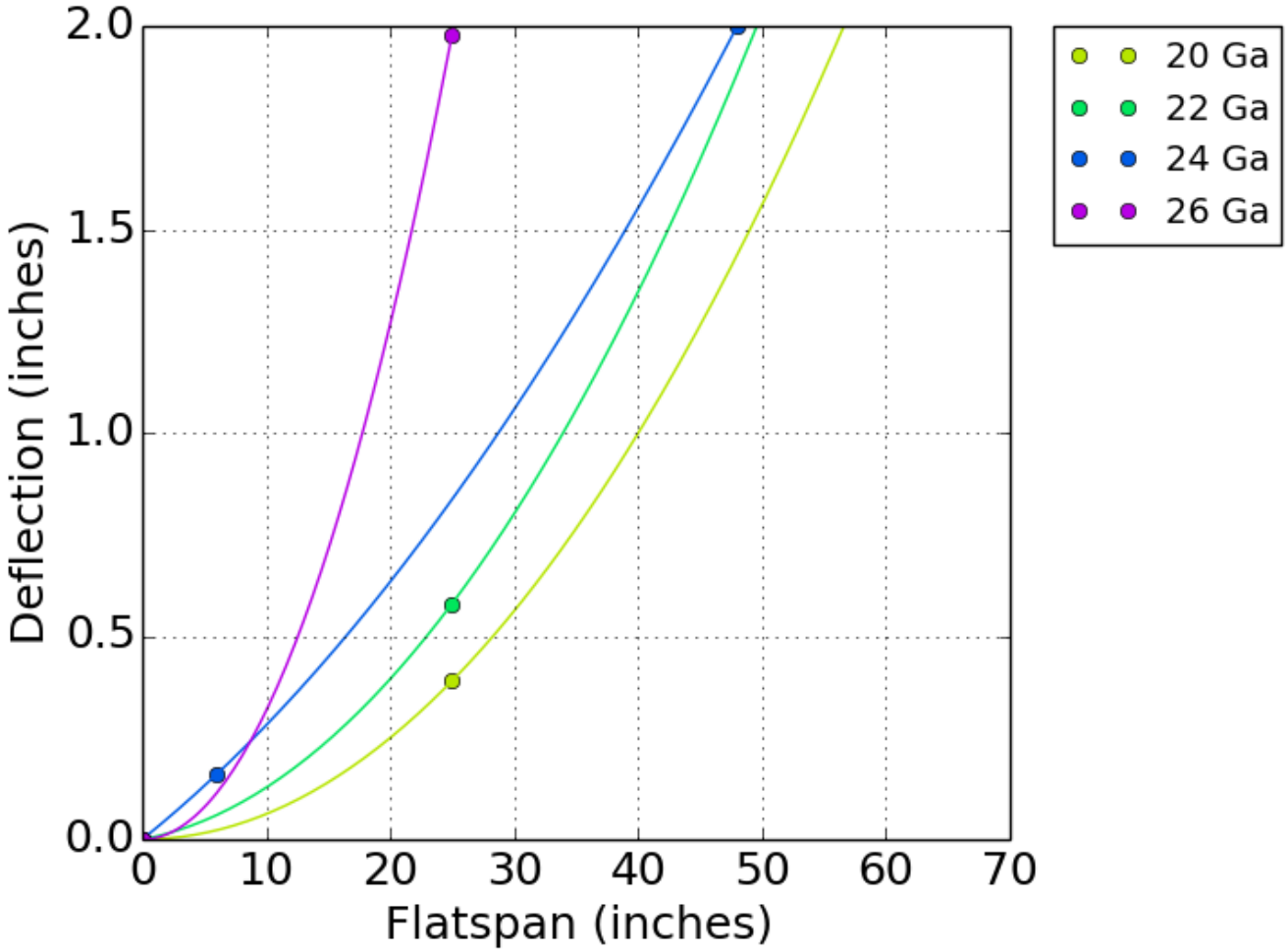


Figure 8.10.: T-25 12ft lengths Flat Span vs Deflection for 1 inWG

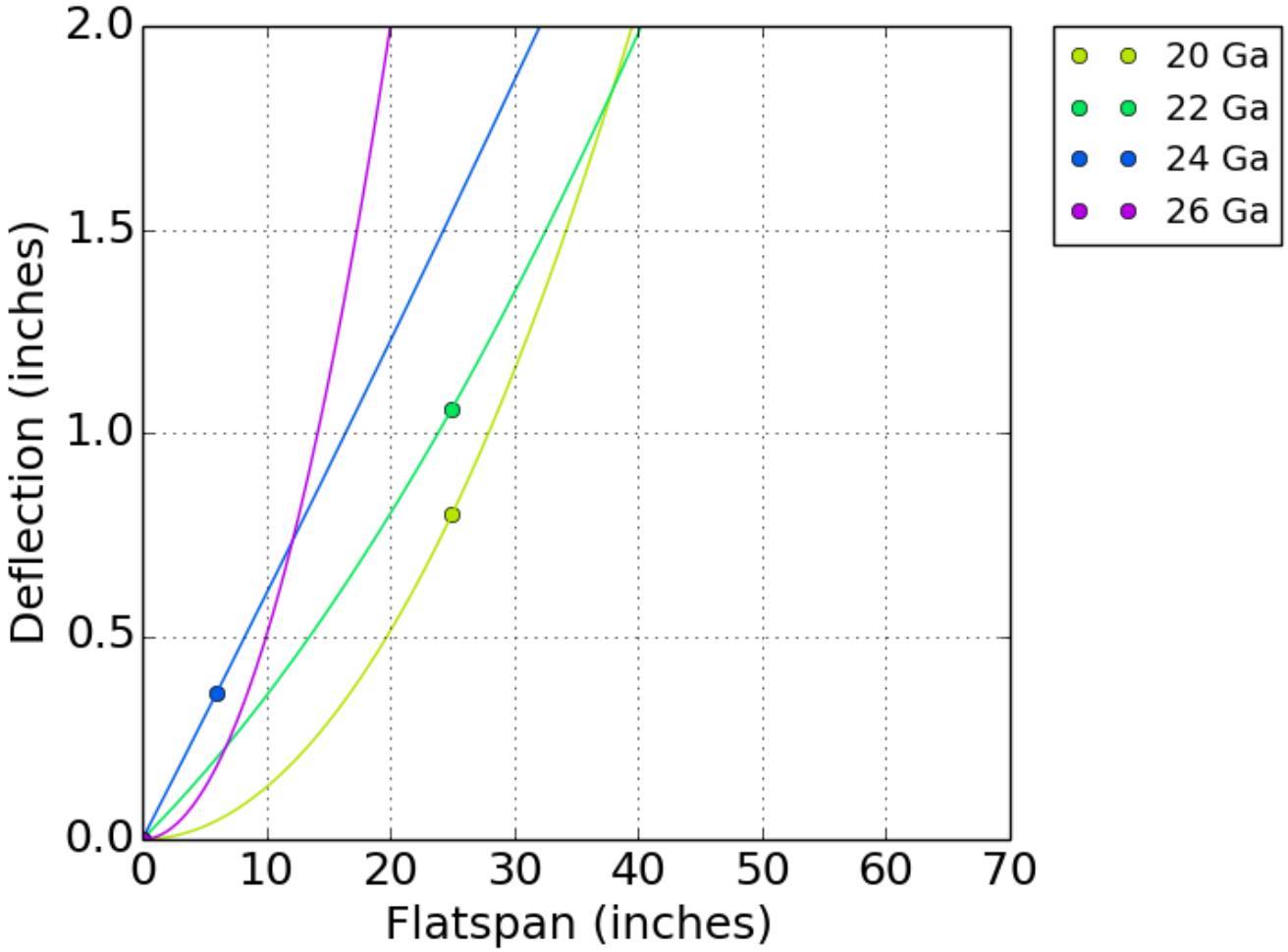


Figure 8.11.: T-25 12ft lengths Flat Span vs Deflection for 2 inWG

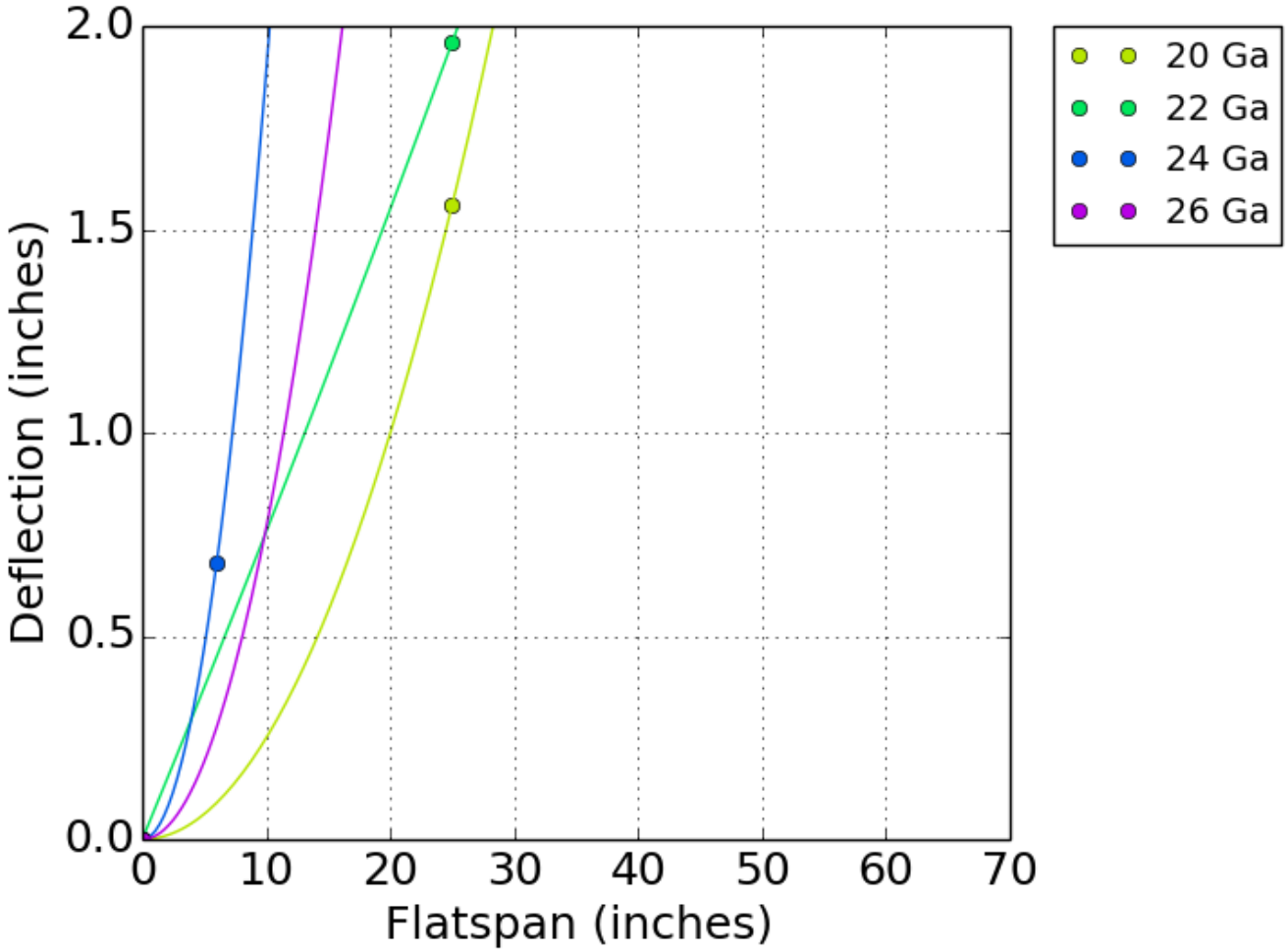


Figure 8.12.: T-25 12ft lengths Flat Span vs Deflection for 4 inWG

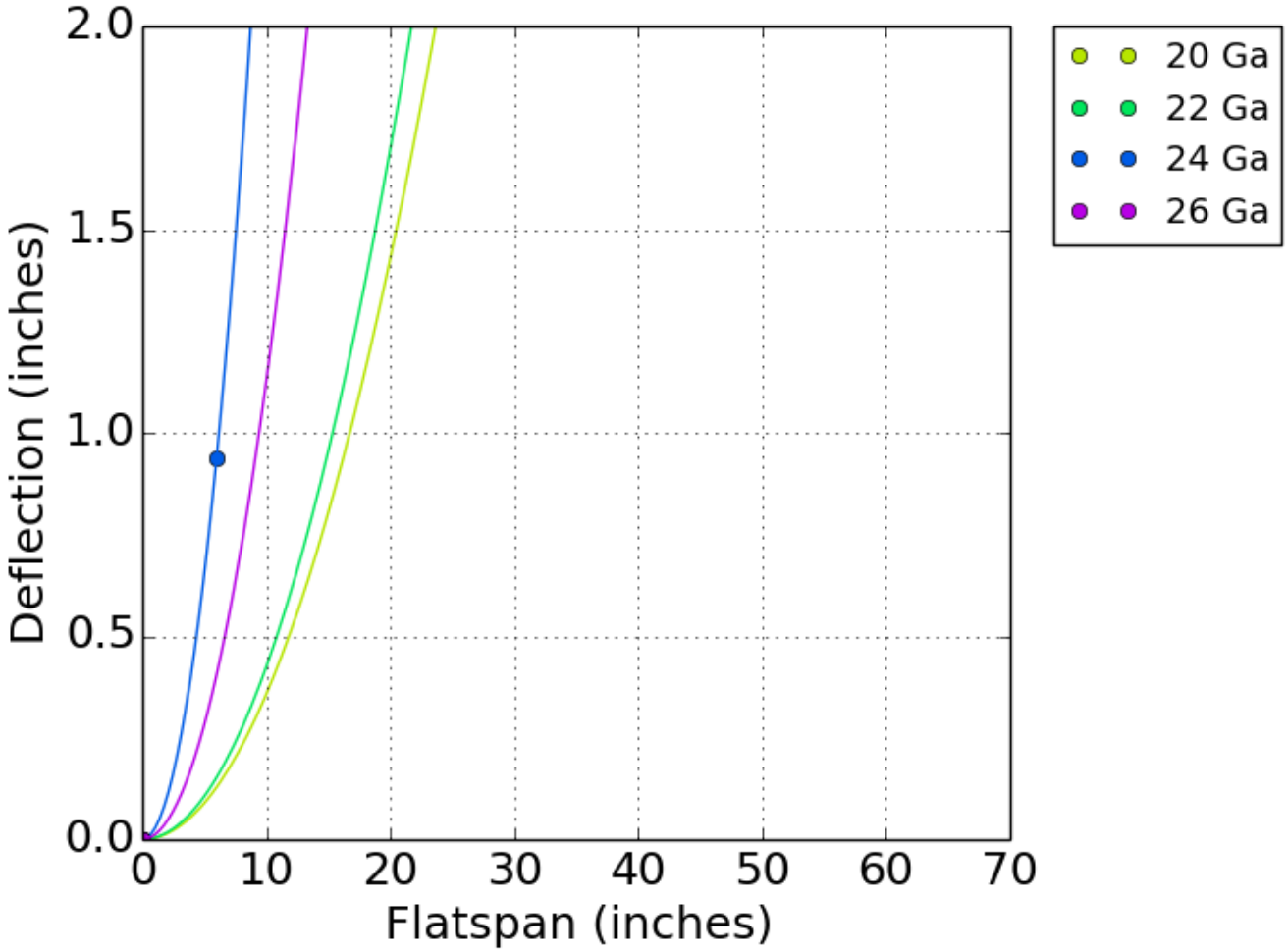


Figure 8.13.: T-25 12ft lengths Flat Span vs Deflection for 6 inWG

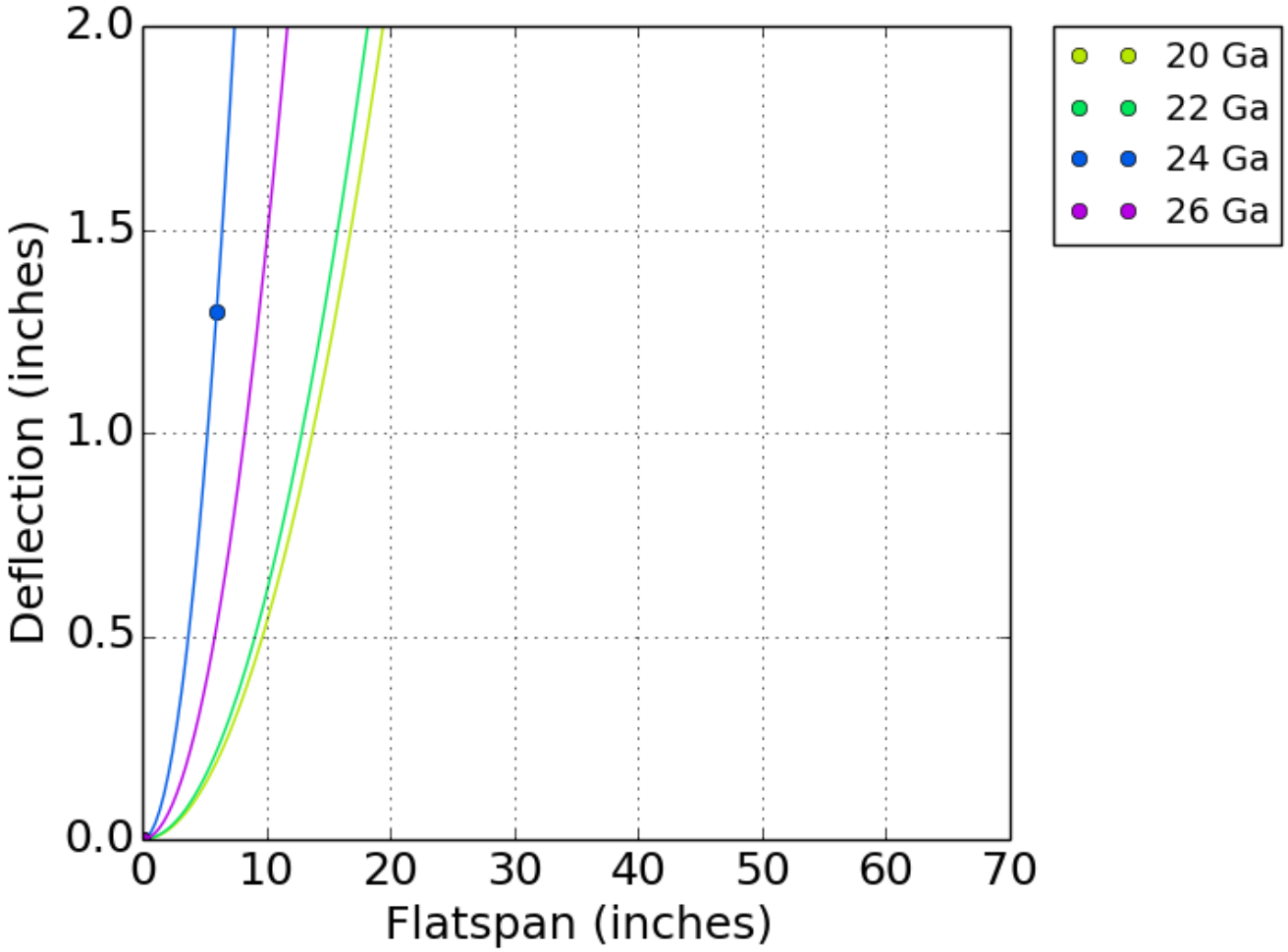


Figure 8.14.: T-25 12ft lengths Flat Span vs Deflection for 10 inWG

8.2 Summary

Flat span verses deflection data for each gauge at a constant pressure for small deflections, in the 0 to 2 inch range, is important because it represents a deflection limit consistent with the limited space found in ceilings and overheads. In this regard, these plots are important for developing a spiral-duct standard.

9. EMPIRICAL MODELING APPROACH AND DEVELOPMENT

Three fourth-dimensional empirical models were developed from the experimental data with these models covering unreinforced positive and negative pressured ducts, along with T-25 positive pressure ducts. Empirical models are important for viewing and analyzing data interpolations.

9.1 Empirical Modeling Approach and Development

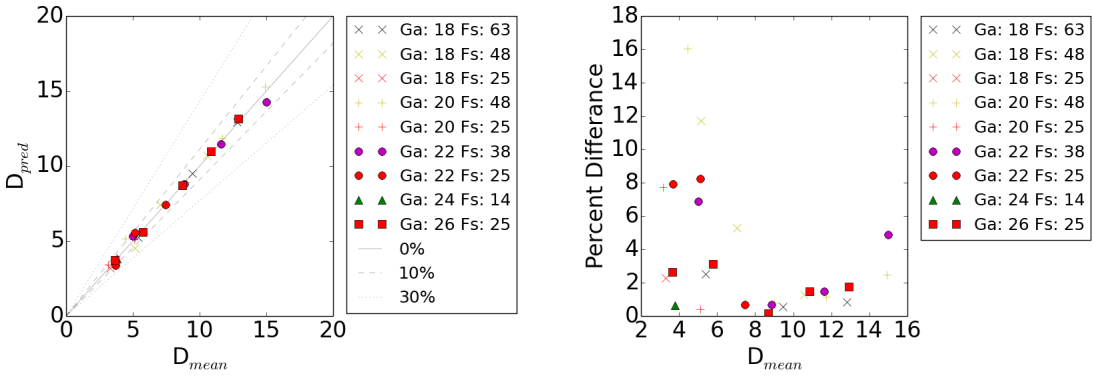
The regression model has three inputs and one output making it a fourth-dimensional object. The three inputs are pressure (P), gauge (Ga), and flat span (Fs). The equation outputs deflection (D). The valid range for the positive unreinforced model is the pressure must be between 1.0 inWG and 10 inWG and the deflection must remain between 2.75 inches and 25 inches. Another regression model is valid for negative unreinforced where pressure is between -6.0 inWG and -1.0 inWG and the deflection remains within -9 inches and -1 inches. Pressure must be between 1.0 inWG and 10inWG and the deflection must remain between 1.25 inches and 25 inches for the T-25 empirical model to remain valid. Manufacturing and assembly differences defined the range for each model. A fish mouth could occur during duct assembly with a slip in collar. Even with a small fish mouth, a phenomenon similar to oil canning can occur during initial pressurization, which could cause a near instantaneous deflection of up to 4 inches. Figure 9.1a, 9.2a, and 9.3a show experimental data verses empirical model. Figure 9.1b compares the model to experimental data, which is approximately 20%. Similarly, the unreinforced negative, Figure 9.2b shows around 25% and T-25 positive, Figure 9.3b shows about 16%. Equation 9.1 represents the 4D best fit polynomial regression used for the model. Table 9.1 shows the coefficients used for each model. In this table, C_9 is zero due to

Table 9.1.: Empirical Model Coefficients

	Unreinforced P	Unreinforced N	T-25 12ft lengths P
C_1	22.81	21.22	41.59
C_2	-2.5	-1.78	-5.41
C_3	-1.13	-0.46	1.07
C_4	1.25	-3.75	-0.67
C_5	0.06	0.02	-0.03
C_6	0.02	0.19	0.07
C_7	0.02	0.04	0.0
C_8	0.05	0.03	0.14
C_9	0.0	0.0	-0.0
C_{10}	-0.1	-0.01	-0.04

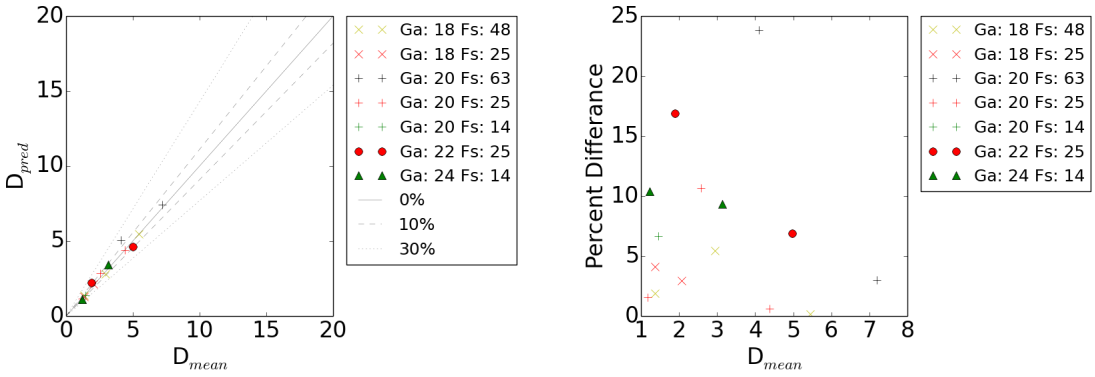
rounding, but is kept because it decreases percent difference of the empirical model and experimental data. The figures in Appendix B.2 on page 114 show a comparison of the model to the raw data. The individual points represent the raw data, whereas the curves represent the model. The data points are near the empirical model curves, which also suggests the empirical model is a good representation of the experimental data.

$$\begin{aligned} d(Ga, Fs, P) = C_1 + C_2 Ga + C_3 Fs + C_4 P + C_5 GaFs \\ + C_6 FsP + C_7 FsP + C_8 Ga^2 + C_9 Fs^2 + C_{10} P^2 \end{aligned} \tag{9.1}$$



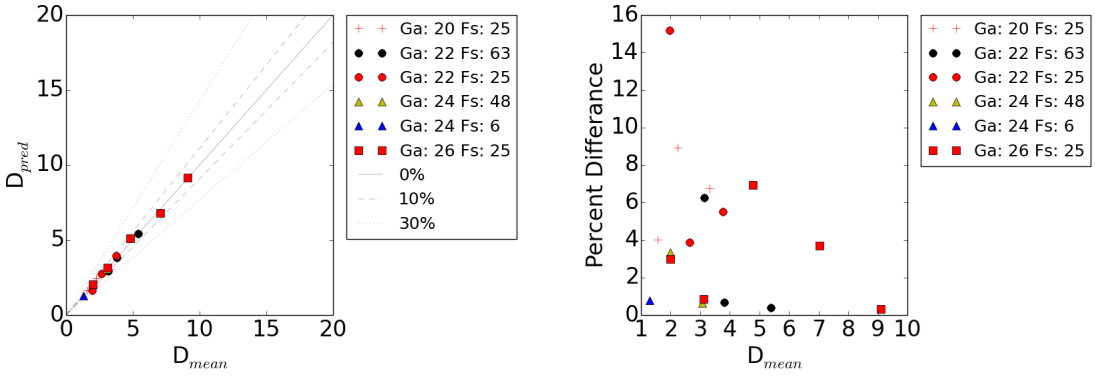
(a) Data vs Model (b) Deflection vs Percent Difference

Figure 9.1.: Unreinforced Positive Pressure Accuracy



(a) Data vs Model (b) Deflection vs Percent Difference

Figure 9.2.: Unreinforced Negative Pressure Accuracy



(a) Data vs Model (b) Deflection vs Percent Difference

Figure 9.3.: T-25 Positive Pressure Accuracy

9.2 Summary

Three empirical models were developed from experimental data. One model is for unreinforced positive pressure, another for unreinforced negative pressure, and the final for T-25 positive pressure. The empirical model of unreinforced positive pressure represented the experimental data within 20%, when the pressure is between 1.0 inWG and 10 inWG and the deflection is between 2.75 inches and 25 inches. The empirical model for unreinforced negative pressure represented the experimental data within 25%, when the pressure is between -6.0 inWG and -1.0 inWG and the deflection is between -9 inches and -1 inches. The empirical model for T-25 positive pressure represented the experimental data within 16%, when the pressure is between 1.0 inWG and 10 inWG and the deflection is between 1.25 inches and 25 inches.

10. DESIGN AND STANDARD PREPARATION USING THE EMPIRICAL MODEL

From the empirical models, contour plots and tables were developed. These contour plots and contour tables reveal trends important to the preparation for the development of a spiral duct standard. These empirical models were validated using percent difference between the empirical model and the experimental data.

10.1 Design and Standard Preparation Using the Empirical Model

The contour graphs (Figures 10.1 through 10.6) and contour tables (Tables 10.1 through 10.11) shows a definite trend of lower deflection for shorter flat span and thicker gauge at a given pressure. These also show a higher pressure for a shorter flat span and thicker gauge at a given deflection. The blank places in the tables are outside of the model's range. The contour graphs and tables reveal another feature: as pressure increases, the deflection increases and as deflection increases, pressure increases. This observation shows not only the model trends with the data, but the model also makes logical sense.

Table 10.1 shows a well fit model. This is the result of a large quantity of samples and large deflections. Table 10.9 also shows a well fit model. Table 10.3 shows a less well fit model. Part of the tables seem to show thinner gauge and wider flat span cause less deflection, this comes from the quadratic part of the regression. Figure 9.2b shows this model is the worst fit of the three, but still represents the data well. This could be due to a small amount of samples, but probably comes from properties of the negative pressure test. A negative pressure unreinforced test has unique properties, which may cause the duct can collapse until it touches itself without causing a leak. These data points were considered failed and pulled from

Table 10.1.: Positive Unreinforced Deflections in inches at Constant Pressure

(a) Pressure 4 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	10	16	22		
48	4	9	14	19	24
38		5	9	13	17
25			3	6	9
14					
6					

(b) Pressure 6 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	13	19			
48	7	12	17	22	
38	4	8	11	16	20
25		3	6	8	11
14					4
6					

(c) Pressure 10 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	17	24			
48	11	15	20		
38	7	10	14	18	23
25	3	5	7	10	13
14				4	6
6					

the data set; there was no correctional area, this could affect the model.

Figure 10.9 shows the effect of the T-25 reinforcing. The lines on the graph are nearly horizontal, which shows the gauge of the duct has less affect than it did in the unreinforced models. This same reaction can be seen in Figure 10.11. Even with the reinforcement a hard elbow appears on the graph, which shows the weakness of the 26 gauge as compared to the rest of the graph.

Table 10.3.: Negative Unreinforced Deflections in inches at Constant Pressure

(a) Pressure -1 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-1				
48	-2				
38	-2	-1			
25	-1	-2	-2	-2	-1
14		-2	-2	-3	-3
6		-2	-3	-3	-4

(b) Pressure -2 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-3	-2			
48	-3	-2	-2	-1	
38	-3	-3	-2	-2	-1
25	-2	-3	-3	-3	-3
14	-1	-2	-3	-4	-4
6		-2	-3	-4	-5

(c) Pressure -4 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-7	-7	-6	-6	-4
48	-6	-6	-6	-6	-6
38	-5	-6	-6	-7	-7
25	-3	-5	-6	-7	-7
14	-2	-4	-5	-7	-8
6		-2	-5	-6	-8

(d) Pressure -6 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63					
48					
38	-7	-9			
25	-5	-7	-9		
14	-2	-5	-7		
6		-3	-6	-9	

Table 10.5.: Positive Unreinforced Pressures in inches at a Constant Deflection

(a) Deflection 3 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	1				
48	3	1			
38	5	3	1		
25	8	6	4	2	
14	9	9	9	7	5
6	8	8	8	9	9

(b) Deflection 4 in

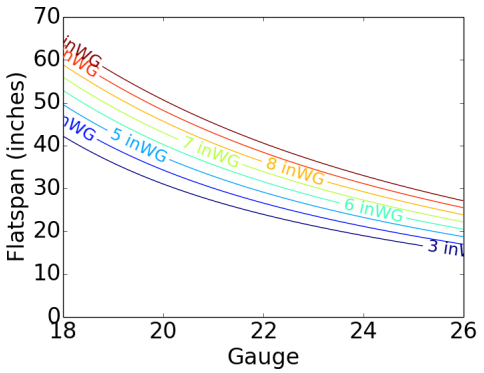
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	2				
48	4	2			
38	6	3	1		
25	10	7	4	3	1
14	9	9	9	9	6
6	8	8	8	9	9

(c) Deflection 5 in

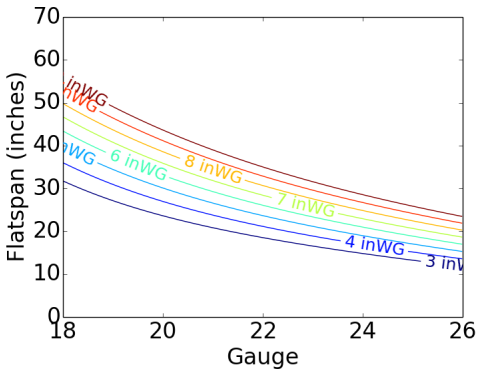
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	2				
48	4	2			
38	6	4	2		
25	10	9	5	3	2
14	9	9	9	9	7
6	8	8	8	9	9

(d) Deflection 6 in

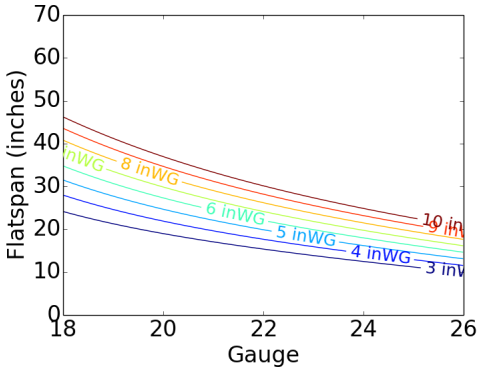
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	2				
48	5	2			
38	8	4	2		
25	10	10	6	4	2
14	9	9	9	9	10
6	8	8	8	9	9



(a) Pressure 4 inWG

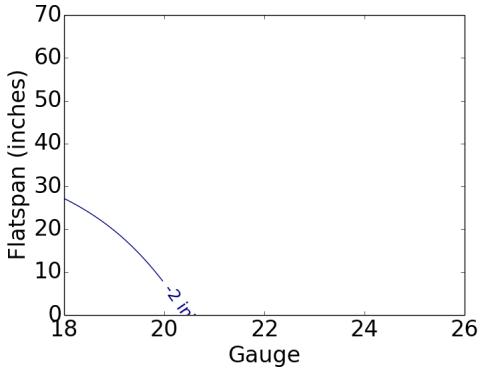


(b) Pressure 6 inWG

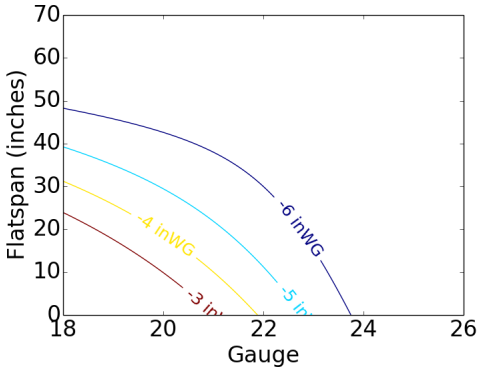


(c) Pressure 10 inWG

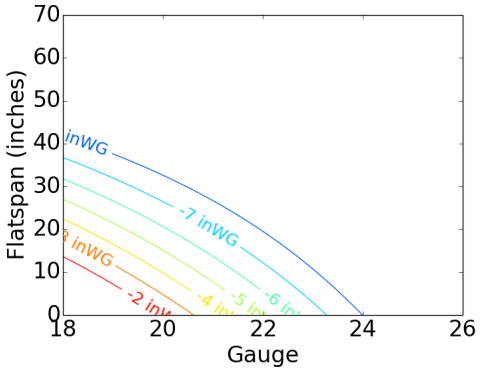
Figure 10.1.: Positive Unreinforced Deflections in inches at Constant Pressure



(a) Pressure -2 inWG



(b) Pressure -4 inWG



(c) Pressure -6 inWG

Figure 10.2.: Negative Unreinforced Deflections in inches at Constant Pressure

Table 10.7.: Negative Unreinforced Pressures in inches at a Constant Deflection

(a) Deflection -1 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63		-2	-2	-3	-3
48		-1	-2	-2	-2
38			-1	-2	-2
25					-1
14					
6					

(b) Deflection -2 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-1	-2	-2	-3	-3
48	-1	-2	-2	-2	-3
38	-2	-2	-2	-2	-2
25	-2	-1	-1	-1	-2
14	-6	-2			
6		-2			

(c) Deflection -3 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-2	-2	-3	-3	-4
48	-2	-2	-2	-3	-3
38	-2	-2	-2	-2	-3
25	-4	-2	-2	-2	-2
14	-6	-3	-2	-1	-1
6		-5	-2		

(d) Deflection -4 in

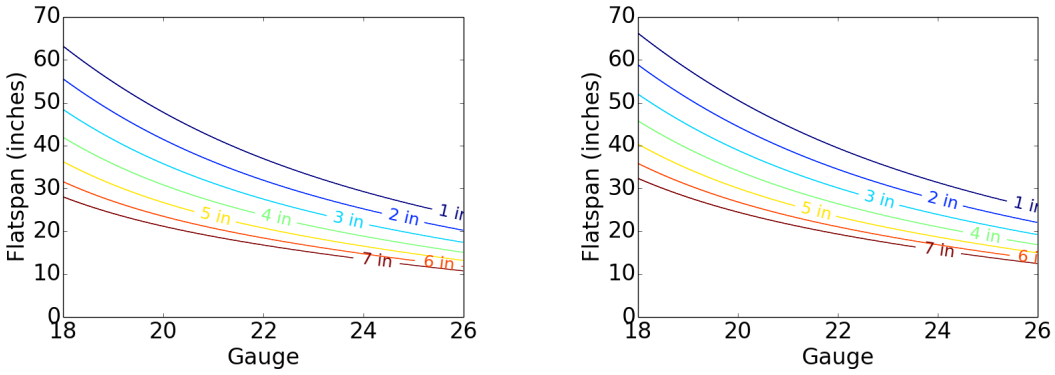
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-2	-3	-3	-4	-4
48	-3	-3	-3	-3	-3
38	-3	-3	-3	-3	-3
25	-5	-3	-3	-2	-2
14	-6	-5	-3	-2	-2
6		-6	-3	-2	-1

(e) Deflection -5 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-3	-3	-4	-4	-4
48	-3	-3	-3	-4	-4
38	-4	-4	-3	-3	-3
25	-6	-4	-3	-3	-3
14	-6	-6	-4	-3	-2
6		-6	-4	-3	-2

(f) Deflection -6 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	-3	-4	-4	-4	-4
48	-4	-4	-4	-4	-4
38	-5	-4	-4	-4	-4
25	-6	-5	-4	-4	-3
14	-6	-6	-5	-4	-3
6		-6	-6	-4	-3



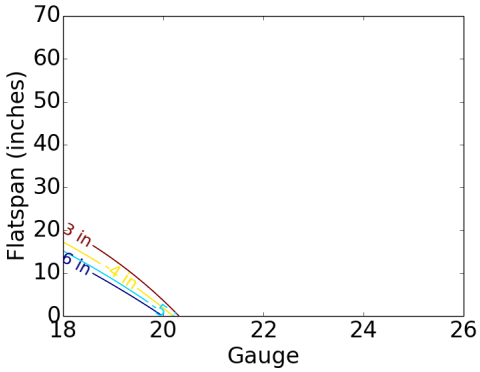
(a) Deflection 3 in

(b) Deflection 4 in

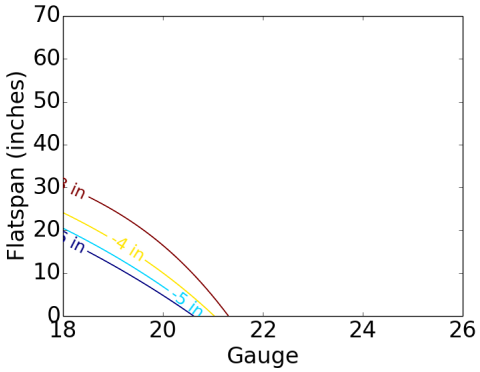
(c) Deflection 5 in

(d) Deflection 6 in

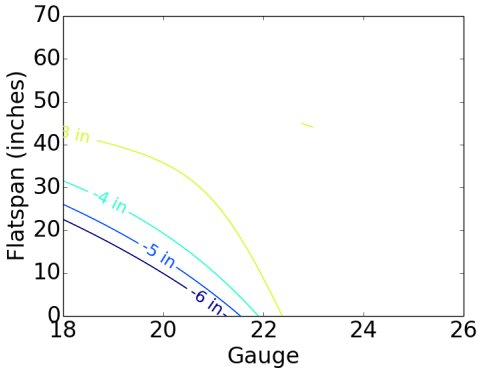
Figure 10.3.: Positive Unreinforced Pressure(inWG) at a Constant Deflection



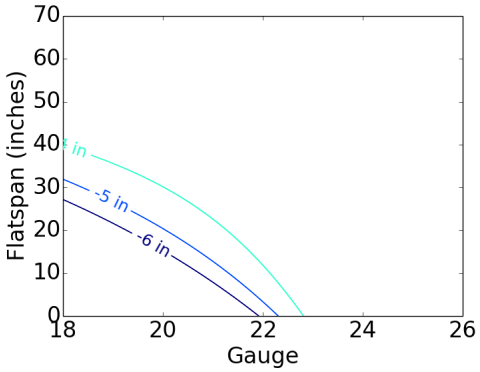
(a) Deflection -2 in



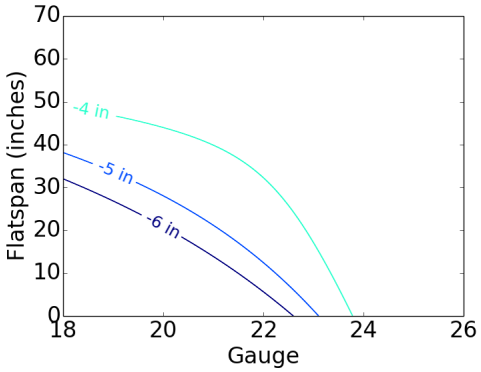
(b) Deflection -3 in



(c) Deflection -4 in



(d) Deflection -5 in



(e) Deflection -6 in

Figure 10.4.: Negative Unreinforced Pressure(inWG) at a Constant Deflection

Table 10.9.: Positive T-25 Deflections in inches at Constant Pressure

(a) Pressure 1 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	10	6	3	1	
48	8	5	3	2	2
38	5	3	2	2	3
25	1				2
14					
6					

(b) Pressure 2 inWG

Flatspan (inches)	Gauge				
	18	20	22	24	26
63	10	6	4	2	2
48	8	5	4	3	4
38	6	4	3	3	4
25	2				3
14					2
6					

(c) Pressure 4 inWG

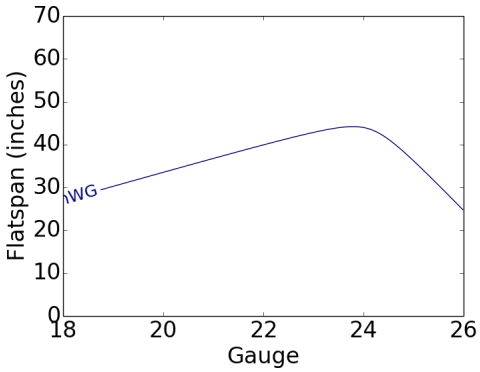
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	11	8	5	4	4
48	9	6	5	5	6
38	7	5	4	4	6
25	3	2	2	3	5
14					4
6					2

(d) Pressure 6 inWG

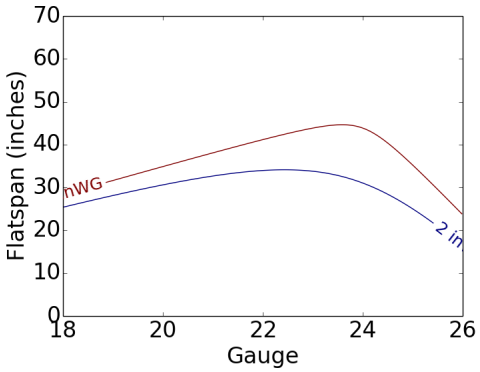
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	12	9	7	6	6
48	10	7	6	6	8
38	7	6	5	6	8
25	3	2	3	4	7
14				2	5
6					3

(e) Pressure 10 inWG

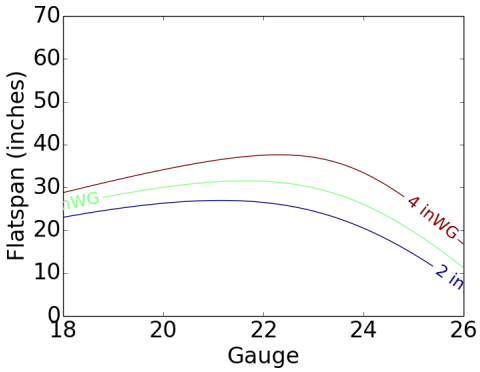
Flatspan (inches)	Gauge				
	18	20	22	24	26
63	12	10	8	8	9
48	10	8	8	8	10
38	8	6	6	8	10
25	3	3	4	6	9
14				4	7
6				1	6



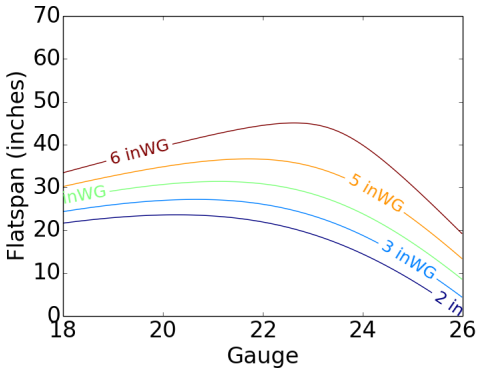
(a) Pressure 1 inWG



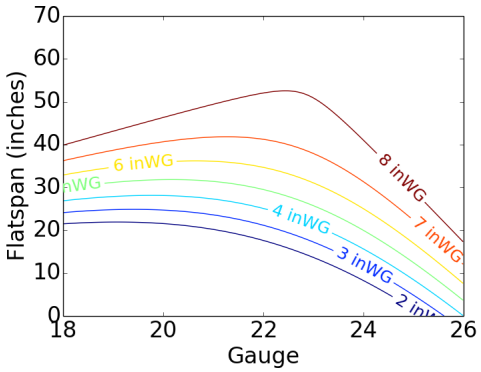
(b) Pressure 2 inWG



(c) Pressure 4 inWG



(d) Pressure 6 inWG



(e) Pressure 10 inWG

Figure 10.5.: Positive T-25 Deflections in inches at Constant Pressure

Table 10.11.: Positive T-25 Pressure(inWG) at a Constant Deflection

(a) Deflection 2 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63				2	2
48				1	
38			1	1	
25	2	5	5	3	1
14	8	10	10	6	2
6	8	10	10	10	4

(b) Deflection 3 in

Flatspan (inches)	Gauge				
	18	20	22	24	26
63			1	3	3
48			1	2	1
38		1	2	2	1
25	5	9	7	4	2
14	8	10	10	8	3
6	8	10	10	10	5

(c) Deflection 4 in

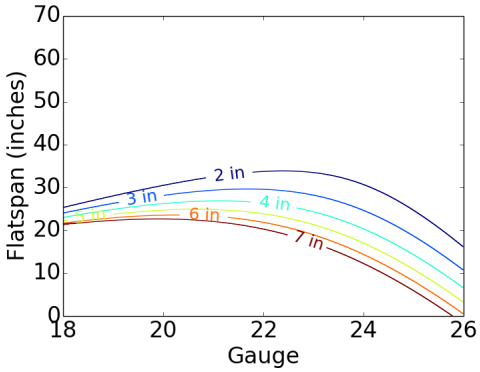
Flatspan (inches)	Gauge				
	18	20	22	24	26
63			2	4	4
48			2	3	2
38		2	4	4	2
25	8	10	10	6	3
14	8	10	10	10	4
6	8	10	10	10	7

(d) Deflection 5 in

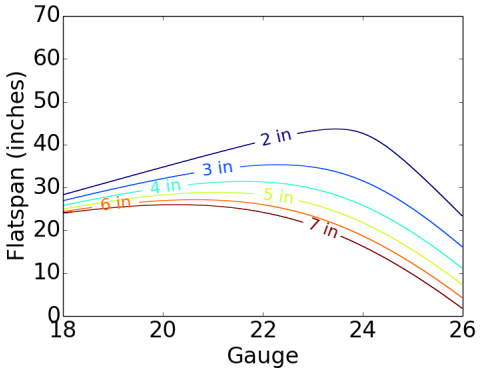
Flatspan (inches)	Gauge				
	18	20	22	24	26
63			3	5	5
48		2	4	4	3
38		4	6	5	3
25	8	10	10	8	4
14	8	10	10	10	6
6	8	10	10	10	9

(e) Deflection 6 in

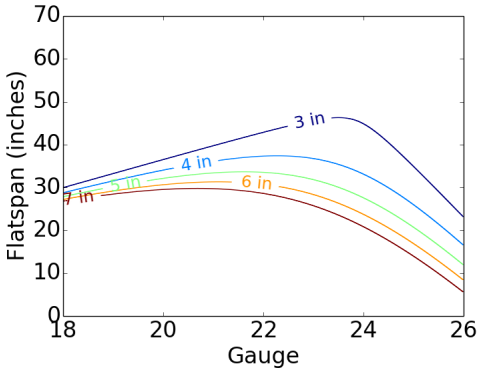
Flatspan (inches)	Gauge				
	18	20	22	24	26
63		1	5	6	6
48		3	5	6	4
38	2	7	8	6	4
25	8	10	10	10	5
14	8	10	10	10	7
6	8	10	10	10	10



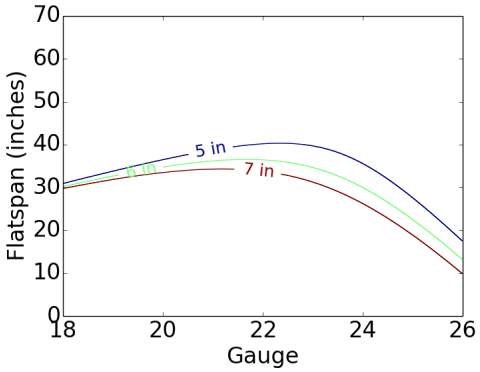
(a) Deflection 2 in



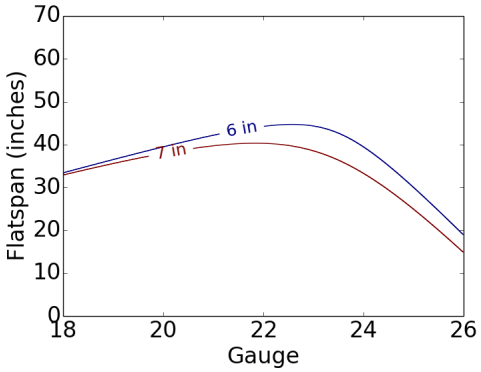
(b) Deflection 3 in



(c) Deflection 4 in



(d) Deflection 5 in



(e) Deflection 6 in

Figure 10.6.: Positive T-25 Pressure(inWG) at a Constant Deflection

10.2 Summary

From the empirical model, contour plots and tables were developed. These plots and tables show that lower deflections are caused by shorter flat spans and thicker gauges. The T-25 plots show deflection is affected less by gauge than flat span.

11. EXPERIMENTAL AND MODELING ANALYSIS FOR OTHER REINFORCEMENTS AND HEIGHT COMPARISON

Qualitative analysis was performed for the other reinforcements, which is consistent with the limited data available. The ratios in these models were obtained by dividing the experimental data with the empirical model. Thesis ratios aided in identifying those reinforcements affecting deflection the most. The height comparison was an attempt to determine a method for transversing different heights or scales of duct.

11.1 Experimental and Modeling Analysis for Other Reinforcements Using the Ratio Method

Standard curve fitting with a polynomial could not be performed for most reinforcements, due to the limited test data available. The first method is to curve fit based on reinforcement spacing; insufficient data made this unsuccessful. Generating a ratio based on the unreinforced duct is the next attempt; the inconsistency of the ratio constants determined this approach not reliable. Quantitative analysis could draw no conclusions; therefore, qualitative analysis will be used.

The ratio method is an attempt to find a multiplier for each type of reinforcement. The ratio was developed by dividing the experimental data with the unreinforced empirical model. Figure 11.1 shows this for the positive ducts. Figure 11.2 shows this for the negative ducts. Any value greater than one means the unreinforced model has less deflection than the reinforced model. The reinforcement has more effect on the deflection as the ratio approaches zero. The ratios are too sporadic to make a multiplier. Interesting conclusions can be drawn from these graphs, such as the effectiveness of the reinforcements. The heavier the gauge and the

shorter the flat span, the less the reinforcement affects the duct. When comparing reinforcements the same duct needs to be compared, meaning the same flat span and gauge. Also, the same reinforcing spacing needs to be compared. This can be seen best in Figures 11.1b and 11.1c. Figure 11.1c shows trapeze reinforcing (P6) is better than attached reinforcing (P4). Similarly Figure 11.1c shows tie rod reinforcing (P9) was better than trapeze reinforcing. This concurs with the conclusions drawn earlier in Section 7.2.

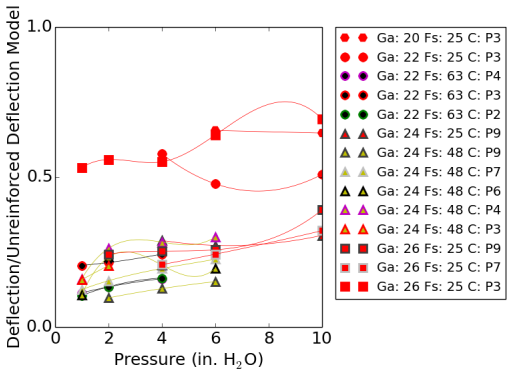
Table 11.1 and Table 11.2 are sorted by pressure, then gauge, then flat span, and finally ratio. E represents the experimental data, M represents the unreinforced model, and R represents the ratio of the experimental data and the unreinforced model. Sorting the table in this way reveals the better reinforcement for a constant, pressure, gauge, and flat span. The 24 gauge and 48 flat span in Table 11.1 shows tie rod is the best, followed by trapeze, and the worst of the three is the attached reinforcing.

Table 11.1.: Average Ratio Positive

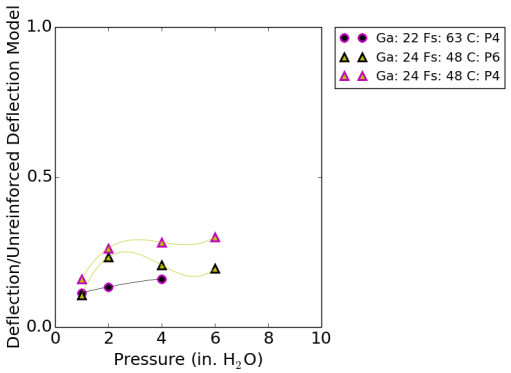
Config	Gauge	Flatspan	Pressure	E	M	R
Attached reinforcing 3ft centers	22	63	1	1.74	15.24	0.11
T-25 6ft lengths	22	63	1	1.6	15.24	0.11
T-25 12ft lengths	22	63	1	3.13	15.24	0.21
Trapeze 3ft centers	24	48	1	1.36	12.66	0.11
Trapeze 6ft centers	24	48	1	1.58	12.66	0.12
Attached reinforcing 3ft centers	24	48	1	2.04	12.66	0.16
T-25 12ft lengths	24	48	1	2.0	12.66	0.16
Attached reinforcing 3ft centers	22	63	2	2.37	17.73	0.13
T-25 6ft lengths	22	63	2	2.38	17.73	0.13
T-25 12ft lengths	22	63	2	3.81	17.73	0.21
Tie Rod 6ft centers	24	48	2	1.45	14.92	0.1
Trapeze 6ft centers	24	48	2	2.3	14.92	0.15
T-25 12ft lengths	24	48	2	3.06	14.92	0.21
Trapeze 3ft centers	24	48	2	3.47	14.92	0.23
Attached reinforcing 3ft centers	24	48	2	3.91	14.92	0.26
Tie Rod 6ft centers	26	25	2	1.36	5.59	0.24
T-25 12ft lengths	26	25	2	3.12	5.59	0.56
Attached reinforcing 3ft centers	22	63	4	3.56	22.11	0.16
T-25 6ft lengths	22	63	4	3.61	22.11	0.16
T-25 12ft lengths	22	63	4	5.38	22.11	0.24
Tie Rod 6ft centers	24	48	4	2.41	18.81	0.13
Trapeze 6ft centers	24	48	4	3.67	18.81	0.2
Trapeze 3ft centers	24	48	4	3.88	18.81	0.21
Attached reinforcing 3ft centers	24	48	4	5.32	18.81	0.28
Trapeze 6ft centers	26	25	4	1.81	8.7	0.21
Tie Rod 6ft centers	26	25	4	2.2	8.7	0.25
T-25 12ft lengths	26	25	4	4.79	8.7	0.55
Tie Rod 6ft centers	24	48	6	3.31	21.89	0.15
Trapeze 3ft centers	24	48	6	4.27	21.89	0.2
Trapeze 6ft centers	24	48	6	4.94	21.89	0.23
Attached reinforcing 3ft centers	24	48	6	6.56	21.89	0.3
Trapeze 6ft centers	26	25	6	2.67	11.0	0.24
Tie Rod 6ft centers	26	25	6	2.83	11.0	0.26
T-25 12ft lengths	26	25	6	7.04	11.0	0.64
Trapeze 6ft centers	26	25	10	4.22	13.14	0.32
Tie Rod 6ft centers	26	25	10	5.13	13.14	0.39
T-25 12ft lengths	26	25	10	9.11	13.14	0.69

Table 11.2.: Average Ratio Negative

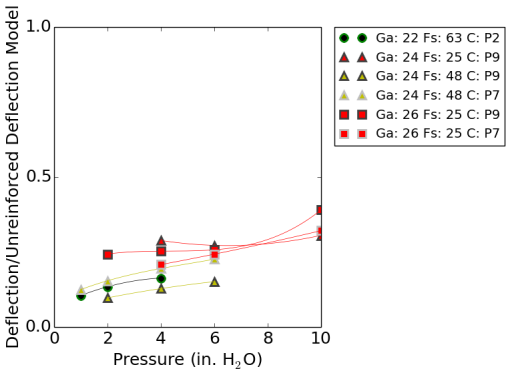
Config	Gauge	Flatspan	Pressure	E	M	R
Tie Rod 6ft centers	24	25	1	1.25	1.88	0.67
Trapeze 3ft centers	24	48	1	3.95	6.69	0.59
Trapeze 6ft centers	24	48	1	6.36	6.69	0.95
T-25 12ft lengths	20	25	2	1.23	1.16	1.06
T-25 12ft lengths	22	25	2	2.28	2.21	1.03
Tie Rod 6ft centers	24	25	2	2.38	3.51	0.68
T-25 12ft lengths	20	25	4	2.98	2.84	1.05
T-25 12ft lengths	22	25	4	4.08	4.64	0.88
Tie Rod 6ft centers	24	25	4	3.18	6.69	0.48
Trapeze 6ft centers	18	25	6	1.44	2.12	0.68
T-25 12ft lengths	20	25	6	4.18	4.41	0.95
T-25 12ft lengths	22	25	6	5.26	6.95	0.76
Trapeze 3ft centers	24	6	6	1.4	2.8	0.5
T-25 12ft lengths	24	6	6	1.61	2.8	0.57
Attached reinforcing 6ft centers	24	6	6	1.69	2.8	0.6



(a) All spacings

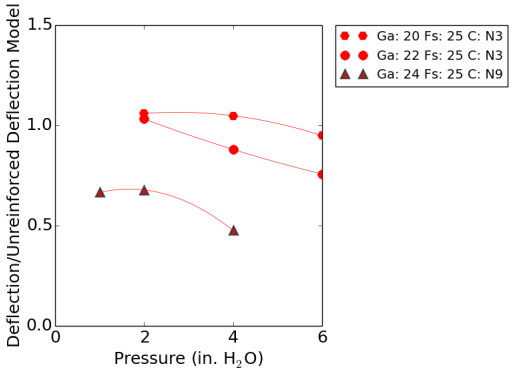


(b) 3 ft spacing

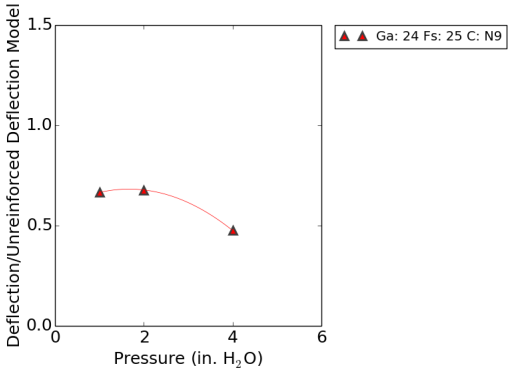


(c) 6 ft spacing

Figure 11.1.: Positive Ratio



(a) All spacings



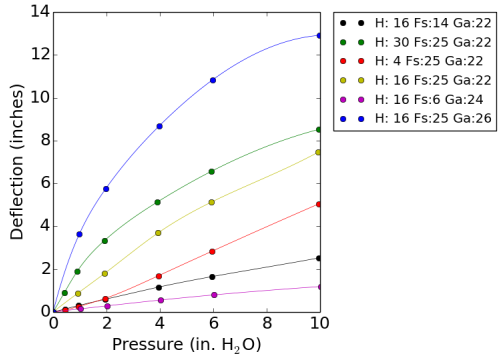
(b) 6 ft spacing

Figure 11.2.: Negative Ratio

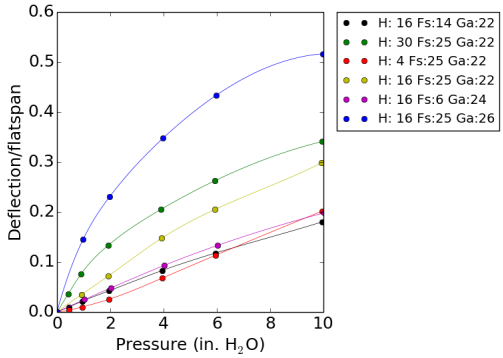
11.2 Height Comparison of Unreinforced Ducts

Determining a method of transversing different heights or scales is important for two main reasons. First it saves of the amount of specimens needed to be tested and second it reduces the number of tables needed in a standard.

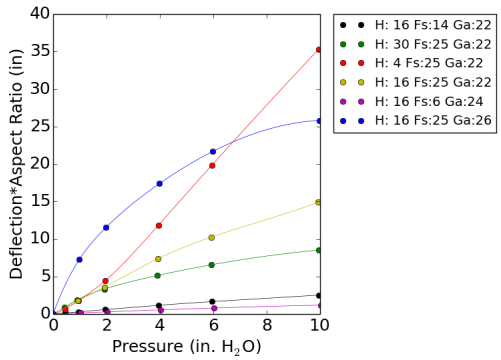
The lack of similarities means deflection could not be normalized based on flat span. Figure 11.3b contains three similar lines, but these three lines have no relation; the lines have different heights, flat spans, widths, and aspect ratios. The ducts shown in these graphs are based on similar flat spans and aspect ratios. Figures 11.3 and Figure 11.4 are normalized using parameters: height, flat span, and aspect ratio. No solid conclusion can be drawn from any of these normalizations due to the lack of trends. Figure 11.4, which compares negative height, has less lines and less graphs due to only testing three ducts which fit the comparison criteria and also show no conclusions can be drawn. More testing needs to be done before a correlation can be formed for the change in height.



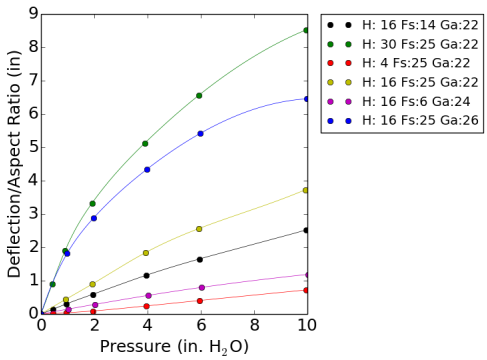
(a) Height Deflection Comparison



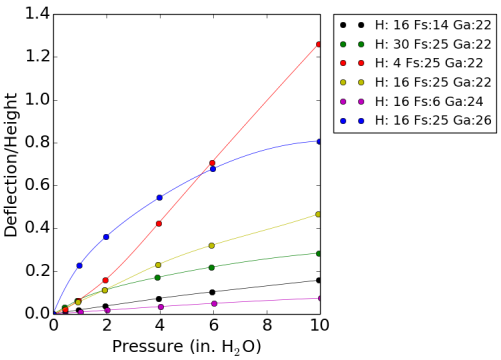
(b) Deflection Divided by Flat Span



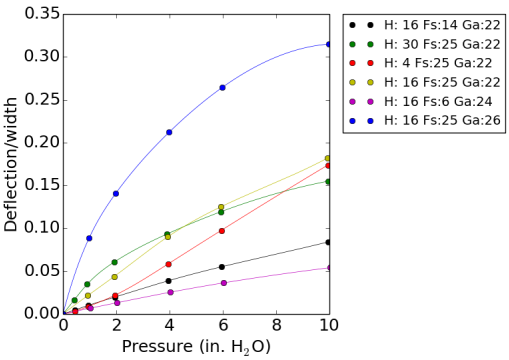
(c) Deflection Multiplied by Aspect Ratio



(d) Deflection Divided by Aspect Ratio

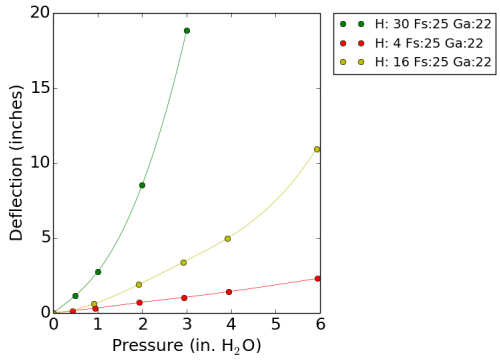


(e) Deflection Divided by Height

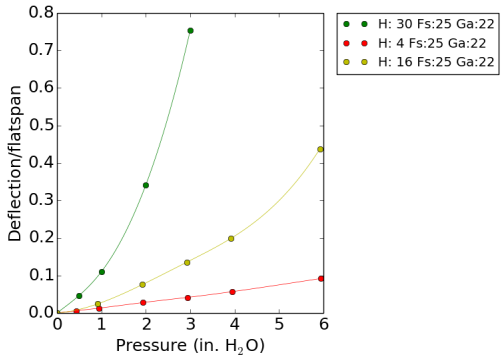


(f) Deflection Divided by Width

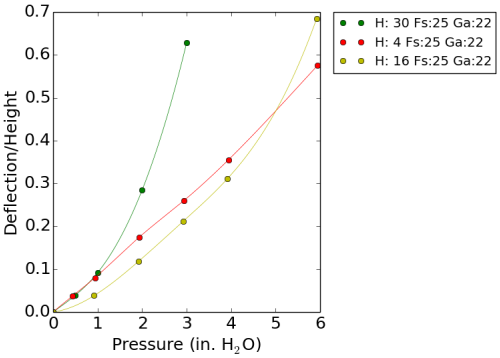
Figure 11.3.: Positive Pressure Unreinforced Height Comparison



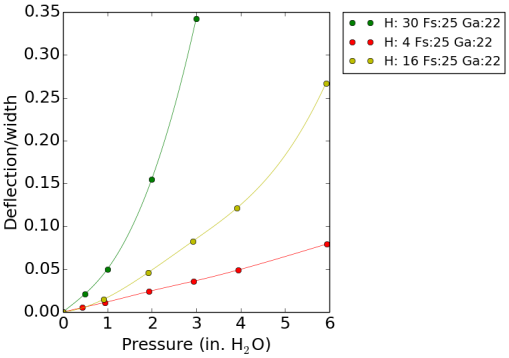
(a) Height Deflection Comparison



(b) Deflection Divided by Flat Span



(c) Deflection Divided by Height



(d) Deflection Divided by Width

Figure 11.4.: Negative Pressure Unreinforced Height Comparison

11.3 Summary

Due to the insufficient amount of data for quantitative analysis, qualitative analysis was used. The ratio method was an attempt to find a multiplier, though it didn't accomplish it's purpose it; however, revealed other interesting features. The heavier the gauge and the shorter the flat span, the less the reinforcement affects the duct. The reinforcements can be ordered from best to worst: tie rod, trapeze, and attached reinforcing. Transversing different heights or scales is important, because it reduces the number of tables needed. More testing needs to be done before a correlation can be formed for the change in height.

12. FUTURE WORKS

Improvements and additions could be made to this project in the future. Instrumentation could be improved on the current setup. With little or no modification, this setup would also work for additional tests beyond its intended design.

12.1 Instrumentation Upgrades

Changing measuring methods and upgrading the program could improve the instrumentation of the current test facility. While trying to achieve maximum range, the 10 inch potentiometers cause difficulty when testing the ducts. Then when they run out have to manually record the tape measures. String potentiometers would allow for a longer range of measurement, around 54 inches. This would be enough to eliminate the tape measures and increase the speed of setup and testing. The range difference in the size of ducts requires the PID controller to be extremely slow. A fuzzy logic controller could improve the speed of the control system, reducing time to reach the desired pressure. Furthermore, automating the system with a fuzzy logic controller would also reduce training for the operators. Both future works would require little modification to the current test facility, yet greatly improve its functionality.

12.2 Additional Tests

With little modification of the test facility, many other tests could be run, such as duct leakage testing, superimposed load testing, puncture testing, and impact testing. A duct leakage testing setup could be obtained with little modification to the test facility. The test facility already has a method of holding the duct and a blower setup. Duct leakage is a concern with HVAC, due to the percent

leakage allowed and energy efficiency.[24] Other tests are durability tests instead of efficiency tests. One of these test is deflection under superimposed loads, which is very similar to the deflection test, but the difference being, the duct is required to support weight. The load can be a concentrated load, an exterior uniform load, or a simulated uniform load.[4] Additionally, puncture testing could be performed with this test facility, which requires a plunger of a given configuration to be dropped on the specimen. This structure could hold the plunger to required specifications and drop it on a specimen.[4] Furthermore, the test facility could be used for impact testing, which requires dropping a specific type of sandbag on the duct carefully. The truss could be retrofitted to hold the sandbag for this test.[4] Every purposed test could be performed with little modification to the test facility.

12.3 Summary

The improvement simplifying operating procedure of this test facility reduces operator training time, and in turn reducing operating costs. Broadening the test facility's test spectrum, increases possible revenue from the setup.

13. CONCLUSION

The development of a standard for flat-oval spiral ducting is important to the HVAC industry because it minimizes the pounds of metal used, consequently reducing cost. To support the development of this standard, a test facility consisting of a truss structure, instrumentation, a blower system, and data acquisition and control system was designed and built. This setup used linear potentiometers to measure deflection to an accuracy of 0.011 inch and pressure transducers to measure pressure to an accuracy of 0.025 inWG. The test matrix consisted of 64 ducts of the same length, but with differing flat spans and gauges. Specifically, the flat spans varied from 6 to 63 inches, while the gauge varied from 18 to 26 gauge. These ducts were tested by moving both sections into the truss support structure and then installing the collars, end caps, and instrumentation. The duct was then pressurized or evacuated to collect the deflection data. From this data, a pressure versus deflection graph was generated for each test. These graphs showed that thinner gauges, longer flat spans, higher pressures, and larger reinforcement spacings all increased deflection. The data results also showed the order from least deflection to most deflection for reinforcements: tie rod, trapeze, attached reinforcing, and unreinforced. Empirical models were developed from the data for unreinforced positive-pressure ducts, unreinforced negative-pressure ducts, and T-25 positive-pressure ducts. Verification showed the models typically represented the experimental data within a 25% difference. The experimental analysis consisted of 0 to 2 inch deflection as a function of flat span for a range of gauge and pressures. This experimental analysis is especially important because it forms the basis for a duct deflection standard.

REFERENCES

[1] ANSI/ASHRAE/IESNA. ANSI/ASHRAE/IESNA standard 90.1-2007 energy standard for buildings except low-rise residential buildings, 2007.

[2] ASHRAE. ASHRAE standard energy-efficient design of low-rise residential buildings, 2007.

[3] ASHRAE. ASHRAE handbook - fundamentals (si edition), 2009.

[4] ASHRAE/SMACNA. *Method of Testing HVAC Air Ducts and Fittings*, 2008.

[5] Thomas G. Beckwith, Roy D. Marangoni, and John H Lienhard V. *Mechanical Measurements*. Prentice Hall, 6 edition, 2007.

[6] Dayton Electric Mfg. Co. Dayton® high pressure belt drive radial blade blowers. 4C129, 4C130, and 4C131.

[7] Duane R Condon, Thomas L Kendall, Larry D Brown, Scott W Randolph, and Dennis L Hart. Method and apparatus for pressure testing pipe lines, December 2 2003. US Patent 6,655,413.

[8] Miller Electric Training Department. *Gas Metal Arc Welding*. Appleton, Wisconsin, 1994.

[9] Gary E Driesen, Dilip A Amin, and TT Wu. Structural integrity justification of a rectangular duct under large external pressure. Technical report, Westinghouse Savannah River Co., Aiken, SC (United States), 1998.

[10] Dwyer Instruments, Inc. Series 616 differential pressure transmitter, 2010.

[11] M.A. Froning, G.C. Moyers, M.S. Price, and D.R. Whitehead. Spiral duct ovalizer, December 14 1999. US Patent 6,000,260.

[12] Jerry D Laging. Apparatus for pressure testing pipe, March 25 1980. US Patent 4,194,389.

[13] Lamont R McClain. Deflection-resistant safety flange for integrally-flanged duct sections, Oct 1994. US Patent 5,358,013.

[14] Omega®. Lp803/lp804, 2008.

[15] SMACNA. *High Pressure Duct Construction Standards*. Chantilly, Virginia, 3 edition, 1975.

[16] SMACNA. *Low Pressure Duct Construction Standards*. Chantilly, Virginia, 5 edition, 1976.

[17] SMACNA. *Round Industrial Duct Construction Standards*. Chantilly, Virginia, 1 edition, 1977.

[18] SMACNA. *Fibrous Glass Duct Construction Standards*. Chantilly, Virginia, 5 edition, 1979.

[19] SMACNA. *Rectangular Industrial Duct Construction Standards*. Chantilly, Virginia, 1 edition, 1980.

[20] SMACNA. *HVAC Systems Duct Design*. Chantilly, Virginia, 2 edition, 1981.

[21] SMACNA. *HVAC Systems Duct Design Tables & Charts*. Chantilly, Virginia, 3 edition, 1981.

[22] SMACNA. *HVAC Duct Construction Standards Metal and Flexible*. Chantilly, Virginia, 2 edition, 1995.

[23] SMACNA. *HVAC Duct Construction Standards Metal and Flexible*. Chantilly, Virginia, 2 edition, 1995.

[24] SMACNA. *HVAC Air Duct Leakage Test Manual*. Chantilly, Virginia, 2 edition, 2012.

[25] Patrick J. Smolinski and Gregory S. Palmer. Flat oval duct modeling by finite element analysis. Master's thesis, University of Pittsburgh, 2005.

[26] SPIDA. Designer's guide standards and dimensional data for air ducts and fittings as manufactured by members of SPIDA, 2010.

[27] SPIDA. SPIDA tolerance standards, 2013.

[28] John H Stratton. Metal duct selection and application. *ASHRAE journal*, June 2000.

[29] Warren C. Young and Richard G. Budynas. *Roark's Formulas for Stress and Strain*. McGraw-Hill Professional Publishing, 7 edition, 2002.

APPENDIX A

TEST FACILITY APPENDIX

Figure A.1.: Tape Measure and 10 inch Potentiometer

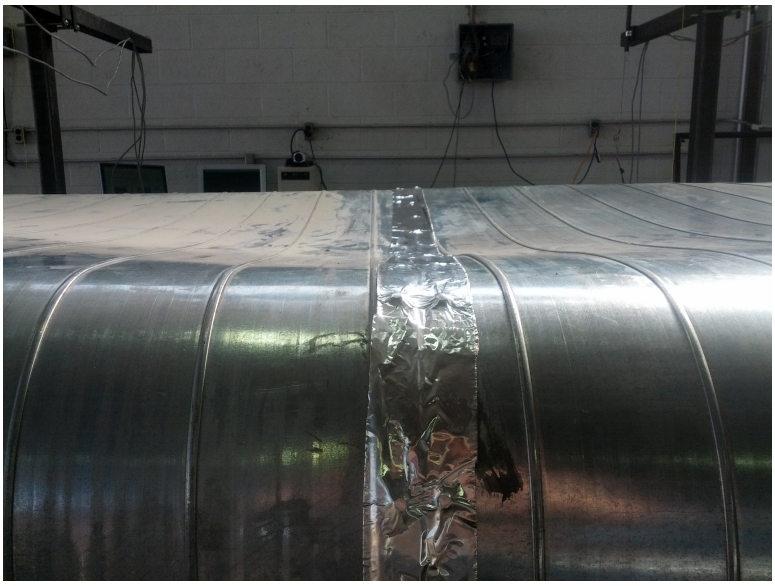


Figure A.2.: Sealed Collar

APPENDIX B

DATA ANALYSIS

B.1 Raw Data

B.1.1 Unreinforced

Table B.1.: 22 Gauge and 38 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	1.4
1.0	2.6
2.0	5.0
4.0	8.9
6.0	11.6
10.0	15.0

Table B.2.: Unreinforced Positive

(a) 24 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.2
2.0	0.3
4.0	0.6
6.0	0.8
10.0	1.2

(b) 20 Gauge and 48 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	1.1
1.0	2.2
2.0	4.4
4.0	9.0
6.0	11.7
9.9	14.9

(c) 26 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	3.6
2.0	5.8
4.0	8.7
6.0	10.8
10.0	12.9

(d) 24 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.3
0.9	0.5
1.9	1.0
3.9	1.9
5.9	2.6
9.9	3.8

(e) 18 Gauge and 48 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.6
1.0	1.2
2.0	2.7
4.0	5.2
6.0	7.0
10.0	10.6

(f) 18 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.3
2.0	0.7
4.0	1.4
6.0	2.1
10.0	3.3

Table B.3. (cont.)

(a) (cont.) 22 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.9	0.9
1.9	1.8
3.9	3.7
5.9	5.1
9.9	7.5

(b) (cont.) 22 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.0
1.0	0.1
2.0	0.1
4.0	0.2
6.0	0.4
10.0	0.6

(c) (cont.) 18 Gauge and 63 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.8
1.0	2.4
2.0	5.4
4.0	9.4
6.0	12.8

(d) (cont.) 20 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.2
0.9	0.5
1.9	1.0
3.9	2.2
5.9	3.2
9.9	5.1

(e) (cont.) 20 Gauge and 14 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.1
0.9	0.2
1.9	0.4
3.9	0.8
5.9	1.2
9.9	1.8

(f) (cont.) 22 Gauge and 14 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.1
0.9	0.3
1.9	0.6
3.9	1.2
5.9	1.6
9.9	2.5

Table B.5.: Unreinforced Negative

(a) 24 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.2
1.0	0.5
2.0	1.2
3.0	2.1
4.0	3.1
6.0	12.8

(b) 26 Gauge and 48 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	6.6
1.0	9.4
2.0	11.7
3.0	13.0
4.0	13.5
6.0	13.8

(c) 20 Gauge and 14 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.1
1.0	0.2
2.0	0.4
3.0	0.6
4.0	0.9
5.9	1.5

(d) 18 Gauge and 48 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.7
1.0	1.4
2.0	2.9
3.0	4.4
4.0	5.4
6.0	10.8

(e) 18 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.3
2.0	0.7
3.0	1.0
4.0	1.4
6.0	2.1

Table B.6. (cont.): Unreinforced Negative Continued

(a) (cont.) 22 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.9	0.6
1.9	1.9
2.9	3.4
3.9	5.0
5.9	10.9

(b) (cont.) 20 Gauge and 63 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	1.9
0.9	4.1
1.9	7.2
2.9	10.9
3.9	10.9
5.9	14.1

(c) (cont.) 18 Gauge and 14 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.0
1.0	0.1
2.0	0.1
3.0	0.2
4.0	0.3
6.0	0.5

(d) (cont.) 22 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.0
1.0	0.1
2.0	0.2
3.0	0.2
4.0	0.3
6.0	0.5

(e) (cont.) 20 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.3
1.0	0.6
2.0	1.2
3.0	1.8
4.0	2.6
6.0	4.4

Table B.8.: Unreinforced 4 in and 30 in height

(a) 22 Gauge, 25 inch Flat Span, and 4 inch Height Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.2
0.9	0.3
1.9	0.7
2.9	1.0
3.9	1.4
5.9	2.3

(b) 22 Gauge, 25 inch Flat Span, and 30 inch Height Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	1.2
1.0	2.8
2.0	8.5
3.0	18.8
4.0	23.1
6.0	24.3

(c) 22 Gauge, 25 inch Flat Span, and 4 inch Height Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.1
0.9	0.2
1.9	0.6
3.9	1.7
5.9	2.8
9.9	5.0

(d) 22 Gauge, 25 inch Flat Span, and 30 inch Height Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.4	0.9
0.9	1.9
1.9	3.3
3.9	5.1
5.9	6.6
9.9	8.5

B.1.2 Transverse Reinforcing

Table B.10.: 22 Gauge and 63 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	1.0
1.0	1.6
2.0	2.4
4.0	3.6
6.0	4.6

Table B.11.: T-25 Negative

(a) 22 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.4
1.0	0.9
2.0	2.3
3.0	3.4
4.0	4.1
6.0	5.3

(b) 20 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.3
1.0	0.6
2.0	1.2
3.0	2.1
4.0	3.0
6.0	4.2

(c) 24 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.2
2.0	0.4
3.0	0.7
4.0	0.9
6.0	1.6

Table B.13.: T-25 Positive

(a) 24 Gauge and 6 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.2
2.0	0.4
4.0	0.7
6.0	0.9
10.0	1.3

(b) 22 Gauge and 63 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	3.1
2.0	3.8
4.0	5.4

(c) 22 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.3
1.0	0.6
2.0	1.1
4.0	2.0
6.0	2.6
10.0	3.8

(d) 20 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.2
1.0	0.4
2.0	0.8
4.0	1.6
6.0	2.2
9.9	3.3

(e) 24 Gauge and 48 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	2.0
2.0	3.1

(f) 26 Gauge and 25 inch Flat Span

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	2.0
2.0	3.1
4.0	4.8
6.0	7.0
10.0	9.1

B.1.3 Trapezed Reinforcing

Table B.15.: Trapeze 3ft

(a) 24 Gauge and 6 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.1
2.0	0.2
3.0	0.5
4.0	0.8
6.0	1.4

(b) 24 Gauge and 48 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	2.3
1.0	4.0
2.0	4.9
3.0	5.1
4.0	5.2
5.6	6.4

(c) 24 Gauge and 6 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.0
2.0	0.1
4.0	0.2
6.0	0.2
9.9	0.4

(d) 24 Gauge and 48 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	1.0
1.0	1.4
2.0	3.5
4.0	3.9
6.0	4.3
9.9	5.0

Table B.17.: Trapeze 6ft

(a) 18 Gauge and 25 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.1
1.0	0.1
2.0	0.3
3.0	0.6
4.0	0.8
6.0	1.4

(b) 24 Gauge and 48 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	4.5
1.0	6.4
2.0	6.6
3.0	6.6
4.0	6.6
6.0	6.6

(c) 26 Gauge and 25 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.3
1.0	0.5
2.0	1.0
4.0	1.8
6.0	2.7
9.8	4.2

(d) 18 Gauge and 25 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.0
1.0	0.1
2.0	0.2
4.0	0.5
6.0	0.7
10.0	1.1

(e) 24 Gauge and 48 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.4
1.0	1.6
2.0	2.3
4.0	3.7
6.0	4.9

B.1.4 Attached Reinforcing

Table B.19.: Attached 3ft

(a) 24 Gauge and 6 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.0
2.0	0.1
3.0	0.2
4.0	0.4
6.0	0.6

(b) 24 Gauge and 48 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	2.0
2.0	3.9
4.0	5.3
6.0	6.6
10.0	9.2

(c) 24 Gauge and 6 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.1
2.0	0.1
4.0	0.2
6.0	0.3
10.0	0.4

(d) 22 Gauge and 63 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	1.7
2.0	2.4
4.0	3.6
6.0	4.9

Table B.21.: Attached 6ft

(a) 24 Gauge and 6 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.2
2.0	0.4
3.0	0.6
4.0	0.9
6.0	1.7

(b) 24 Gauge and 6 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.2
2.0	0.3
4.0	0.7
6.0	0.9
10.0	1.3

(c) 22 Gauge and 63 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	2.3
1.5	3.8
4.5	7.4

B.1.5 Tie Rod Reinforcing

Table B.23.: Tie Rod 3ft

(a) 22 Gauge and 63 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.5	0.4
1.0	0.8
2.0	1.3
3.0	1.6
4.0	1.3
6.0	1.4

(b) 24 Gauge and 48 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.9	0.8
1.9	0.9
3.9	0.8
5.9	1.2

Table B.25.: Tie Rod 6ft

(a) 24 Gauge and 6 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.1
2.0	0.2
3.0	0.3
4.0	0.3
6.0	0.4

(b) 24 Gauge and 25 inch Flat Span Negative Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	1.2
2.0	2.4
3.0	2.9
4.0	3.2
6.0	3.8

(c) 26 Gauge and 25 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.8
2.0	1.4
4.0	2.2
6.0	2.8
9.9	5.1

(d) 24 Gauge and 48 inch Flat Span Positive Pressure

Pressure (inWG)	Deflection (in)
0.0	0.0
0.9	0.9
1.9	1.4
4.0	2.4
5.9	3.3

(e) 24 Gauge and 6 inch Flat Span Positive Pressure

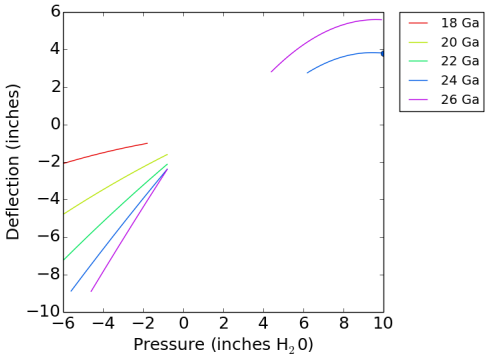
Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.1
2.0	0.2
4.0	0.3
6.0	0.4
10.0	0.6

(f) 24 Gauge and 25 inch Flat Span Positive Pressure

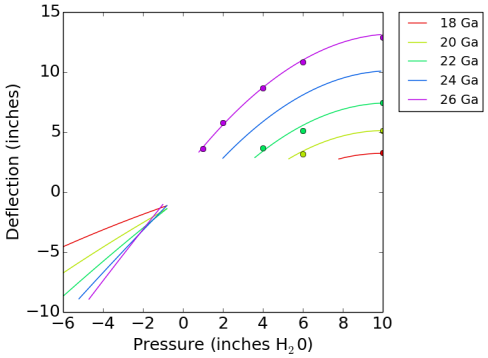
Pressure (inWG)	Deflection (in)
0.0	0.0
1.0	0.6
2.0	1.0
4.0	1.7
6.0	2.2
9.9	3.1

B.2 Polynomial Graphs

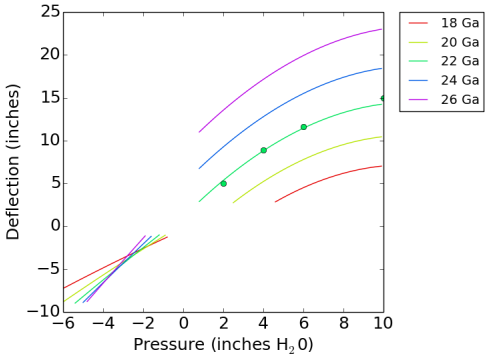
B.2.1 Unreinforced



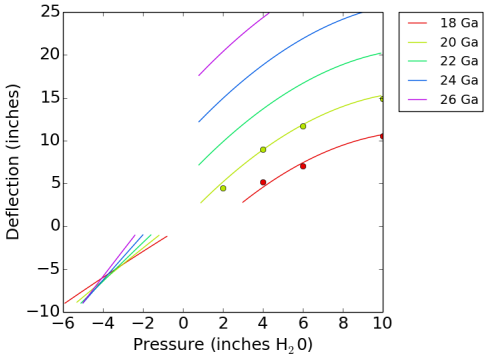
(a) 14 in



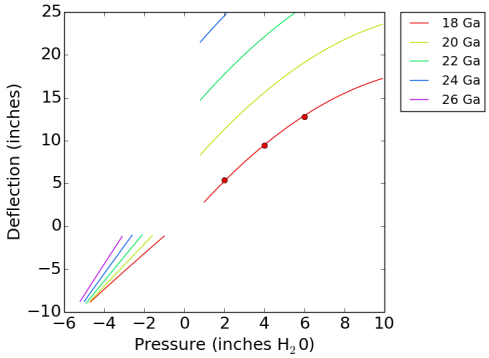
(b) 25 in



(c) 38 in

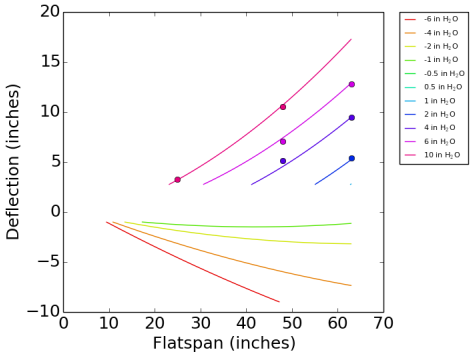


(d) 48 in

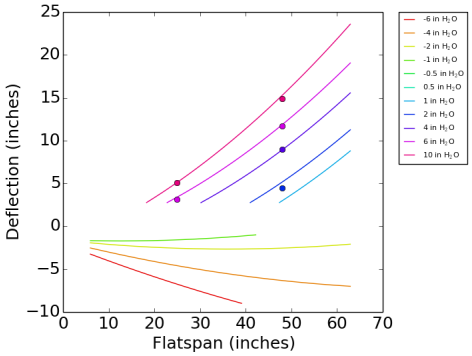


(e) 63 in

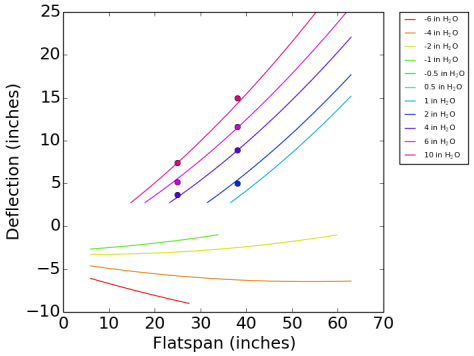
Figure B.1.: Unreinforced Pressure vs Deflection for a Constant Flat Span



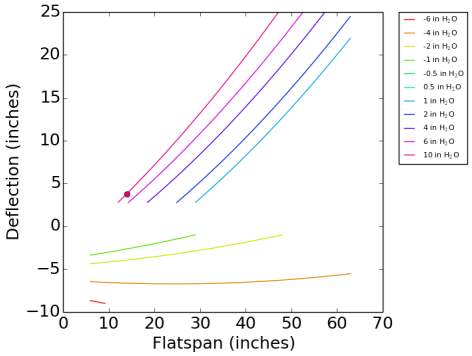
(a) 18 ga



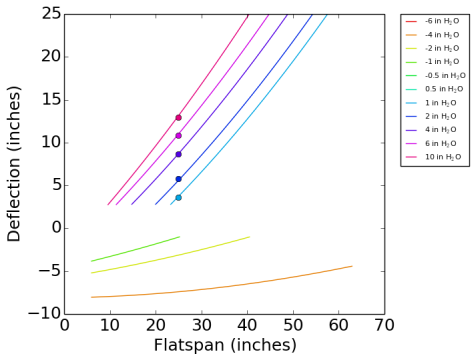
(b) 20 ga



(c) 22 ga

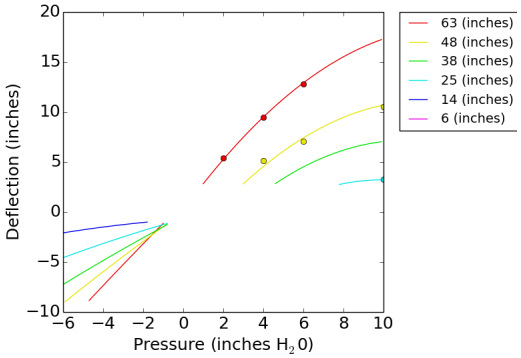


(d) 24 ga

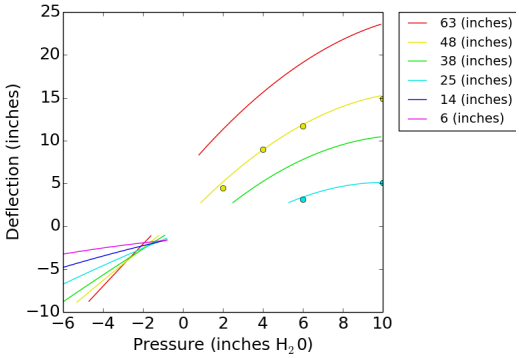


(e) 26 ga

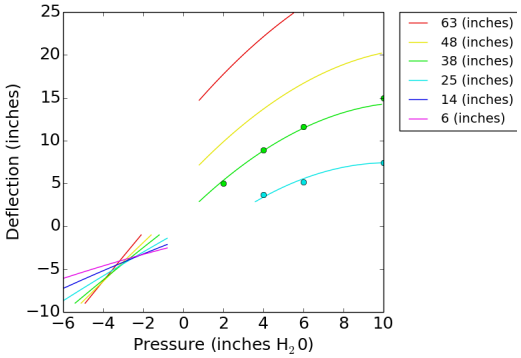
Figure B.2.: Unreinforced Flat Span vs Deflection for a Constant Gauge



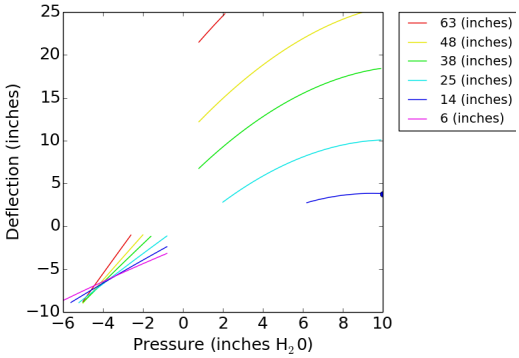
(a) 18 ga



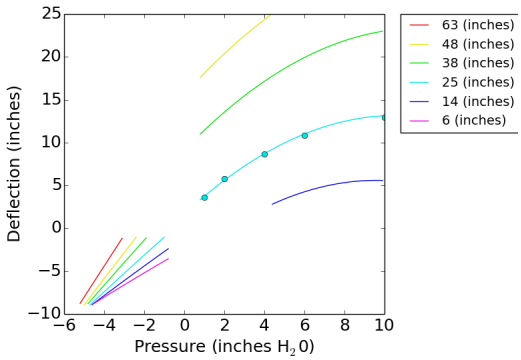
(b) 20 ga



(c) 22 ga

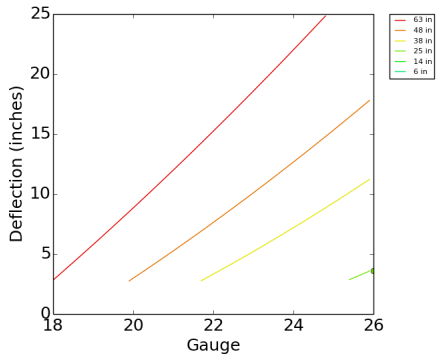


(d) 24 ga

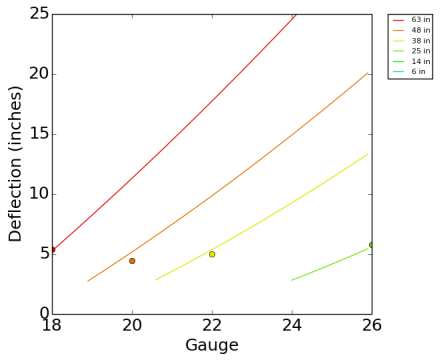


(e) 26 ga

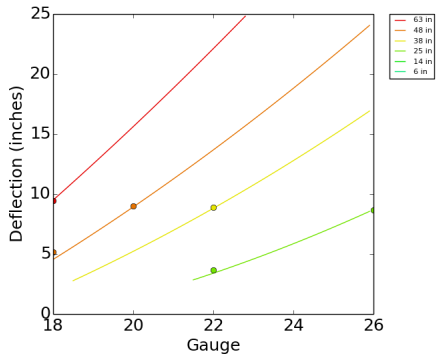
Figure B.3.: Unreinforced Pressure vs Deflection for a Constant Gauge



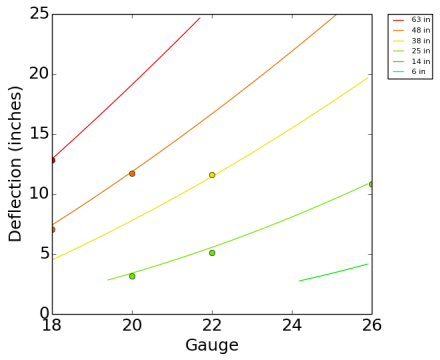
(a) 1 inWG



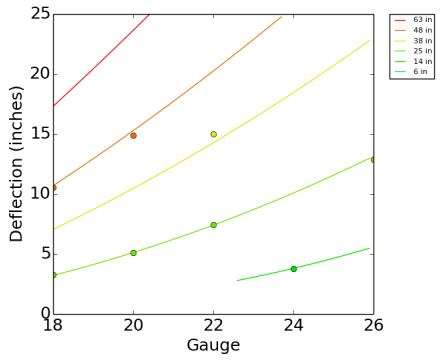
(b) 2 inWG



(c) 4 inWG

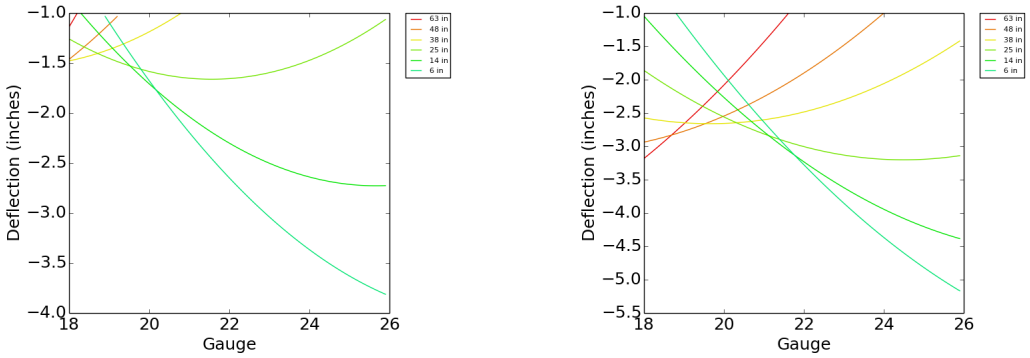


(d) 6 inWG



(e) 10 inWG

Figure B.4.: Unreinforced Gauge vs Deflection for a Constant Positive Pressure



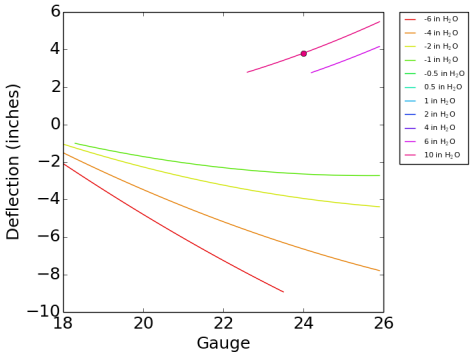
(a) -1 inWG

(b) -2 inWG

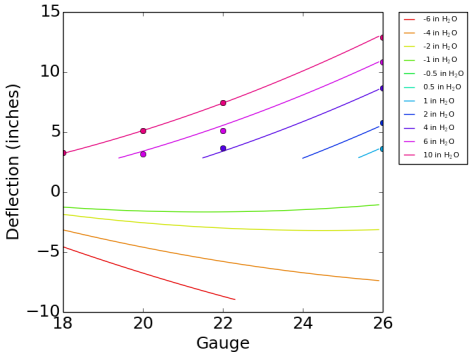
(c) -4 inWG

(d) -6 inWG

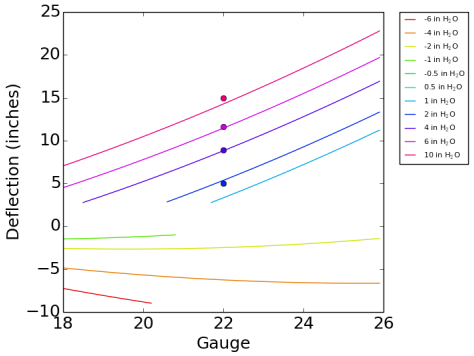
Figure B.5.: Unreinforced Gauge vs Deflection for a Constant Negative Pressure



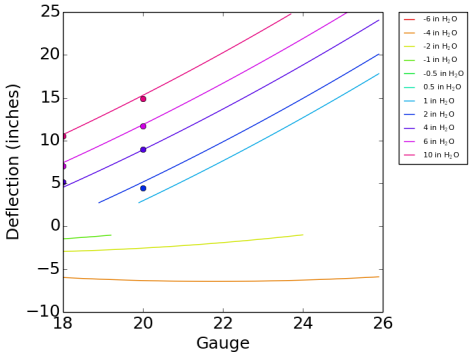
(a) 14 in



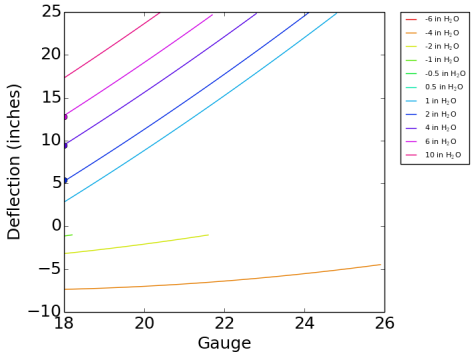
(b) 25 in



(c) 38 in

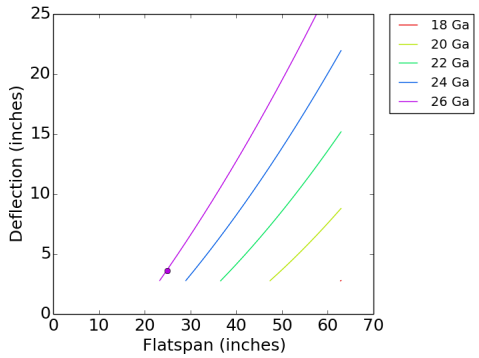


(d) 48 in

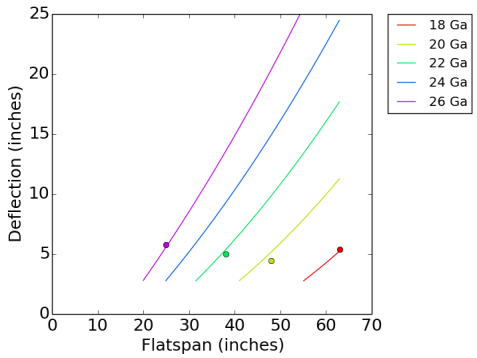


(e) 63 in

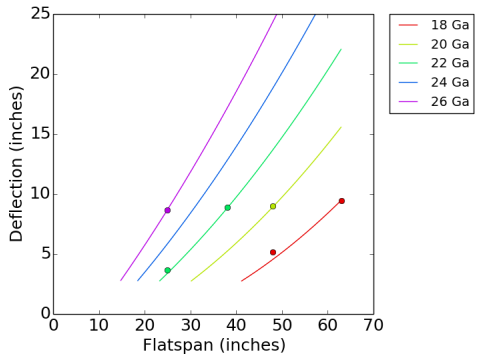
Figure B.6.: Unreinforced Gauge vs Deflection for a Constant Flat Span



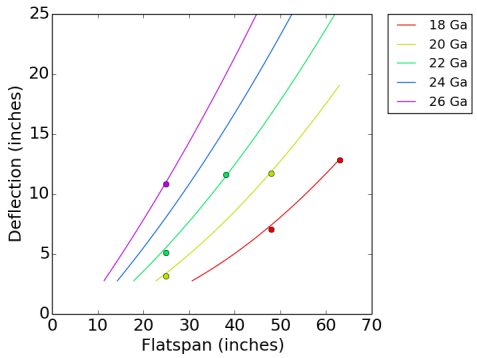
(a) 1 inWG



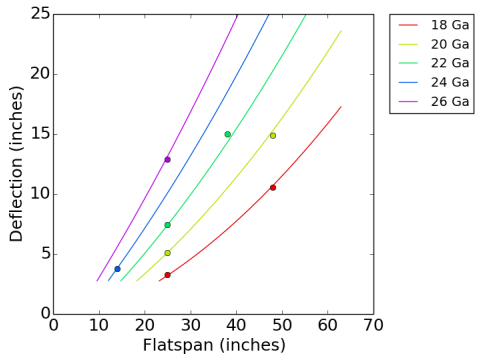
(b) 2 inWG



(c) 4 inWG

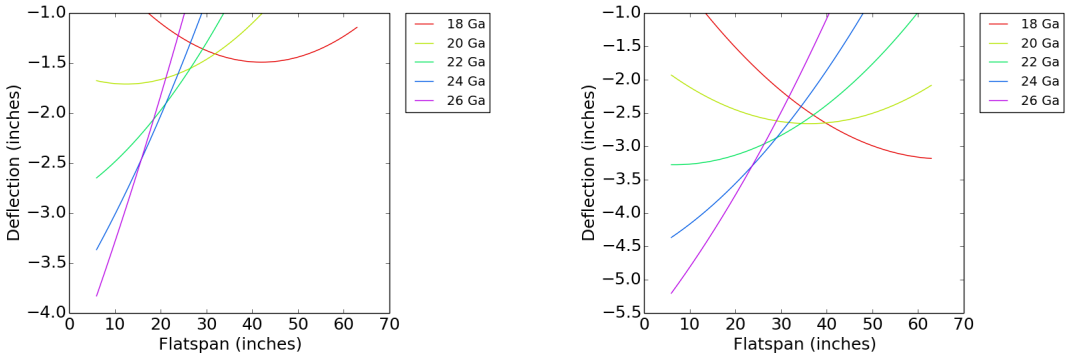


(d) 6 inWG



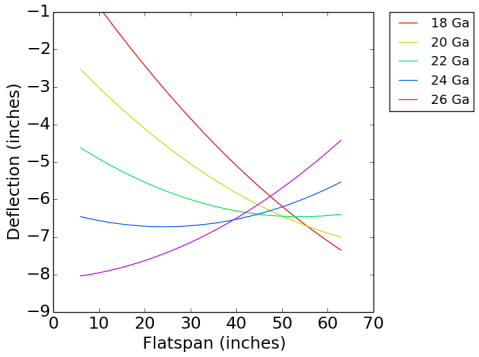
(e) 10 inWG

Figure B.7.: Unreinforced Flat Span vs Deflection for a Constant Positive Pressure

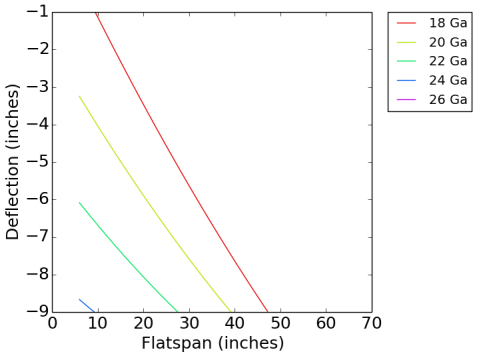


(a) -1 inWG

(b) -2 inWG



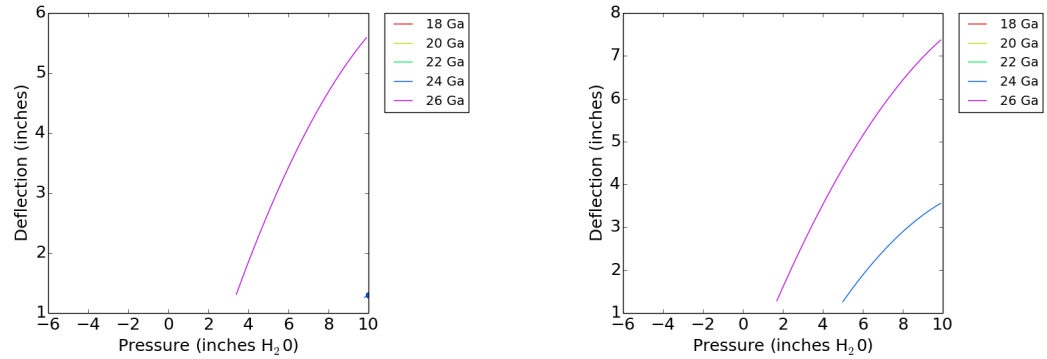
(c) -4 inWG



(d) -6 inWG

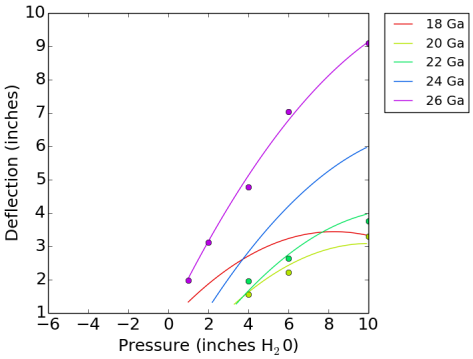
Figure B.8.: Unreinforced Flat Span vs Deflection for a Constant Negative Pressure

B.2.2 T-25 12 foot lengths

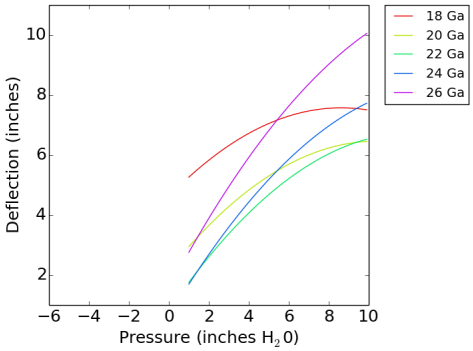


(a) 6 in

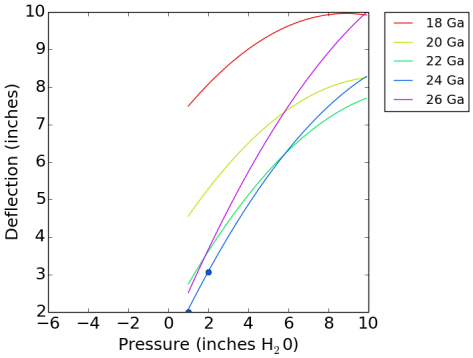
(b) 14 in



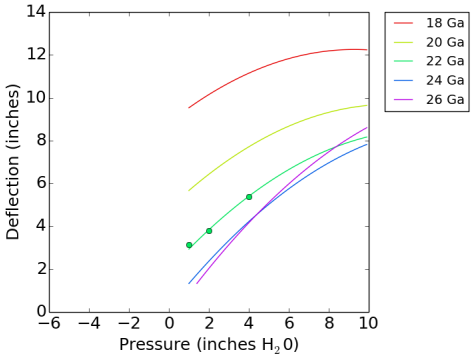
(c) 25 in



(d) 38 in

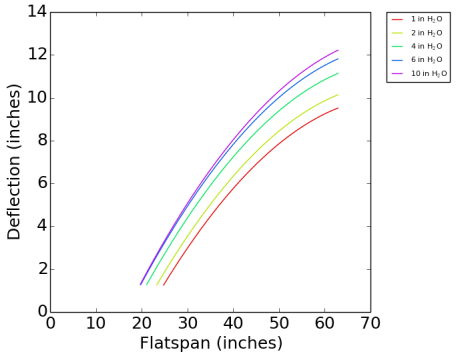


(e) 48 in

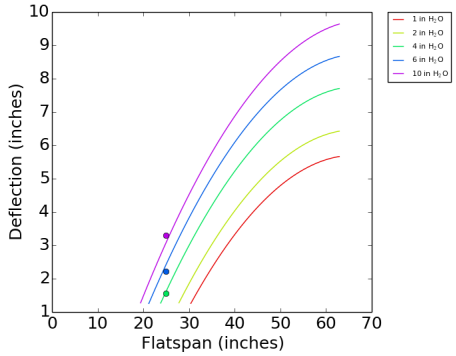


(f) 63 in

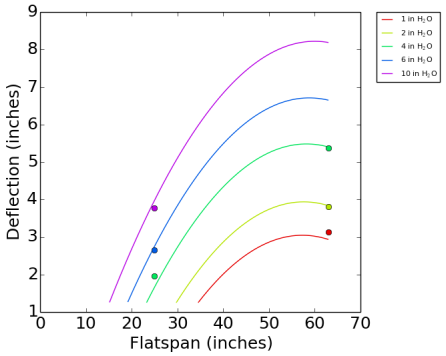
Figure B.9.: T-25 Pressure vs Deflection for a Constant Flat Span



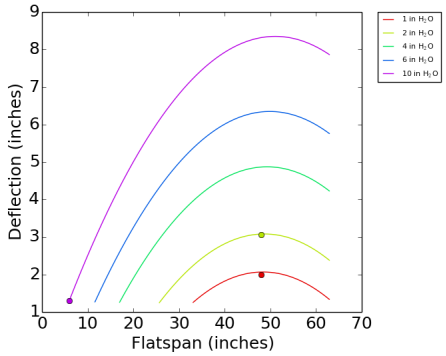
(a) 18 ga



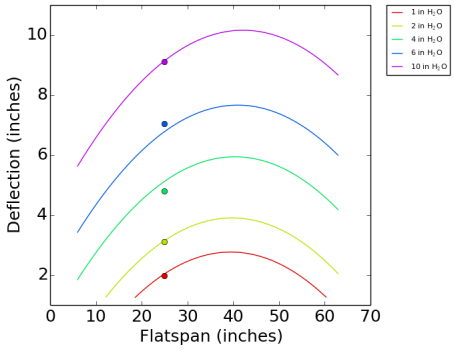
(b) 20 ga



(c) 22 ga

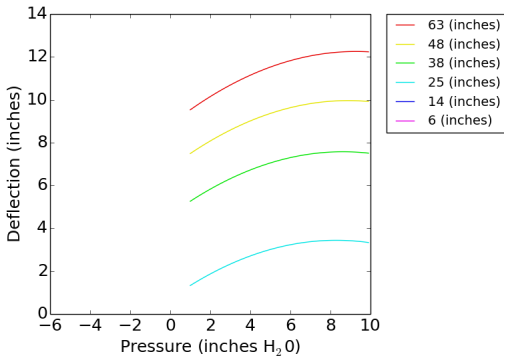


(d) 24 ga

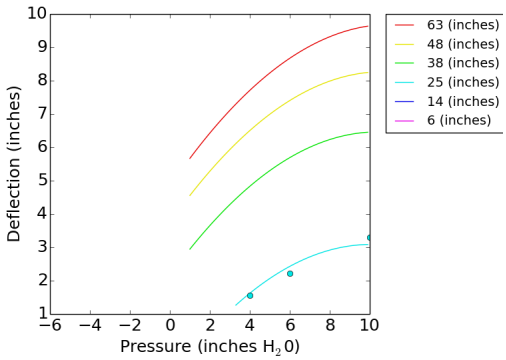


(e) 26 ga

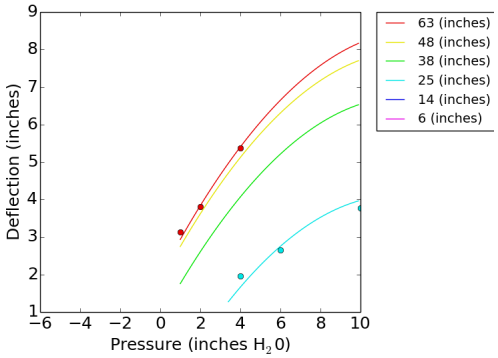
Figure B.10.: T-25 Flat Span vs Deflection for a Constant Gauge



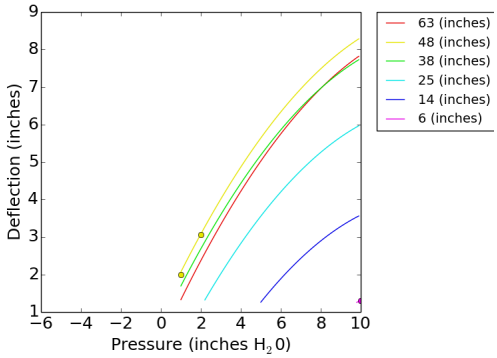
(a) 18 ga



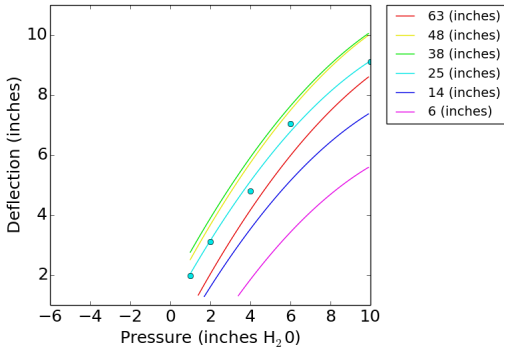
(b) 20 ga



(c) 22 ga

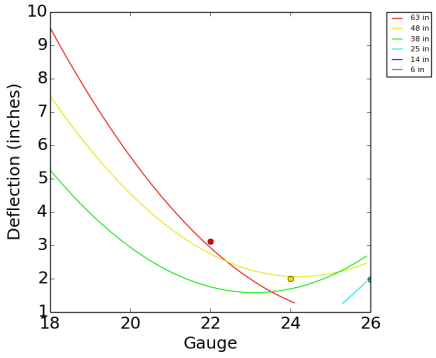


(d) 24 ga

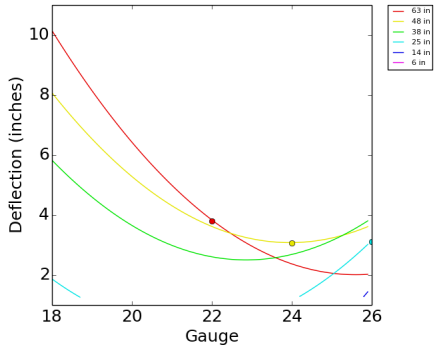


(e) 26 ga

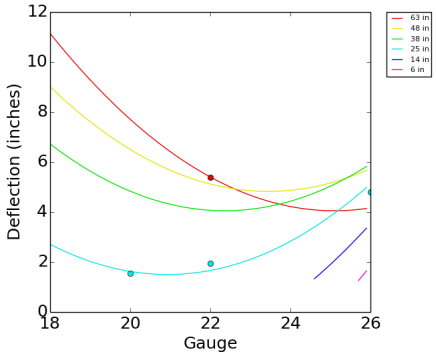
Figure B.11.: T-25 Pressure vs Deflection for a Constant Gauge



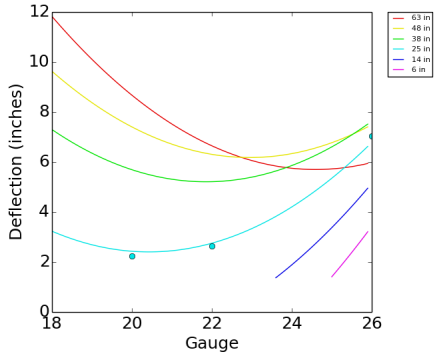
(a) 1 inWG



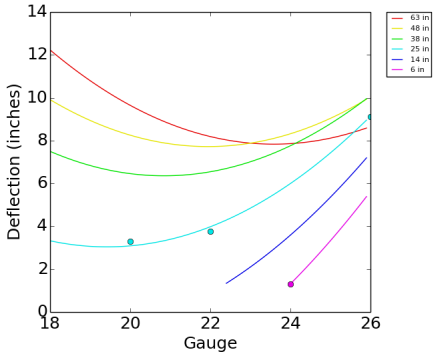
(b) 2 inWG



(c) 4 inWG

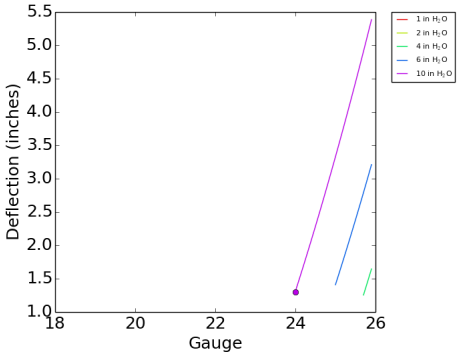


(d) 6 inWG

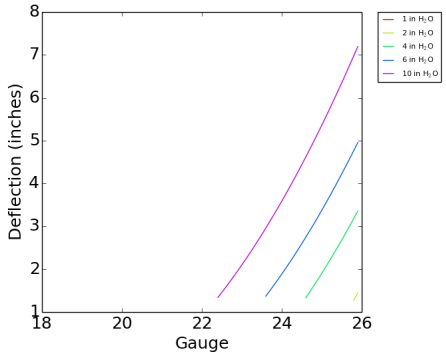


(e) 10 inWG

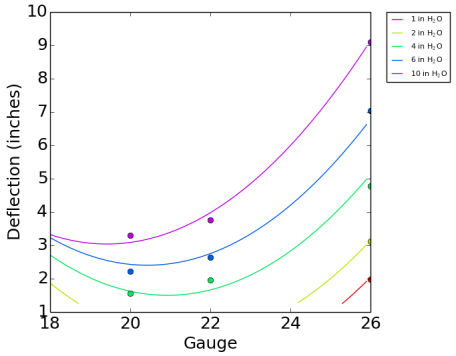
Figure B.12.: T-25 Gauge vs Deflection for a Constant Positive Pressure



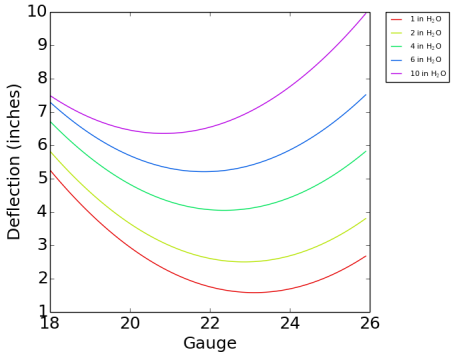
(a) 6 in



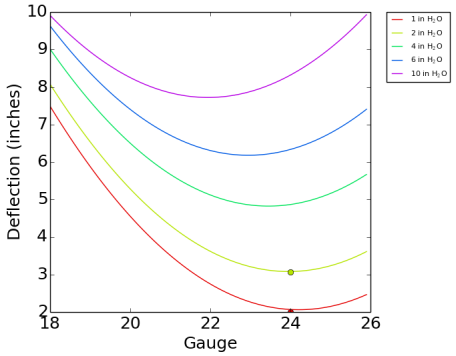
(b) 14 in



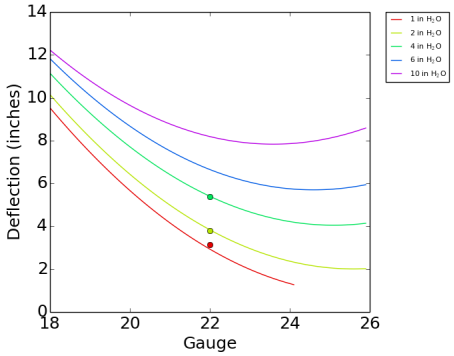
(c) 25 in



(d) 38 in

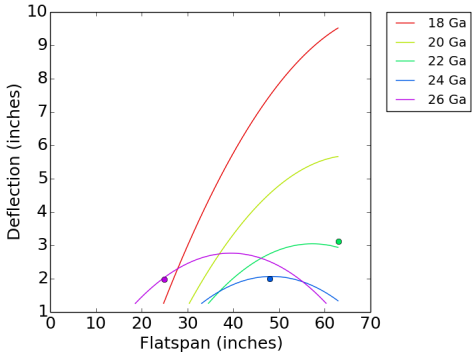


(e) 48 in

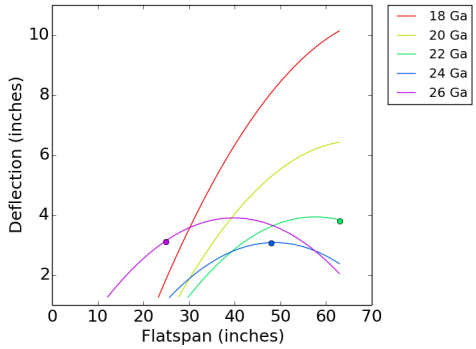


(f) 63 in

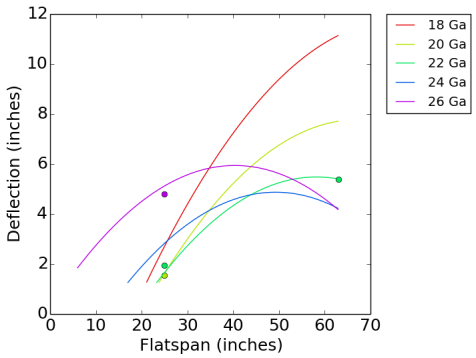
Figure B.13.: T-25 Gauge vs Deflection for a Constant Flat Span



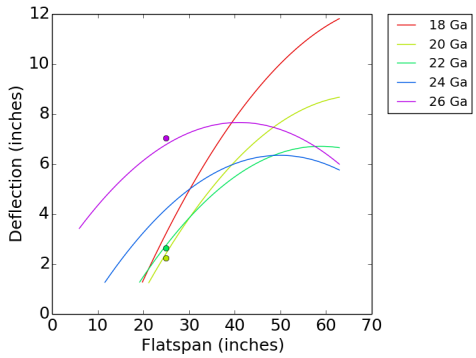
(a) 1 in



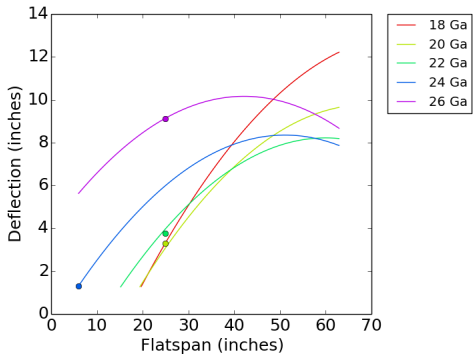
(b) 2 in



(c) 4 in



(d) 6 in



(e) 10 in

Figure B.14.: T-25 Flat Span vs Deflection for a Constant Positive Pressure

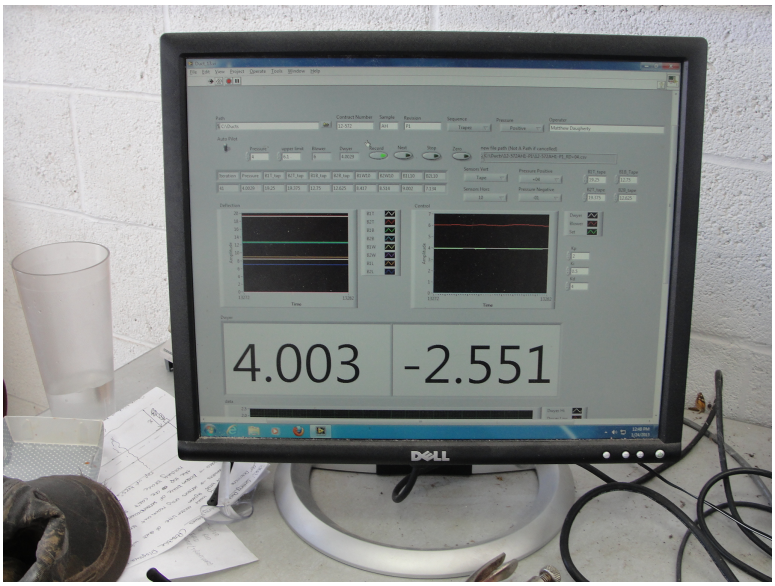


Figure B.15.: Technion Setup

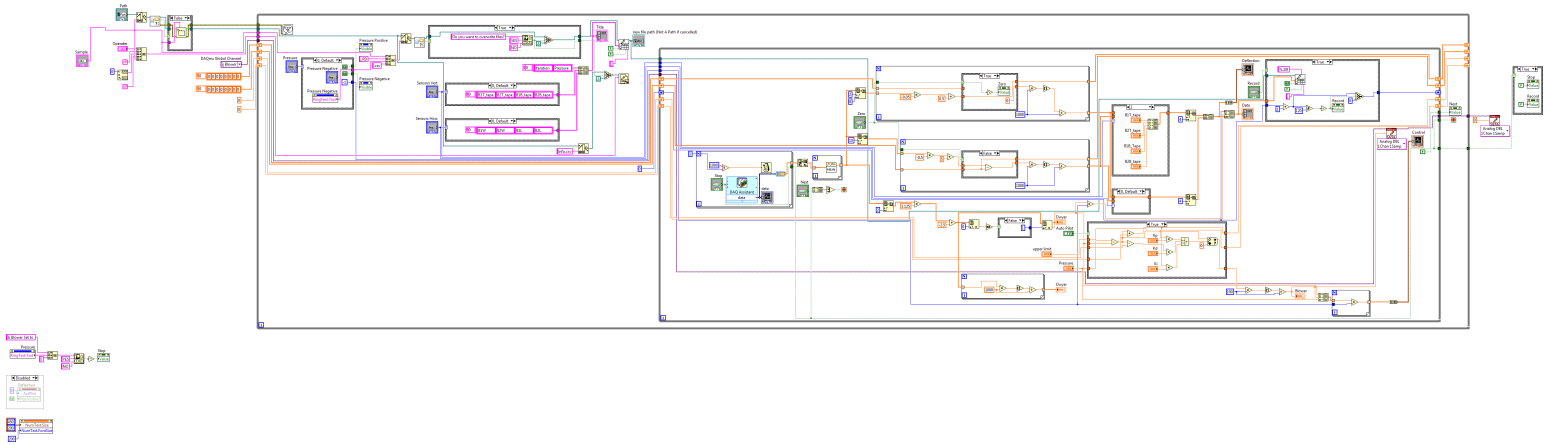


Figure B.16.: Program Block Diagram

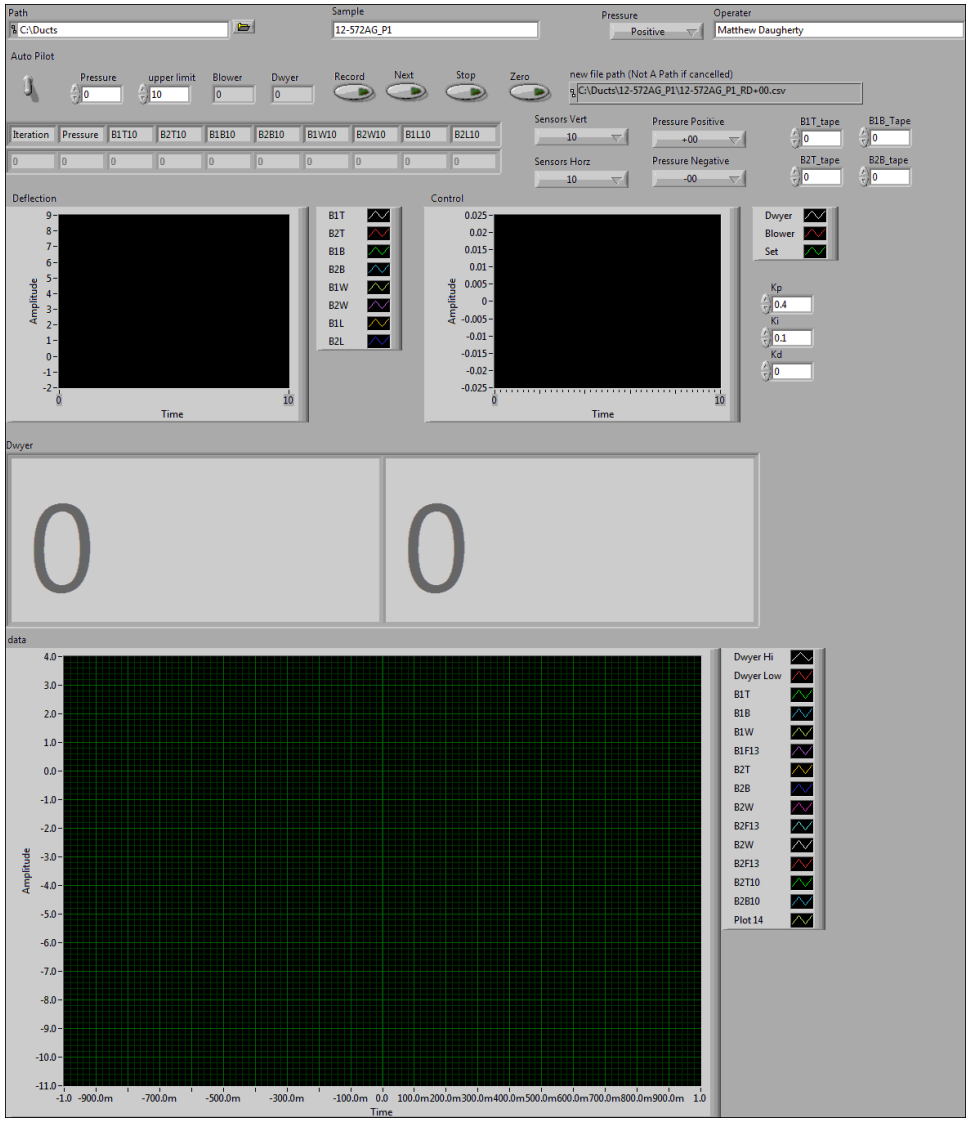


Figure B.17.: User Interface

```
1  #!/usr/bin/python
2  def weight(h,d):
3      from math import pi, sin, radians, ceil
4      density=500/12**3
5      RL=h/sin(radians(60))
6      Rod=RL*d**2*pi/4
7      Web_count=ceil(2*(24*12/RL))
8      print("Web_count="+str(Web_count))
9      print("Rod_Length="+str(ceil(Web_count*4*18.5/12))+"ft")
10     AIL=RL*Web_count/2
11     AIW=AIL*(1/8*2+1/8*2)
12     print("AIL="+str(ceil(AIL))+"␣in\t"+str(AIL/12)+"␣ft")
13     RW=Web_count*Rod
14     return density*4*(RW+AIW)
15
16 def length(h,d,L):
17     from math import pi, sin, radians, ceil
18     density=500/12**3
19     RL=h/sin(radians(60))
20     Rod=RL*d**2*pi/4
21     Web_count=ceil(2*(L*12/RL))
22     Rod_count=Web_count*4
23     print("Web_count="+str(Web_count))
24     print("Rod_count="+str(Rod_count))
25     print("Rod_Length="+str(ceil(Web_count*4*18.5/12))+"ft")
26     AIL=RL*Web_count/2
27     AIW=AIL*(1/8*2+1/8*2)
28     print("AIL="+str(ceil(AIL))+"␣in\t"+str(AIL/12)+"␣ft")
29     RW=Web_count*Rod
30     return density*4*(RW+AIW)
```

```
1  #!/usr/bin/python
2  #from math import pi, radian, sin
3  from duct import weight
4  d=8             #in
5
6  E=30*10**6   #psi
7  L=24*12         #in
8  #V=L*4*0.5 #in**3
9  TW=weight(2*d,3/8) #lb
10 W=(300+TW)/L        #lb/in
11 b1=2-1/8        #in
12 h1=1/8          #in
13 b2=1/8          #in
14 h2=2            #in
15 e=d-1/16        #in
16
17 I=b1*h1**3/12+b1*h1*d**2+b2*h2**3/12+b2*h2*e**2        #in**4
18
19 delta=5*W*L**4/(E*4*I*384)        #in
20
21 print("TW="+str(round(TW,2)))
22 print("I="+str(round(I,2)))
23 print("delta="+str(round(delta,2)))
```

```
1  #!/usr/bin/python2
2  print "Loading Libs"
3  import os
4  print 'PID:\t'+str(os.getpid())
5  os.nice(10)
6  import pickle
7  from argparse import ArgumentParser
8  import numpy
9  from glob import glob
10 from sys import stdout
11 import multiprocessing as mp
12 poolsize=mp.cpu_count()
13 from time import sleep
14 import matplotlib as mpl
15 mpl.use('Agg')
16 import matplotlib.pyplot as plt
17
18 print "Loading Custom Files"
19 from Duct import Duct
20 from table import *
21 from graph import *
22 from ratio import *
23 from height import *
24
25 def run_worker(obj):
26     obj.run()
27     return obj
28
29 FS_list=[63, 48, 38, 25, 14, 6]
30 g_list=[18, 20, 22, 24, 26]
```

```
1  P_list=[-6,-4,-2,-1,-0.5,0.5,1,2,4,6,10]
2  config=['Unreinforced', 'T-25 6ft lengths', 'T-25 12ft
       ↪ lengths', 'Attached reinforcing 3ft centers',
       ↪ 'Attached reinforcing 6ft centers', 'Trapeze 3ft
       ↪ centers', 'Trapeze 6ft centers', 'Tie Rod 3ft
       ↪ centers', 'Tie Rod 6ft centers']
3  PorN=['P', 'N']
4
5  def main():
6
7      p = ArgumentParser()
8      p.add_argument('-p', '--pickle', action='store_true',
           ↪ help='Reads a pre-stored file of ducts for faster
           ↪ run if available')
9      p.add_argument('-s', '--standard', action='store_true',
           ↪ help='Generates individual duct graphs')
10     p.add_argument('-m', '--matrix', action='store_true',
           ↪ help='Generates a matrix of tested and inventoried
           ↪ ducts')
11     p.add_argument('-g', '--graphs', action='store_true',
           ↪ help='Generates a graphs of poly and data')
12     p.add_argument('-S', '--splines', action='store_true',
           ↪ help='Adds splines to Graphs')
13     p.add_argument('-B', '--bounds', action='store_true',
           ↪ help='Adds splines to Bounds')
14     p.add_argument('-C', '--contour', action='store_true',
           ↪ help='Plots contour graphs graphs must be enabled')
15     p.add_argument('-t', '--tables', action='store_true',
           ↪ help='Generates a tables of poly')
16     p.add_argument('-r', '--ratio', action='store_true',
           ↪ help='Ratio stuff')
```

```
1   p.add_argument('-H', '--height', action='store_true',
        ↪ help='Height stuff')
2   p.add_argument('-o', '--objgraphs', action='store_true',
        ↪ help='Height stuff and Reinforcement lengths')
3   p.add_argument('-F', '--fast', action='store_true',
        ↪ help='Grainy contour graphs')
4   p.add_argument('-R', '--Raw', action='store_true',
        ↪ help='Raw data graphs')
5   p.add_argument('-c', '--coef', action='store_true',
        ↪ help='Model Coefficients')
6   args = p.parse_args()
7   del p
8
9   if args.pickle and os.path.isfile('/tmp/Duct.obj'):
10      print "Loading Pickle"
11      file = open("/tmp/Duct.obj",'rb')
12      Ducts = pickle.load(file)
13      file.close()
14      del file
15  else:
16      os.chdir('Ducts')
17      folders=glob.glob('12-572[A-Z][A-Z]_[PN][0-9]')
18      folders.remove('12-572AE_P1')
19      folders.remove('12-572BC_P1')
20      folders.remove('12-572AC_P1')
21
22      Ducts=[]
23      print "Generating Objects"
24      for duct in folders:
25          Ducts.append(Duct(duct))
```

```
1           pool=mp.Pool(poolsize)
2           Ducts=pool.map(run_worker, Ducts)
3           pool.close()
4           pool.terminate()
5           pool.join()
6           del pool, folders, duct
7
8           os.chdir('..')
9           print "Saving Pickle"
10          file = open("/tmp/Duct.obj","wb")
11          pickle.dump(Ducts,file)
12          file.close()
13          del file
14
15      tmp=[]
16      for i in Ducts:
17          if i.name != '12-572AI_P1' and i.name !=
                ↪ '12-572AD_P1' and i.name != '12-572AE_N1':
18              if i.name!='12-572AB_N1' and i.name !=
                    ↪ '12-572AC_N1':# and '12-572AB_P3' and
                    ↪ i.name != '12-572AC_P3':
19                  tmp.append(i)
20      Ducts=tmp
21      del tmp
22
23      print 'Ducts:\t'+str(len(Ducts))
24      threads=[]
25      tables=[]
26      for c in ['Unreinforced', 'T-25 12ft lengths']:
27          if c == 'Unreinforced':
28              tPorN=['P', 'N']
```

```
        else:
            tPorN=['P']
        for p in tPorN:
            try:
                tmp=table(Ducts,p,c)
                tables.append(tmp)
                del tmp
                print c+' '+p+' Passed'
            except AttributeError:
                print c+' '+p+' Failed AE'
            except UnboundLocalError:
                print c+' '+p+' Failed UL'
            except TypeError:
                print c+' '+p+' Failed TE'
    del c, p, tPorN

    if args.matrix:
        mat=matrix(Ducts,tables)
        threads.append(mp.Process(target=mat.run))
        del mat

    if args.ratio:
        for t in tables:
            if t.config is 'Unreinforced':
                for s in [None, 3, 6]:
                    ra=ratioall(t,Ducts,s)
                    threads.append(mp.Process(target=ra.run))
                for c in config:
                    if c is not 'Unreinforced':
                        try:
```

```
1                               r=ratio(t,Ducts,c)
2                               threads.append(mp.Process(target=r.run))
3                           except:
4                               print c+' Failed'
5           del r, ra
6
7       if args.height:
8           for p in PorN:
9               h=height(Ducts,p)
10              threads.append(mp.Process(target=h.run))
11
12
13      if args.graphs:
14          graphs=[]
15          agraphs=[]
16          for t in tables:
17              graphs.append(graph(Ducts,tables,t.config,t.PorN))
18          if args.fast:
19              for g in graphs:
20                  g.step=0.5
21          agraphs.append(allgraphs(tables,'Unreinforced'))
22          agraphs.append(allgraphs(tables,'T-25 12ft lengths'))
23          print 'Generating Graphs'
24          for g in graphs:
25              if args.contour:
26                  if g.PorN=='P':
27                      for p in [1,2,4,6,10]:
28                          threads.append(mp.Process(target=g.contourP,
                                ↪ args=(p,)))
29                      if g.config == 'Unreinforced':
```

```
                        for d in range(3,7):
                            threads.append(mp.Process(target=g.contourD,
                                ↪ args=(d ,)))
                    else:
                        for d in range(2,7):
                            threads.append(mp.Process(target=g.contourD,
                                ↪ args=(d ,)))
                else:
                    for p in [-6,-4,-3,-2,-1]:
                        threads.append(mp.Process(target=g.contourP,
                            ↪ args=(p ,)))
                    for d in range(1,7):
                        threads.append(mp.Process(target=g.contourD,
                            ↪ args=(-1*d ,)))
            threads.append(mp.Process(target=g.accuracy45))
            threads.append(mp.Process(target=g.accuracyD))
        for g in agraphs:
            g.S=args.splines
            g.B=args.bounds
            threads.append(mp.Process(target=g.P_constant))
            threads.append(mp.Process(target=g.Ga_constant))
            threads.append(mp.Process(target=g.Fs_constant))
            threads.append(mp.Process(target=g.GaFs_constant))
            threads.append(mp.Process(target=g.Ga_FsP))
            threads.append(mp.Process(target=g.Ga_PFs))
    if args.objgraphs:
        o=objgraph(Ducts)
        threads.append(mp.Process(target=o.run))
        del o
```

```
1     if args.tables:
2         ptables=[]
3         for t in tables:
4             ptables.append(ptable(Duct,tables,t.config,t.PorN))
5         ptables[0].contourP(1.0)
6         for pt in ptables:
7             if pt.PorN == 'P':
8                 for p in [1,2,4,6,10]:
9                     threads.append(mp.Process(target=pt.contourP,
                          ↪ args=(p ,)))
10                    if pt.config == 'Unreinforced':
11                        for d in range(3,7):
12                            threads.append(mp.Process(target=pt.contourD,
                                  ↪ args=(d ,)))
13                    else:
14                        for d in range(2,7):
15                            threads.append(mp.Process(target=pt.contourD,
                                  ↪ args=(d ,)))
16            else:
17                for p in [-6,-4,-3,-2,-1]:
18                    threads.append(mp.Process(target=pt.contourP,
                          ↪ args=(p ,)))
19                for d in range(1,7):
20                    threads.append(mp.Process(target=pt.contourD,
                          ↪ args=(-1*d ,)))
21        del ptables, pt, d
22
23    if args.coef:
24        coefs=[]
```

```
1           coefs.append(['␣','config+'␣
               ↪ '+tables[0].PorN,'config+'␣
               ↪ '+tables[1].PorN,'config+'␣
               ↪ '+tables[2].PorN])
2           dec=2
3           for i in range(len(tables[0].C)):
4               coefs.append(['C'+str(i),
                   ↪ round(tables[0].C[i],dec),
                   ↪ round(tables[1].C[i],dec),
                   ↪ round(tables[2].C[i],dec)])
5           filename='png/Coef.csv'
6           text=""
7           for row in coefs:
8               for cell in row:
9                   text+=str(cell)
10                  text+=','
11              text=text[:-1]
12              text+='\n'
13          f=open(filename, 'w')
14          f.write(text)
15          f.close()
16
17      if args.Raw:
18          for c in [config[0], config[1]]:
19              r=Raw(Ducts, c, 'P')
20              for p in [1,2,4,6,10]:
21                  threads.append(mp.Process(target=r.graph,
                       ↪ args=(p,)))
22              r=Raw(Ducts, c, 'N')
```

```
1                for p in [-6,-4,-2,-1]:
2                    threads.append(mp.Process(target=r.graph,
                         ↪ args=(p ,)))
3
4        if args.standard:
5            print 'Generating Standard Graphs'
6            for duct in Ducts:
7                threads.append(mp.Process(target=duct.graph))
8            del duct
9        del Ducts, args, tables
10       count=0
11       print 'Running '+str(len(threads))+' threads'
12       for thread in threads:
13           while len(mp.active_children())==poolsize:
14               sleep(0.25)
15           thread.start()
16           count+=1
17           stdout.write('\r'+str(count-len(mp.active_children()))
                 ↪ +'/'+str(len(threads)))
18       if 'thread' in locals():
19           del thread
20       while len(mp.active_children())!=0:#this is an easy way
             ↪ to join zombie threads.
21           stdout.write('\r'+str(count-len(mp.active_children()))
                 ↪ +'/'+str(len(threads)))
22           sleep(0.25)
23       stdout.write('\r'+str(count-len(mp.active_children()))
             ↪ +'/'+str(len(threads))+'\n')
24       del threads
25
26   if __name__=='__main__':
27       main()
```

```
#!/usr/bin/python2
import os
import numpy
import glob
from string import ascii_uppercase
from scipy.optimize import leastsq
from scipy import interpolate
from openpyxl.reader.excel import load_workbook
from warnings import simplefilter
simplefilter('ignore', UserWarning)
from textwrap import wrap
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt

diff=0.3

def spline(ix,iy):
    for i, j in enumerate(iy):
        if not numpy.isnan(j):
            if 'x' in locals():
                x=numpy.append(x,[[ix[i]],[iy[i]]],axis=1)
            else:
                x=numpy.array([[ix[i]],[iy[i]]])
    x=x[:,x[0].argsort(axis=0)]
    k=len(x[0])-1
    if k==0:
        return x[0],x[1]
    elif k>3:
        k=3
```

```
    tck=interpolate.splrep(x[0],x[1],k=k,s=0)
    xn=numpy.linspace(x[0][0],x[0][-1],200)
    yn=interpolate.splev(xn,tck,der=0)
    return xn, yn

def save_table(filename, table):
    filename=filename.replace(' ','_')+'.csv'
    text=""
    for row in table:
        for cell in row:
            if cell==0.0:
                text+=""
            else:
                text+=str(cell)
            text+=','
        text=text[:-1]
        text+='\n'
    f=open(filename, 'w')
    f.write(text)
    f.close()

class Duct(object):
    def __init__(self, name):
        self.name=name

    def run(self):
        data=[]
        self.head=[]
        self.list_of_files=[]
        self.Map={}
```

```
1
2           #Duct properties Section
3           #Height
4           if self.name[6:8] == 'AM':
5               self.height=30
6           elif self.name[6:8] == 'AL':
7               self.height=4
8           else:
9               self.height=16
10
11          #Width
12          if self.name[6:8] in ['AH', 'AR', 'AY']:
13              self.width=79
14          elif self.name[6:8] in ['AA', 'AC', 'AI', 'AS',
              ↪ 'AZ']:
15              self.width=64
16          elif self.name[6:8] == 'AM':
17              self.width=55
18          elif self.name[6:8] in ['AD', 'AJ', 'AT', 'BA']:
19              self.width=54
20          elif self.name[6:8] in ['AV', 'BB', 'AB', 'AK',
              ↪ 'AE', 'AN', 'AO', 'AU', 'BB']:
21              self.width=41
22          elif self.name[6:8] in ['AP', 'AW', 'BC']:
23              self.width=30
24          elif self.name[6:8] == 'AL':
25              self.width=29
26          elif self.name[6:8] in ['AG', 'AF', 'AQ', 'AX']:
27              self.width=22
28          else:
29              print 'Error: Width'
```

```
1               print self.name
2               self.width=0
3
4           if self.name[6:] in ['AF_P1', 'AF_N1']:
5               self.width=30
6
7           self.flatspan=int(int(self.width)-int(self.height))
8
9           #Gauge AF_P7 26 Gauge?
10          if self.name[6:8] in ['BB', 'AY', 'AZ', 'BA', 'BC']:
11              self.gauge=18
12          elif self.name [6:8] in ['AV', 'AN', 'AR', 'AS',
              ↪ 'AT', 'AU', 'AW', 'AX']:
13              self.gauge=20
14          elif self.name [6:8] in ['AK', 'AH', 'AI', 'AJ',
              ↪ 'AL', 'AM', 'AO', 'AP', 'AQ']:
15              self.gauge=22
16          elif self.name [6:8] in ['AC', 'AD', 'AE', 'AF',
              ↪ 'AG']:
17              self.gauge=24
18          elif self.name [6:8] in ['AB', 'AA']:
19              self.gauge=26
20          else:
21              print 'Error: Gauge'
22              print self.name
23              self.gauge=0
24
25          #Material
26          if self.name[6:8] == 'AN':
27              self.material='ALUM'
28          elif self.name[6:8] == 'AO':
```

```
1               self.material='S304'
2           else:
3               self.material='Galv.'
4
5           #Positive or Negative
6           self.PorN=self.name[-2]
7
8           #Reinforcement Configuration
9           config=['None', 'Unreinforced', 'T-25 6ft lengths',
                ↪ 'T-25 12ft lengths', 'Attached reinforcing 3ft
                ↪ centers', 'Attached reinforcing 6ft centers',
                ↪ 'Trapeze 3ft centers', 'Trapeze 6ft centers',
                ↪ 'Tie Rod 3ft centers', 'Tie Rod 6ft centers']
10          self.config=config[int(self.name[-1])]
11          del config
12
13          if int(self.name[-1]) in [4,6,8]:
14              self.spacing=3
15          elif int(self.name[-1]) in [2,5,7,9]:
16              self.spacing=6
17          elif int(self.name[-1]) == 3:
18              self.spacing=12
19          elif int(self.name[-1]) == 1:
20              self.spacing=None
21          else:
22              print 'Spacing error'
23
24          #Data Section
25          os.chdir(self.name)
```

```
1         for i in ['00', '00.5', '01', '02', '03', '04',
              ↪ '06', '10']:            #Getting files from
              ↪ directory
2           for j in glob.glob('*'+str(i)+'.csv'):
3               self.list_of_files.append(j)
4
5         self.head=open(self.list_of_files[0]).readline().rstrip().split(',')
              ↪         #Ripping the head off one of the files
6
7         for file in self.list_of_files:
8               tmp=numpy.genfromtxt(file, skiprows=1,
                    ↪ delimiter=',')      #Grabbing the data
                    ↪ from the files
                if len(tmp) <= 1:
9                   print '\nWarning: Empty file '+str(file)
10              else:
11                  data.append(tmp)
12        del tmp, file
13
14        os.chdir('..') #cd back so it can find the next one
15
16        #Matrix generation and data averaging
17        data_mean=[]
18        for file in data:
19            data_mean.append(numpy.around(file.mean(axis=0),decimals=2))
20
21        #Creating an associative array
22        for i in range(len(self.head)):
23            self.Map[str(self.head[i])]=i
24
25        self.matrix=numpy.zeros([len(data_mean),len(data_mean[0])])
26        #Zeroing the data
27
```

```
        for line in range(len(data_mean)):
            if round(abs(data_mean[line][1]),1)!=0.5:
                self.Map[str(int(round(data_mean[line][1])))]=line
            else:
                self.Map[str(round(data_mean[line][1],1))]=line
            for column in range(len(data_mean[0])):
                self.matrix[line][column]=data_mean[line][column]

        del data_mean
        self.matrix=self.matrix-self.matrix[0]

        #Reading Tape Data
        file=self.name+'/'+self.name+'_TD.xlsx'
        if os.path.exists(file):
            wb = load_workbook(filename = file)
            if self.config in ['Unreinforced', 'T-25 12ft
              ↪ lengths']:
                if self.PorN == 'P':
                    sheet='Positive'
                else:
                    sheet='Negative'
            else:
                if self.PorN == 'P':
                    sheet='Positive_Reinforeced'
                else:
                    sheet='Negative Reinforced'
            sheet_ranges = wb.get_sheet_by_name(name = sheet)
            self.tape_data=numpy.zeros([8,19])

            self.tapeMap={}
            self.T=[]
```

```
1               for i, column in enumerate(ascii_uppercase[1:9]):
2                   cell=str(column)+str(6)
3                   if self.PorN == 'N':
4                       self.tapeMap[str(-1*sheet_ranges.cell(cell).value)]=i
5                   else:
6                       self.tapeMap[str(sheet_ranges.cell(cell).value)]=i
7                   cell=str(column)+str(7)
8                   self.T.append(str(sheet_ranges.cell(cell).value))
9                   for j, row in enumerate(range(8,27)):
10                      cell=str(column)+str(row)
11                      try:
12                          self.tape_data[i][j]=
                                ↪ float(sheet_ranges.cell(cell).value)
13                      except TypeError:
14                          if
                                ↪ sheet_ranges.cell(cell).value==None:
15                              self.tape_data[i][j]=numpy.NaN
16                          else:
17                              print
                                    ↪ sheet_ranges.cell(cell).value
18                      except ValueError:
19                          pass
20
21              column='A'
22              for i, row in enumerate(range(8,27)):
23                  cell=str(column)+str(row)
24                  self.tapeMap[str(sheet_ranges.cell(cell).value)]=i
25              self.tape_data=numpy.around(self.tape_data[0]
                  ↪ -self.tape_data,decimals=2)
26
27              del sheet, sheet_ranges, wb
```

```
1            tape_file=True
2        else:
3            print 'Tape data for '+self.name+' does not
                 ↪ exist'
4            tape_file=False
5        del file
6
7        self.Norm=numpy.empty([2,len(self.matrix)])
8        self.Norm_Tape=numpy.empty([2,len(self.matrix)])
9        self.Norm_point=numpy.empty([5,len(self.matrix)])
10       pressures = []
11       if self.PorN == 'N':
12           for i in [0, -0.5, -1, -2, -3, -4, -6]:
13               if str(i) in self.Map:
14                   pressures.append(i)
15       else:
16           for i in [0, 0.5, 1, 2, 4, 6, 10]:
17               if str(i) in self.Map:
18                   pressures.append(i)
19
20       #Normalizing the data
21       #This is also making depictions, because there were
             ↪ sensor issues during testing, some died others
             ↪ were wired up backwards.
22       if tape_file:
23           for i, p in enumerate(pressures):
24               if 'B1T10' in self.Map and
                     ↪ self.T[self.tapeMap[str(p)]]=='S':
25                   B1T=self.matrix[self.Map[str(p)]]
                         ↪ [self.Map['B1T10']]
26                   B2T=self.matrix[self.Map[str(p)]]
                         ↪ [self.Map['B2T10']]
```

```
1                   B1B=self.matrix[self.Map[str(p)]]
                        ↪ [self.Map['B1B10']]
2                   B2B=self.matrix[self.Map[str(p)]]
                        ↪ [self.Map['B2B10']]
3               elif self.T[self.tapeMap[str(p)]]=='T':
4                   B1T=self.tape_data[self.tapeMap[str(p)]]
                        ↪ [self.tapeMap['B1T']]
5                   B2T=self.tape_data[self.tapeMap[str(p)]]
                        ↪ [self.tapeMap['B2T']]
6                   B1B=self.tape_data[self.tapeMap[str(p)]]
                        ↪ [self.tapeMap['B1B']]
7                   B2B=self.tape_data[self.tapeMap[str(p)]]
                        ↪ [self.tapeMap['B2B']]
8               else:
9                   if self.T[self.tapeMap[str(p)]]!='S':
10                      print '\nXLSX␣file␣screwed␣up,␣using␣
                            ↪ sensor␣data␣for␣'+self.name
11                  if 'B1T10' in self.Map:
12                      B1T=self.matrix[self.Map[str(p)]]
                            ↪ [self.Map['B1T10']]
13                      B2T=self.matrix[self.Map[str(p)]]
                            ↪ [self.Map['B2T10']]
14                      B1B=self.matrix[self.Map[str(p)]]
                            ↪ [self.Map['B1B10']]
15                      B2B=self.matrix[self.Map[str(p)]]
                            ↪ [self.Map['B2B10']]
16                  elif 'B1T_tape' in self.Map:
17                      B1T=self.matrix[self.Map[str(p)]]
                            ↪ [self.Map['B1T_tape']]
18                      B2T=self.matrix[self.Map[str(p)]]
                            ↪ [self.Map['B2T_tape']]
```

```
1                           B1B=self.matrix[self.Map[str(p)]]
                              ↪ [self.Map['B1B_tape']]
2                           B2B=self.matrix[self.Map[str(p)]]
                              ↪ [self.Map['B2B_tape']]
3                       else:
4                           print 'Error Using Sensors'
5                           B1B, B2B, B1T, B2T=0.0, 0.0, 0.0, 0.0
6                   self.Norm[0][i]=
                      ↪ round(self.matrix[i][self.Map['Pressure']],2)
7                   self.Norm[1][i]=round((B1T+B2T)/2,2)
8                   self.Norm_point[0][i]=
                      ↪ round(self.matrix[i][self.Map['Pressure']],2)
9                   self.Norm_point[1][i]=round(B1T,2)
10                  self.Norm_point[2][i]=round(B2T,2)
11                  self.Norm_point[3][i]=round(B1B,2)
12                  self.Norm_point[4][i]=round(B2B,2)
13                  del B1B, B2B, B1T, B2T
14          else:
15              for i, line in enumerate(self.matrix):
16                  if 'B1T10' in self.Map:
17                      B1T=line[self.Map['B1T10']]
18                      B2T=line[self.Map['B2T10']]
19                      B1B=line[self.Map['B1B10']]
20                      B2B=line[self.Map['B2B10']]
21                  elif 'B1T_tape' in self.Map:
22                      B1T=line[self.Map['B1T_tape']]
23                      B2T=line[self.Map['B2T_tape']]
24                      B1B=line[self.Map['B1B_tape']]
25                      B2B=line[self.Map['B2B_tape']]
26                  else:
27                      print 'Error Using Sensors'
```

```
                    B1B, B2B, B1T, B2T=0.0, 0.0, 0.0, 0.0
                self.Norm[0][i]=round(line[self.Map['Pressure']],2)
                self.Norm[1][i]=round((B1T+B2T)/2,2)
                self.Norm_point[0][i]=
                    ↪ round(self.matrix[i][self.Map['Pressure']],2)
                self.Norm_point[1][i]=round(B1T,2)
                self.Norm_point[2][i]=round(B2T,2)
                self.Norm_point[3][i]=round(B1B,2)
                self.Norm_point[4][i]=round(B2B,2)
                del B1B, B2B, B1T, B2T
        del pressures
        while True:
            if (self.Norm[0][-1]<0.1) and
                ↪ (self.Norm[0][-1]>-0.1):
                self.Norm=numpy.around(numpy.delete(self.Norm,
                    ↪ -1, 1), decimals=2)
                self.Norm_point=numpy.around(numpy.delete(self.Norm_point,
                    ↪ -1, 1), decimals=2)
            else:
                break
        if self.PorN == 'P':
            self.Norm=numpy.absolute(self.Norm)
            self.Norm_point=numpy.absolute(self.Norm_point)
        else:
            self.Norm=numpy.absolute(self.Norm)
            self.Norm_point=numpy.absolute(self.Norm_point)

    def find_deflectionP(self,P):
        if self.config == 'Unreinforced':
            PLlimit=2.75
            PHlimit=25
            NLlimit=9
```

```
            NHlimit=1
        else:
            PLlimit=1.25
            PHlimit=25
            NLlimit=9
            NHlimit=1
        if P<0.75 and self.PorN == 'P':
            return None, None
        for i in range(len(self.Norm[0])):
            if (P-0.25<=self.Norm[0][i]<=P+0.25):
                if self.PorN == 'P' and
                    ↪ PLlimit<=abs(self.Norm[1][i])<=PHlimit:
                    return self.Norm[0][i], self.Norm[1][i]
                elif self.PorN == 'N' and
                    ↪ NHlimit<=abs(self.Norm[1][i])<=NLlimit:
                    return self.Norm[0][i], self.Norm[1][i]
                else:
                    return None, None
        return None, None

    def find_deflectionL(self,P):
        P=abs(P)
        for i in range(len(self.Norm[0])):
            if (P-0.25<=abs(self.Norm[0][i])<=P+0.25):
                return self.Norm[1][i]

    def find_deflection(self,P):
        press, deflection =self.find_deflectionP(P)
        return deflection
```

```
1     def graph(self):
2         fig=plt.figure()
3         fig.add_axes([0.15, 0.15, 0.5, 0.75])
4         plt.rc('font', size=26.0)
5         #line='Ga: '+str(self.gauge)+' Fs:
              ↪ '+str(self.flatspan)
6         line='Normalized'
7         markersize=10
8         linewidth=1
9         sen=[None, 'B1T', 'B2T', 'B1B', 'B2B']
10        for i in range(1,len(self.Norm_point)):
11            if i==1:
12                co='r'
13                m='s'
14            elif i==2:
15                #markersize-=2
16                co='g'
17                m='d'
18            elif i==3:
19                #markersize-=2
20                co='k'
21                m='^'
22            elif i==4:
23                co='k'
24                m='^'
25            else:
26                print 'Duct.py marker died'
27            xn, yn =
                  ↪ spline(self.Norm_point[0],self.Norm_point[i])
28            plt.plot(self.Norm_point[0], self.Norm_point[i],
                  ↪ m, ms=markersize, color=co, label=sen[i])
29            plt.plot(xn,yn,'-',lw=linewidth,color=co)
```

```
1           #markersize-=2
2           xn,yn=spline(self.Norm[0],self.Norm[1])
3           plt.plot(self.Norm[0], self.Norm[1], 'o',
                ↪ ms=markersize, label=line, color='b')
4           plt.plot(xn,yn,'-',lw=linewidth,color='b')
5           plt.xlabel('Pressure (inWG)')
6           plt.ylabel('Deflection (inches)')
7           plt.xticks(numpy.arange(0,12,2))
8           plt.xlim(xmax=10)
9           plt.ylim(ymin=0)
10          plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                ↪ borderaxespad=0., fontsize=18)
11          #plt.title(self.name)
12          filename=self.name
13          dirs=self.config.replace(' ','_')+'/'+self.PorN
14          dirs='png/'+dirs+'/Duct'
15          if not os.path.exists(dirs):
16              os.makedirs(dirs)
17          filename=dirs+'/'+filename.replace(' ','_')
18          plt.savefig(filename+'.png')
19          plt.close()
20          save_table(filename,
                ↪ numpy.around(numpy.transpose(self.Norm),
                ↪ decimals=1))
21
22  class House(object):
23      def __init__(self, cname):
24          self.num=cname[0]
25          self.name=cname[2:-1]
26          #Height
```

```
1           if self.name[6:8] == 'AM':
2               self.height=30
3           elif self.name[6:8] == 'AL':
4               self.height=4
5           else:
6               self.height=16
7
8           #Width
9           if self.name[6:8] in ['AH', 'AR', 'AY']:
10              self.width=79
11          elif self.name[6:8] in ['AA', 'AC', 'AI', 'AS',
              ↪ 'AZ']:
12              self.width=64
13          elif self.name[6:8] == 'AM':
14              self.width=55
15          elif self.name[6:8] in ['AD', 'AJ', 'AT', 'BA']:
16              self.width=54
17          elif self.name[6:8] in ['AV', 'BB', 'AB', 'AK',
              ↪ 'AE', 'AN', 'AO', 'AU', 'BB']:
18              self.width=41
19          elif self.name[6:8] in ['AP', 'AW', 'BC']:
20              self.width=30
21          elif self.name[6:8] == 'AL':
22              self.width=29
23          elif self.name[6:8] in ['AG', 'AF', 'AQ', 'AX']:
24              self.width=22
25          else:
26              print 'Error: Width'
27              print self.name
28              self.width=0
```

```
1           if self.name[6:] in ['AF_P1', 'AF_N1']:
2               self.width=30
3
4           self.flatspan=int(int(self.width)-int(self.height))
5
6           #Gauge AF_P7 26 Gauge?
7           if self.name[6:8] in ['BB', 'AY', 'AZ', 'BA', 'BC']:
8               self.gauge=18
9           elif self.name [6:8] in ['AV', 'AN', 'AR', 'AS',
              ↪ 'AT', 'AU', 'AW', 'AX']:
10              self.gauge=20
11          elif self.name [6:8] in ['AK', 'AH', 'AI', 'AJ',
              ↪ 'AL', 'AM', 'AO', 'AP', 'AQ']:
12              self.gauge=22
13          elif self.name [6:8] in ['AC', 'AD', 'AE', 'AF',
              ↪ 'AG']:
14              self.gauge=24
15          elif self.name [6:8] in ['AB', 'AA']:
16              self.gauge=26
17          else:
18              print 'Error: Gauge'
19              print self.name
20              self.gauge=0
21
22          #Material
23          if self.name[6:8] == 'AN':
24              self.material='ALUM'
25          elif self.name[6:8] == 'AO':
26              self.material='S304'
27          else:
```

```
1               self.material='Galv.'
2
3           #Positive or Negative
4           self.PorN=self.name[-2]
5
6           #Reinforcement Configuration
7           config=['None', 'Unreinforced', 'T-25 6ft lengths',
                ↪ 'T-25 12ft lengths', 'Attached reinforcing 3ft
                ↪ centers', 'Attached reinforcing 6ft centers',
                ↪ 'Trapeze 3ft centers', 'Trapeze 6ft centers',
                ↪ 'Tie Rod 3ft centers', 'Tie Rod 6ft centers']
8           self.config=config[int(self.name[-1])]
```

```
1  #!/usr/bin/python2
2  import numpy
3  from scipy import interpolate
4  import os
5  from colorsys import hsv_to_rgb
6  from textwrap import wrap
7  from table import *
8  from scipy import optimize
9  import matplotlib as mpl
10 mpl.use('Agg')
11 import matplotlib.pyplot as plt
12 FS_list=[63, 48, 38, 25, 14, 6]
13 g_list=[18, 20, 22, 24, 26]
14 P_list=[-6,-4,-2,-1,-0.5,0.5,1,2,4,6,10]
15 config=['Unreinforced', 'T-25 6ft lengths', 'T-25 12ft
       ↪ lengths', 'Attached reinforcing 3ft centers',
       ↪ 'Attached reinforcing 6ft centers', 'Trapeze 3ft
       ↪ centers', 'Trapeze 6ft centers', 'Tie Rod 3ft
       ↪ centers', 'Tie Rod 6ft centers']
16
17 def get_color(color):
18     for hue in range(color):
19         hue = 1. * hue / color
20         col = [int(x) for x in hsv_to_rgb(hue, 1.0, 230)]
21         yield "#{0:02x}{1:02x}{2:02x}".format(*col)
22
23 def spline(ix,iy):
24     for i, j in enumerate(iy):
25         if not numpy.isnan(j):
26             if 'x' in locals():
27                 x=numpy.append(x,[[ix[i]],[iy[i]]],axis=1)
```

```
            else:
                x=numpy.array([[ix[i]],[iy[i]]])
    x=x[:,x[0].argsort(axis=0)]
    k=len(x[0])-1
    if k==0:
        return x[0],x[1]
    elif k>3:
        k=3
    tck=interpolate.splrep(x[0],x[1],k=k,s=0)
    xn=numpy.linspace(x[0][0],x[0][-1],200)
    yn=interpolate.splev(xn,tck,der=0)
    return xn, yn

class graph(object):
    def __init__(self, objects, tables, config, PorN):
        self.objects=objects
        for t in tables:
            if t.config == config and t.PorN == PorN:
                self.table=t
        self.config=config
        self.PorN=PorN
        self.step=0.01

    def run(self):
        self.accuracy()
        if self.PorN=='P':
            self.contourP(1)
            self.countourD(1)
        else:
            self.contourP(-1)
            self.countourD(-1)
```

```
1
2       def contourP(self, P):
3           fig=plt.figure()
4           fig.add_axes([0.15, 0.15, 0.75, 0.75])
5           plt.rc('font', size=24.0)
6           FS_stop=70
7           Ga_stop=26
8           if self.PorN=='P':
9               if self.config=='Unreinforced':
10                  levels=[3,4,5,6,7,8,9,10]
11              else:
12                  if P==1:
13                      FS_stop=45
14                      Ga_stop=26
15                      levels=[2]
16                  elif P==2:
17                      FS_stop=50
18                      Ga_stop=26
19                      levels=[2,3]
20                  elif P==4:
21                      FS_stop=40
22                      Ga_stop=26
23                      levels=[2,3,4]
24                  elif P==6:
25                      FS_stop=50
26                      Ga_stop=26
27                      levels=[2,3,4,5,6]
28                  elif P==10:
29                      FS_stop=55
30                      Ga_stop=26
31                      levels=[2,3,4,5,6,7,8]
```

```
                else:
                    FS_stop=22
                    Ga_stop=26
                    levels=[2,3,4,5,6,7,8,9,10]
        else:
            #levels=[0,-1,-2,-3,-4,-5,-6]
            if P==-1:
                FS_stop=22
                Ga_stop=26
                levels=[-1]
            elif P==-2:
                FS_stop=30
                Ga_stop=26
                levels=[-1,-2]
            elif P==-3:
                FS_stop=45
                Ga_stop=23
                levels=[-1,-2,-3,-4,-5,-6,-7,-8,-9,-10]
            elif P==-4:
                FS_stop=50
                Ga_stop=24
                levels=[-3,-4,-5,-6]
            else:
                levels=[-1,-2,-3,-4,-5,-6,-7,-8,-9,-10]
        g=numpy.arange(18,26,self.step)
        FS=numpy.arange(0,70,self.step)
        D=numpy.empty([len(FS),len(g)])
        for x, ig in enumerate(g):
            for y, iFS in enumerate(FS):
                if ig <= Ga_stop and iFS <= FS_stop:
                    D[y][x]=self.table.D(ig,iFS,P)
```

```
                else:
                    D[y][x]=None
        CS=plt.contour(g, FS, D, levels)
        plt.clabel(CS, inline=1, fontsize=18, fmt='%1i inWG')
        plt.xticks(numpy.arange(18,28,2))
        plt.xlabel('Gauge')
        plt.ylabel('Flatspan (inches)')
        plt.axis([18,26,0,70])
        #plt.title('Contour deflection at '+str(P)+'in
            ↪ H$_2$O')
        dirs=self.config+'/'+self.PorN
        dirs=dirs.replace(' ','_')
        filename='P_Contour'+'_'+str(P)+'.png'
        filename='png/'+dirs+'/'+filename.replace(' ','_')
        plt.savefig(filename)
        plt.close()

    def contourD(self, d):
        fig=plt.figure()
        fig.add_axes([0.15, 0.15, 0.75, 0.75])
        plt.rc('font', size=24.0)
        g=numpy.arange(18,26,self.step)
        FS=numpy.arange(0,70,self.step)
        P=numpy.empty([len(FS),len(g)])
        Ga_stop=26
        FS_stop=70
        Ga_start=0
        FS_start=0
        if self.PorN=='P':
            if self.config=='Unreinforced':
                levels=[1,2,3,4,5,6,7]
```

```
1                else:
2                    if d==2:
3                        Ga_start=0
4                        FS_start=0
5                        Ga_stop=26
6                        FS_stop=50
7                        levels=[2,3,4,5,6,7]
8                    elif d==3:
9                        Ga_start=0
10                       FS_start=0
11                       Ga_stop=26
12                       FS_stop=50
13                       levels=[2,3,4,5,6,7]
14                   elif d==4:
15                       Ga_start=0
16                       FS_start=0
17                       Ga_stop=26
18                       FS_stop=50
19                       levels=[3,4,5,6,7]
20                   elif d==5:
21                       Ga_start=0
22                       FS_start=0
23                       Ga_stop=26
24                       FS_stop=45
25                       levels=[5,6,7]
26                   elif d==6:
27                       Ga_start=0
28                       FS_start=0
29                       Ga_stop=26
30                       FS_stop=50
```

```
                levels=[6,7]
            else:
                levels=[1,2,3,4,5,6,7]
        else:
            if d ==-1:
                Ga_start=0
                FS_start=0
                Ga_stop=22
                FS_stop=30
                levels=[-2,-3,-4,-5,-6]
            elif d==-3 or d==-2:
                Ga_start=20
                FS_start=10
                Ga_stop=22
                FS_stop=40
                levels=[-3,-4,-5,-6]
            elif d==-4:
                Ga_start=20
                FS_start=10
                Ga_stop=23
                FS_stop=45
                levels=[-1,-3,-4,-5,-6]
            elif d==-5:
                Ga_start=20
                FS_start=10
                Ga_stop=24
                FS_stop=40
                levels=[-1,-4,-5,-6]
            elif d==-6:
                Ga_start=20
```

```
1                   FS_start=10
2                   Ga_stop=24
3                   FS_stop=50
4                   levels=[-1,-2,-4,-5,-6]
5               else:
6                   levels=[-1,-2,-3,-4,-5,-6]
7           for x, ig in enumerate(g):
8               FS_lim=FS_start*(ig-Ga_start)/(18-Ga_start)
9               for y, iFS in enumerate(FS):
10                  if ig >= Ga_stop or iFS >= FS_stop or (ig <=
                        ↪ Ga_start and iFS <= FS_lim):
11                      P[y][x]=None
12                  else:
13                      P[y][x]=self.table.P(ig,iFS,d)
14          CS=plt.contour(g, FS, P, levels)
15          plt.clabel(CS, inline=1, fontsize=18, fmt='%1i in')
16          plt.xticks(numpy.arange(18,28,2))
17          plt.xlabel('Gauge')
18          plt.ylabel('Flatspan (inches)')
19          #plt.title('Contour pressure at '+str(d)+'in
                ↪ deflection')
20          dirs=self.config+'/'+self.PorN
21          dirs=dirs.replace(' ','_')
22          filename='D_Contour'+'_'+str(d)+'.png'
23          filename='png/'+dirs+'/'+filename.replace(' ','_')
24          plt.savefig(filename)
25          plt.close()
26
27      def accuracy45(self):
28          fig=plt.figure()
29          fig.add_axes([0.15, 0.15, 0.5, 0.75])
30          plt.rc('font', size=24)
```

```
1           for g in g_list:
2               for FS in FS_list:
3                   for duct in self.objects:
4                       if duct.gauge==g and duct.flatspan == FS
                            ↪ and duct.config == self.config and
                            ↪ duct.PorN==self.PorN and
                            ↪ duct.height==16:
5                           for P in P_list:
6                               d=duct.find_deflection(P)
7                               p=self.table.D(g,FS,P)
8                               if (d is not None) and (p is not
                                    ↪ None):
9                                   d=abs(d)
10                                  p=abs(p)
11                                  if 'x' in locals():
12                                      x=numpy.append(x,d)
13                                      y=numpy.append(y,p)
14                                  else:
15                                      x=numpy.array([d])
16                                      y=numpy.array([p])
17                          if 'x' in locals():
18                              line='Ga:␣'+str(duct.gauge)+'␣
                                    ↪ Fs:␣'+str(duct.flatspan)
19                              plt.plot(x, y,
                                    ↪ marker=self.m(duct.gauge),
                                    ↪ color=self.co(duct.flatspan),
                                    ↪ lw=0, label=line, ms=8)
20                              del x, y
21
22          try:
23              x_reg=numpy.array([0,25])
```

```
1               y_reg=numpy.array([0,25])
2               #for i in [1, 0.9, 1.1, 0.7, 1.3]:
3               plt.plot(1*x_reg, y_reg, color='k', lw=0.2,
                    ↪ label='0%')
4               plt.plot(0.9*x_reg, y_reg, '—', color='k',
                    ↪ lw=0.2, label='10%')
5               plt.plot(1.1*x_reg, y_reg, '—', color='k',
                    ↪ lw=0.2)
6               plt.plot(0.7*x_reg, y_reg, ':', color='k',
                    ↪ lw=0.2, label='30%')
7               plt.plot(1.3*x_reg, y_reg, ':', color='k',
                    ↪ lw=0.2)
8
9               plt.xlabel('D$_{mean}$')
10              plt.ylabel('D$_{pred}$')
11              #plt.title('M accuracy')
12              plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                    ↪ borderaxespad=0., fontsize=18)
13              plt.axis([0,20,0,20])
14              self.savegraph('M')
15              plt.axis([0,4,0,4])
16              self.savegraph('M4')
17          except TypeError:
18              print '\nx=None\t'+self.config+'\t'+self.PorN
19          plt.close()
20
21      def accuracyD(self):
22          fig=plt.figure()
23          fig.add_axes([0.15, 0.15, 0.5, 0.75])
24          plt.rc('font', size=24)
25          for g in g_list:
26              for FS in FS_list:
27                  for duct in self.objects:
```

```
1                       if duct.gauge==g and duct.flatspan == FS
                           ↪ and duct.config == self.config and
                           ↪ duct.PorN==self.PorN and
                           ↪ duct.height==16:
2                           for P in P_list:
3                               d=duct.find_deflection(P)
4                               Mod=self.table.D(g,FS,P)
5                               if (d is not None) and (Mod is
                                   ↪ not None):
6                                   p=abs(d-Mod)*100/d
7                                   if 'x' in locals():
8                                       x=numpy.append(x,d)
9                                       y=numpy.append(y,p)
10                                  else:
11                                      x=numpy.array([d])
12                                      y=numpy.array([p])
13                              if 'x' in locals():
14                                  line='Ga: '+str(duct.gauge)+'
                                       ↪ Fs: '+str(duct.flatspan)
15                                  plt.plot(x, y,
                                       ↪ marker=self.m(duct.gauge),
                                       ↪ color=self.co(duct.flatspan),
                                       ↪ lw=0, label=line, ms=8)
16                                  del x, y
17
18          try:
19              plt.xlabel('D$_{mean}$')
20              plt.ylabel('Percent Differance')
21              #plt.title('M diff')
```

```
1               plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                    ↪ borderaxespad=0., fontsize=18)
2               self.savegraph('DM')
3           except TypeError:
4               print '\nx=None\t'+self.config+'\t'+self.PorN
5           plt.close()
6
7       def savegraph(self,filename):
8           dirs='png/'+self.config+'/'+self.PorN
9           dirs=dirs.replace(' ','_')
10          if not os.path.exists(dirs):
11              os.makedirs(dirs)
12          dirs=dirs+'/'
13          filename=dirs+filename.replace(' ','_')+'.png'
14          plt.savefig(filename)
15
16      def m(self,Ga):
17          if Ga==18:
18              return 'x'
19          elif Ga==20:
20              return '+'
21          elif Ga==22:
22              return 'o'
23          elif Ga==24:
24              return '^'
25          elif Ga==26:
26              return 's'
27          else:
28              print 'self.m('+str(Ga)+') broke'
```

```
    def co(self,Fs):
        if Fs==6:
            return 'b'
        elif Fs==14:
            return 'g'
        elif Fs==25:
            return 'r'
        elif Fs==38:
            return 'm'
        elif Fs==48:
            return 'y'
        elif Fs==63:
            return 'k'
        else:
            print 'self.co('+str(Fs)+') broke'

class allgraphs(object):
    def __init__(self, tables, config):
        self.config=config
        self.P_list=[1,2,4,6,10]
        for i in tables:
            if i.config==config:
                if i.PorN == 'P':
                    self.ptable=i
                else:
                    self.ntable=i
                    self.P_list=[-6,-4,-2,-1,-0.5,0.5,1,2,4,6,10]
        self.B=False
        self.S=False
```

```
    def run(self):
        self.P_constant()
        self.Ga_constant()
        self.Fs_constant()
        self.GaFs_constant()
        self.Ga_FsP()
        self.Ga_PFs()

    def eq(self,Ga,Fs,P):
        if P>=0:
            return self.ptable.D(Ga,Fs,P)
        else:
            return self.ntable.D(Ga,Fs,P)

    def deflection(self,Ga,Fs,P):
        if P>0:
            return self.ptable.deflection(Ga,Fs,P)
        else:
            return self.ntable.deflection(Ga,Fs,P)

    def P_constant(self):
        for pressure in self.P_list:
            color=get_color(len(g_list))
            fig=plt.figure()
            fig.add_axes([0.15, 0.15, 0.6, .75])
            plt.rc('font', size=18)
            for g in g_list:
                reg_x=numpy.arange(min(FS_list),max(FS_list),0.1)
                reg_y=numpy.empty(len(reg_x))
                for i in range(len(reg_x)):
```

```
1                       reg_y[i]=self.eq(g,reg_x[i],pressure)
2                   data_x=numpy.empty(len(FS_list))
3                   data_y=numpy.empty(len(FS_list))
4                   for i in range(len(FS_list)):
5                       data_x[i]=FS_list[i]
6                       data_y[i]=self.deflection(g,FS_list[i],pressure)
7                   acolor=next(color)
8                   plt.plot(reg_x,reg_y,label=str(g)+' Ga',color=acolor)
9                   if self.B:
10                      plt.plot(reg_x, 0.9*reg_y, '--', reg_x, 1.1*reg_y, '--', color=acolor)
11                  if not numpy.isnan(data_y).all():
12                      plt.plot(data_x,data_y,'o',color=acolor)
13                      if self.S:
14                          xn,yn=spline(data_x,data_y)
15                          plt.plot(xn,yn,'-',color=acolor)
16              plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=14)
17              plt.xticks(numpy.arange(0,80,10))
18              plt.xlabel('Flatspan (inches)')
19              plt.ylabel('Deflection (inches)')
20              title=self.config+' duct deflection with respect to flatspan at '+str(pressure)+' (inches H$_2$O)'
21              #plt.title("\n".join(wrap(title,60)))
22              self.savegraph('P_'+str(pressure))
23              plt.close()
24
25      def Ga_constant(self):
26          for g in g_list:
27              color=get_color(len(self.P_list))
```

```
1               fig=plt.figure()
2               fig.add_axes([0.15, 0.15, 0.6, .75])
3               plt.rc('font', size=18)
4               for pressure in self.P_list:
5                   reg_x=numpy.arange(min(FS_list),max(FS_list),0.1)
6                   reg_y=numpy.empty(len(reg_x))
7                   for i in range(len(reg_x)):
8                       reg_y[i]=self.eq(g,reg_x[i],pressure)
9                   data_x=numpy.empty(len(FS_list))
10                  data_y=numpy.empty(len(FS_list))
11                  for i in range(len(FS_list)):
12                      data_x[i]=FS_list[i]
13                      data_y[i]=self.deflection(g,FS_list[i],pressure)
14                  acolor=next(color)
15                  plt.plot(reg_x, reg_y, label=str(pressure)+'␣
                      ↪ in␣H$_2$O', color=acolor)
16                  if self.B:
17                      plt.plot(reg_x, 0.9*reg_y, '—', reg_x,
                          ↪ 1.1*reg_y, '—', color=acolor)
18                  if not numpy.isnan(data_y).all():
19                      plt.plot(data_x,data_y,'o',color=acolor)
20                      if self.S:
21                          xn,yn=spline(data_x,data_y)
22                          plt.plot(xn,yn,'-',color=acolor)
23              plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                  ↪ borderaxespad=0.,
                  ↪ prop={'size':8},fontsize=14)
24              plt.xticks(numpy.arange(0,80,10))
25              plt.xlabel('Flatspan␣(inches)')
26              plt.ylabel('Deflection␣(inches)')
```

```
1               title=self.config+' duct deflection with respect
                    ↪ to flatspan at '+str(g)+' Ga.'
2               #plt.title("\n".join(wrap(title,60)))
3               self.savegraph('Ga_'+str(g))
4               plt.close()
5
6       def Fs_constant(self):
7           for FS in FS_list:
8               color=get_color(len(g_list))
9               fig=plt.figure()
10              fig.add_axes([0.15, 0.15, 0.6, .75])
11              plt.rc('font', size=18)
12              for g in g_list:
13                  reg_x=numpy.arange(min(self.P_list),max(self.P_list),0.1)
14                  reg_y=numpy.empty(len(reg_x))
15                  for i in range(len(reg_x)):
16                      reg_y[i]=self.eq(g,FS,reg_x[i])
17                  data_x=numpy.empty(len(self.P_list))
18                  data_y=numpy.empty(len(data_x))
19                  for i in range(len(data_x)):
20                      data_x[i]=self.P_list[i]
21                      data_y[i]=self.deflection(g,FS,self.P_list[i])
22                  acolor=next(color)
23                  plt.plot(reg_x,reg_y,label=str(g)+'
                        ↪ Ga',color=acolor)
24                  if self.B:
25                      plt.plot(reg_x, 0.9*reg_y, '—', reg_x,
                            ↪ 1.1*reg_y, '—', color=acolor)
26                  if not numpy.isnan(data_y).all():
27                      plt.plot(data_x,data_y,'o', color=acolor)
28                      if self.S:
```

```
                    xn,yn=spline(data_x,data_y)
                    plt.plot(xn,yn,'-',color=acolor)
            plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
              ↪ borderaxespad=0., fontsize=14)
            plt.xticks(numpy.arange(-6,12,2))
            plt.xlabel('Pressure (inches H$_2$0)')
            plt.ylabel('Deflection (inches)')
            title=self.config+' duct deflection with respect
              ↪ to pressure at '+str(FS)+' inches flatspan'
            #plt.title("\n".join(wrap(title,60)))
            self.savegraph('FS_'+str(FS))
            plt.close()

    def GaFs_constant(self):
        for g in g_list:
            color=get_color(len(FS_list))
            fig=plt.figure()
            fig.add_axes([0.15, 0.15, 0.6, .75])
            plt.rc('font', size=18)
            for FS in FS_list:
                reg_x=numpy.arange(min(self.P_list),max(self.P_list),0.1)
                reg_y=numpy.empty(len(reg_x))
                for i in range(len(reg_x)):
                    reg_y[i]=self.eq(g,FS,reg_x[i])
                data_x=numpy.empty(len(self.P_list))
                data_y=numpy.empty(len(data_x))
                for i in range(len(data_x)):
                    data_x[i]=self.P_list[i]
                    data_y[i]=self.deflection(g,FS,self.P_list[i])
                acolor=next(color)
```

```
1                   plt.plot(reg_x,reg_y,label=str(FS)+'
                      ↪ (inches)',color=acolor)
2                   if self.B:
3                       plt.plot(reg_x, 0.9*reg_y, '—', reg_x,
                          ↪ 1.1*reg_y, '—', color=acolor)
4                   if not numpy.isnan(data_y).all():
5                       plt.plot(data_x,data_y,'o', color=acolor)
6                       if self.S:
7                           xn,yn=spline(data_x,data_y)
8                           plt.plot(xn,yn,'-', color=acolor)
9               plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                  ↪ borderaxespad=0.,fontsize=14)
10              plt.xticks(numpy.arange(-6,12,2))
11              plt.xlabel('Pressure (inches H$_2$0)')
12              plt.ylabel('Deflection (inches)')
13              title=self.config+' duct deflection with respect
                  ↪ to pressure at '+str(g)+' Ga.'
14              #plt.title("\n".join(wrap(title,60)))
15              self.savegraph('GaFs_'+str(g))
16              plt.close()
17
18      def Ga_FsP(self):
19          for pressure in self.P_list:
20              color=get_color(len(self.P_list)+1)
21              fig=plt.figure()
22              fig.add_axes([0.15, 0.15, 0.6, .75])
23              plt.rc('font', size=18)
24              for FS in FS_list:
25                  reg_x=numpy.arange(min(g_list),max(g_list),0.1)
26                  reg_y=numpy.empty(len(reg_x))
27                  for i in range(len(reg_x)):
```

```
1                       reg_y[i]=self.eq(reg_x[i],FS,pressure)
2                   data_x=numpy.empty(len(g_list))
3                   data_y=numpy.empty(len(g_list))
4                   for i in range(len(g_list)):
5                       data_x[i]=g_list[i]
6                       data_y[i]=self.deflection(g_list[i],FS,pressure)
7                   acolor=next(color)
8                   plt.plot(reg_x,reg_y,label=str(FS)+' in',
                        ↪ color=acolor)
9                   if self.B:
10                      plt.plot(reg_x,0.9*reg_y,'—',reg_x,1.1*reg_y,'—',
                            ↪ color=acolor)
11                  if not numpy.isnan(data_y).all():
12                      plt.plot(data_x,data_y,'o', color=acolor)
13                      if self.S:
14                          xn,yn=spline(data_x,data_y)
15                          plt.plot(xn,yn,'-',color=acolor)
16              plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                    ↪ borderaxespad=0.,
                    ↪ prop={'size':8},fontsize=14)
17              plt.xticks(numpy.arange(18,28,2))
18              plt.xlabel('Gauge')
19              plt.ylabel('Deflection (inches)')
20              title=self.config+' duct deflection with respect
                    ↪ to gauge at '+str(pressure)+' (inches
                    ↪ H$_2$O)'
21              #plt.title("\n".join(wrap(title,60)))
22              self.savegraph('Ga_FsP_'+str(pressure))
23              plt.close()
24
25      def Ga_PFs(self):
26          for FS in FS_list:
27              color=get_color(len(self.P_list))
28              fig=plt.figure()
```

```
1               fig.add_axes([0.15, 0.15, 0.6, .75])
2               plt.rc('font', size=18)
3               for pressure in self.P_list:
4                   reg_x=numpy.arange(min(g_list),max(g_list),0.1)
5                   reg_y=numpy.empty(len(reg_x))
6                   for i in range(len(reg_x)):
7                       reg_y[i]=self.eq(reg_x[i],FS,pressure)
8                   data_x=numpy.empty(len(g_list))
9                   data_y=numpy.empty(len(g_list))
10                  for i in range(len(g_list)):
11                      data_x[i]=g_list[i]
12                      data_y[i]=self.deflection(g_list[i],FS,pressure)
13                  acolor=next(color)
14                  plt.plot(reg_x,reg_y,label=str(pressure)+'␣
                      ↪ in␣H$_2$O',color=acolor)
15                  if self.B:
16                      plt.plot(reg_x, 0.9*reg_y, '—', reg_x,
                          ↪ 1.1*reg_y, '—', color=acolor)
17                  if not numpy.isnan(data_y).all():
18                      plt.plot(data_x,data_y,'o', color=acolor)
19                      if self.S:
20                          xn,yn=spline(data_x,data_y)
21                          plt.plot(xn,yn,'-',color=acolor)
22              plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                  ↪ borderaxespad=0.,
                  ↪ prop={'size':8},fontsize=14)
23              plt.xticks(numpy.arange(18,28,2))
24              plt.xlabel('Gauge')
25              plt.ylabel('Deflection␣(inches)')
26              title=self.config+'␣duct␣deflection␣with␣respect␣
                  ↪ to␣gauge␣at␣'+str(FS)+'␣inches␣flatspan'
27              #plt.title("\n".join(wrap(title,60)))
```

```
1               self.savegraph('Ga_PFs_'+str(FS))
2               plt.close()
3
4       def savegraph(self,filename):
5           dirs='png/'+self.config
6           dirs=dirs.replace(' ','_')
7           if not os.path.exists(dirs):
8               os.makedirs(dirs)
9           dirs=dirs+'/'
10          filename=dirs+filename.replace(' ','_')+'.png'
11          plt.savefig(filename)
12
13  class Raw(object):
14      def __init__(self, Ducts, c, p):
15          self.Ducts=[]
16          self.config=c
17          self.PorN=p
18          for duct in Ducts:
19              if duct.config == c and duct.PorN == self.PorN
                    ↪ and duct.height == 16:
20                  self.Ducts.append(duct)
21
22      def reg(self, x, y):
23          fitfunc = lambda C, x: x*abs(C[0])+x*x*abs(C[1])
24          errfunc = lambda params, x, y: fitfunc(params, x) - y
25          init_p=numpy.zeros(2)
26          self.C, success = leastsq(errfunc, init_p.copy(),
                ↪ args=(x, y), maxfev=100000)
27          #xn=numpy.linspace(0,numpy.amax(x),200)
28          xn=numpy.linspace(0,70,200)
```

```
        yn=numpy.empty(len(xn))
        for i, xt in enumerate(xn):
            yn[i]=fitfunc(self.C, xt)
        return xn, yn


    def graph(self, pressure):
        color=get_color(len(g_list))
        fig=plt.figure()
        fig.add_axes([0.15, 0.15, 0.6, .75])
        plt.grid(True)
        plt.rc('font', size=18)
        for g in g_list:
            acolor=next(color)
            data_x=numpy.array([0.0])
            data_y=numpy.array([0.0])
            for i, FS in enumerate(FS_list):
                for duct in self.Ducts:
                    if duct.gauge == g and duct.flatspan ==
                    ↪ FS:
                        d=duct.find_deflectionL(pressure)
                        if d is not None:
                            data_x=numpy.append(data_x,[FS])
                            data_y=numpy.append(data_y,[abs(d)])
            if len(data_x)>=2:
                label=str(g)+' Ga'
                plt.plot(data_x,data_y,'o',color=acolor,label=label)
                xn,yn=self.reg(data_x,data_y)
                plt.plot(xn,yn,'-',color=acolor)
        plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
        ↪ borderaxespad=0.,fontsize=14)
```

```
1           plt.axis([0,70,0,2])
2           plt.xlabel('Flatspan (inches)')
3           plt.ylabel('Deflection (inches)')
4           title=self.config+' duct deflection with respect to
              ↪ flatspan at '+str(pressure)+' (inches H$_2$O)'
5           #plt.title("\n".join(wrap(title,60)))
6           self.savegraph('Raw_'+str(pressure))
7           plt.close()
8
9       def savegraph(self,filename):
10          dirs='png/'+self.config
11          dirs=dirs.replace(' ','_')
12          if not os.path.exists(dirs):
13              os.makedirs(dirs)
14          dirs=dirs+'/'
15          filename=dirs+filename.replace(' ','_')+'.png'
16          plt.savefig(filename)
17
18  class objgraph(object):
19      #These graphs are generated from objects rather than
          ↪ tables. They are the height differences and
          ↪ Reenforcement plots
20      def __init__(self, objects):
21          self.objects=objects
22
23      def run(self):
24          config=['None', 'Unreinforced', 'T-25 6ft lengths',
              ↪ 'T-25 12ft lengths', 'Attached reinforcing 3ft
              ↪ centers', 'Attached reinforcing 6ft centers',
              ↪ 'Trapeze 3ft centers', 'Trapeze 6ft centers',
              ↪ 'Tie Rod 3ft centers', 'Tie Rod 6ft centers']
```

```
1           for PorN in ['P', 'N']:
2               fig=plt.figure()
3               fig.add_axes([0.15, 0.15, 0.5, .75])
4               plt.rc('font', size=18.0)
5               for i in self.objects:
6                   if i.flatspan == 25 and i.gauge == 22 and
                        ↪ i.config == config[1] and i.PorN ==
                        ↪ PorN:
7                     plt.plot(i.Norm[0], i.Norm[1],
                          ↪ marker='o', label=str(i.height)+'␣
                          ↪ Height␣(inches)')
8               plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
                  ↪ borderaxespad=0.,fontsize=14)
9               plt.xlabel('Pressure␣(inches␣H$_2$O)')
10              plt.ylabel('Deflection␣(inches)')
11              filename='png/25'+PorN+'.png'
12              plt.savefig(filename)
13              plt.close()
14
15          things_P=[]
16          things_N=[]
17          for i in self.objects:
18              if (i.PorN == 'P') and (i.config != 'T-25␣12ft␣
                    ↪ lengths'):
19                  things_P.append(i.name)
20              elif (i.PorN == 'N') and (i.config != 'T-25␣12ft␣
                    ↪ lengths'):
21                  things_N.append(i.name)
22          things_P.sort()
23          things_N.sort()
```

```
1           for P in [1,2,4,6]:
2               data=numpy.empty((2,3))
3               data1=numpy.empty((2,3))
4               data2=numpy.empty((2,3))
5               data3=numpy.empty((2,3))
6               count=0
7               count1=0
8               count2=0
9               count3=0
10              for i in self.objects:
11                  if ('AH' in i.name) and 'P' == i.PorN and
                        ↪ P!=6:
12                      if i.config == config[1] or i.config ==
                            ↪ config[2] or i.config == config[3]:
13                          if i.config == config[1]:
14                              data[0][count]=12
15                          elif i.config == config[2]:
16                              data[0][count]=3
17                          elif i.config == config[3]:
18                              data[0][count]=6
19                          data[1][count]=i.find_deflection(P)
20                          count+=1
21
22                  if ('AC' in i.name) and 'P' == i.PorN:
23                      if i.config == config[1] or i.config ==
                            ↪ config[8] or i.config == config[9]:
24                          if i.config == config[1]:
25                              data1[0][count1]=12
26                          elif i.config == config[8]:
27                              data1[0][count1]=3
28                          elif i.config == config[9]:
```

```
                        data1[0][count1]=6
                    data1[1][count1]=i.find_deflection(P)
                    count1+=1
            if 'AF' in i.name and 'P' == i.PorN:
                if i.config == config[1] or i.config ==
                  ↪ config[4] or i.config == config[5]:
                    if i.config == config[1]:
                        data2[0][count2]=12
                    elif i.config == config[4]:
                        data2[0][count2]=3
                    elif i.config == config[5]:
                        data2[0][count2]=6
                    data2[1][count2]=i.find_deflection(P)
                    count2+=1
            if 'AF' in i.name and 'N' == i.PorN:
                if i.config == config[1] or i.config ==
                  ↪ config[4] or i.config == config[5]:
                    if i.config == config[1]:
                        data3[0][count3]=12
                    elif i.config == config[4]:
                        data3[0][count3]=3
                    elif i.config == config[5]:
                        data3[0][count3]=6
                    data3[1][count3]=i.find_deflection(-P)
                    count3+=1

        data=data[:,data[0].argsort(axis=0)]
        data1=data1[:,data1[0].argsort(axis=0)]
        data2=data2[:,data2[0].argsort(axis=0)]
        data3=data3[:,data3[0].argsort(axis=0)]
        plt.figure()
```

```
1               if P!=6:
2                   plt.plot(data[0],data[1],marker='o',label='AHP3␣
                        ↪ +'+str(P)+'in␣H$_2$O␣Rod')
3               plt.plot(data1[0],data1[1],marker='o',label='ACP9␣
                    ↪ +'+str(P)+'in␣H$_2$O␣Rod')
4               plt.plot(data2[0],data2[1],marker='o',label='AFP5␣
                    ↪ +'+str(P)+'in␣H$_2$O␣Angle')
5               plt.plot(data3[0],data3[1],marker='o',label='AFN5␣
                    ↪ -'+str(P)+'in␣H$_2$O␣Angle')
6               plt.legend(loc=2)
7               plt.xlabel('Renforcement␣Spacing␣(feet)')
8               plt.ylabel('Deflection␣(inches)')
9               filename='png/config_'+str(P)+'.png'
10              self.savegraph('P')
11              plt.close()
12
13
14      def savegraph(self,filename):
15          dirs='png/config'
16          dirs=dirs.replace('␣','_')
17          if not os.path.exists(dirs):
18              os.makedirs(dirs)
19          dirs=dirs+'/'
20          filename=dirs+filename.replace('␣','_')+'.png'
21          plt.savefig(filename)
```

```
1  #!/usr/bin/python2
2  import numpy
3  from Duct import *
4  from scipy.optimize import leastsq
5  from scipy.interpolate import interp1d
6  from copy import copy
7  FS_list=[63, 48, 38, 25, 14, 6]
8  g_list=[18, 20, 22, 24, 26]
9  P_list=[-6,-4,-2,-1,-0.5,0.5,1,2,4,6,10]
10 config=['Unreinforced', 'T-25 6ft lengths', 'T-25 12ft
       ↪ lengths', 'Attached reinforcing 3ft centers',
       ↪ 'Attached reinforcing 6ft centers', 'Trapeze 3ft
       ↪ centers', 'Trapeze 6ft centers', 'Tie Rod 3ft
       ↪ centers', 'Tie Rod 6ft centers']
11
12 def print_table(table):
13     text=""
14     for row in table:
15         for cell in row:
16             text+=str(cell)+'\t'
17         text=text[:-1]
18         text+='\n'
19     print text
20
21 def save_table(filename, table):
22     filename=filename.replace(' ','_')+'.csv'
23     text=""
24     for row in table:
25         for cell in row:
26             if cell==0.0:
27                 text+=""
```

```
            else:
                text+=str(cell)
            text+=','
        text=text[:-1]
        text+='\n'
    f=open(filename, 'w')
    f.write(text)
    f.close()

class table(object):
    def __init__(self,objects,PorN,config):
        self.ratio_gauge=20
        self.config=config
        self.PorN=PorN
        self.objects=[]
        for duct in objects:
            if (duct.config == self.config) and (duct.PorN
              ↪ == self.PorN) and (duct.height == 16):
                self.objects.append(duct)

        self.table=numpy.zeros([5,6,11])
        self.table[:]=numpy.NaN
        self.table2=numpy.zeros([5,6,11])
        self.table2[:]=numpy.NaN
        self.tableMapGa={}
        self.tableMapFs={}

        self.tableMapP={}
        for ig, g in enumerate(g_list):
            self.tableMapGa[str(g)]=ig
```

```
1               for iFS, FS in enumerate(FS_list):
2                   self.tableMapFs[str(FS)]=iFS
3                   for duct in self.objects:
4                       if (duct.flatspan == FS) and (duct.gauge
                            ↪ == g):
5                           for ip, p in enumerate(P_list):
6                               self.tableMapP[str(p)]=ip
7                               pres, d=duct.find_deflectionP(p)
8                               if d != None: #This is to keep
                                    ↪ it from over writting pos
                                    ↪ and neg on the same line
9                                   if d != None:
10                                      self.table[ig][iFS][ip]=d
11                                  if hasattr(self, 'array'):
12                                      self.array=numpy.append(self.array,[[g,
                                            ↪ FS, pres, d]],
                                            ↪ axis=0)
13                                  else:
14                                      self.array=numpy.array([[g,
                                            ↪ FS, pres, d]])
15          for row in self.array:
16              if row[3]>=1 and row [2]>=1:
17                  if 'tmp' in locals():
18                      tmp=numpy.append(tmp,[row],axis=0)
19                  else:
20                      tmp=numpy.array([row])
21          self.array=numpy.transpose(tmp)
22
23          Ga=self.array[0]
24          Fs=self.array[1]
25          P=self.array[2]
26          D=self.array[3]
```

```
1           print len(D)
2           self.fitfunc = lambda C, Ga, Fs, P: C[0]+ Ga*C[1]+
              ↪ Fs*C[2]+ P*C[3]+ Ga*Fs*C[4]+ Ga*P*C[5]+
              ↪ Fs*P*C[6]+ Ga*Ga*C[7]+ Fs*Fs*C[8]+ P*P*C[9]
3           errfunc = lambda params, Ga, Fs, P, D:
              ↪ self.fitfunc(params, Ga, Fs, P) - D
4           init_p=numpy.zeros(10)
5           self.C, success = leastsq(errfunc, init_p.copy(),
              ↪ args=(Ga, Fs, P, D))
6
7       #def save_C(self):
8       #     table=[]
9       #     for i, f in enumerate(self.C):
10      #         table.append(['C'+str(i), f])
11      #     filename='C'
12      #     dirs='png/'+self.config
13      #     dirs=dirs.replace(' ','_')
14      #     if not os.path.exists(dirs):
15      #         os.makedirs(dirs)
16      #     dirs=dirs+'/'+self.PorN+'/'
17      #     filename=dirs+filename.replace(' ','_')
18      #     save_table(filename, table)
19
20      def tableGa(self, gauge):
21          return numpy.insert(numpy.insert(
              ↪ self.table[self.tableMapGa[str(gauge)]], 0,
              ↪ P_list, axis=0), 0, [0]+FS_list, axis=1)
22
23      def tableGa_nohead(self, gauge):
24          return self.table[self.tableMapGa[str(gauge)]]
```

```
1
2       def tableFS(self, FS):
3           return numpy.insert(numpy.insert(self.table[:,
                ↪ self.tableMapFs[FS], :], 0, P_list, axis=0),
                ↪ 0, [0]+g_list, axis=1)
4
5       def tableP(self, P):
6           return numpy.insert(numpy.insert(self.table[:, :,
                ↪ self.tableMapP[P]], 0, FS_list, axis=0), 0,
                ↪ [0]+g_list, axis=1)
7
8       def deflection(self, Ga, FS, P):
9           return self.table[self.tableMapGa[str(Ga)]]
                ↪ [self.tableMapFs[str(FS)]][self.tableMapP[str(P)]]
10
11      def deflection_ratio(self, Ga, FS, P):
12          return self.table2[self.tableMapGa[str(Ga)]]
                ↪ [self.tableMapFs[str(FS)]][self.tableMapP[str(P)]]
13
14      def D(self, Ga, Fs, P): # Returns None if the model is
            ↪ not in range.
15          d=self.Def(Ga, Fs, P)
16          if self.PorN == 'P' and 0.75<P<11 and 2.75<d<25 and
                ↪ self.config=='Unreinforced':
17              return d
18          elif self.PorN == 'N' and 1<=abs(d)<=9 and
                ↪ abs(P)>0.75:# and d < 0:
19              return d
20          elif self.PorN == 'P' and 0.75<P<11 and 1.25<d<25
                ↪ and self.config=='T-25 12ft lengths':
21              return d
```

```
1           else:
2               return None
3
4       def Def(self, Ga, Fs, P):
5           return self.fitfunc(self.C, Ga, Fs, P)
6
7       def P(self, Ga, Fs, d):
8           # This finds as close to zero as it can I had
                ↪ problems with scipy.optimization functions.
9           f = lambda q: abs(d-self.Def(Ga, Fs, q))
10
11          if self.PorN == 'P':
12              #pressure=numpy.arange(0,10,0.001)
13              pressure=numpy.arange(0,10,0.1)
14          else:
15              #pressure=numpy.arange(-6,0,0.001)
16              pressure=numpy.arange(-6,0,0.1)
17          nf=100
18          for p in pressure:
19              tmp=f(p)
20              if tmp<nf:
21                  nf=tmp
22                  np=p
23          return np
24
25      def Fs(self, iGa, iFs, iP):
26          for row in numpy.transpose(self.array):
27              if row[1]==iFs:
28                  if 'tables' in locals():
29                      tables=numpy.append(tables,[row],axis=0)
30                  else:
```

```
1                       tables=numpy.array([row])
2
3           Ga=self.array[0]
4           P=self.array[2]
5           D=self.array[3]
6           fitfunc = lambda C, Ga, P:
                ↪ C[0]+Ga*C[1]+P*C[2]+Ga*P*C[3]+Ga*Ga*C[4]+P*P*C[5]
7           errfunc = lambda params, Ga, P, D: fitfunc(params,
                ↪ Ga, P) - D
8           init_p=numpy.zeros(6)
9           C, success = leastsq(errfunc, init_p.copy(),
                ↪ args=(Ga, P, D))
10          return round(fitfunc(C, iGa, iP), 2)
11
12      def Ga(self, iGa, iFs, iP):
13          for row in numpy.transpose(self.array):
14              if row[0]==iGa:
15                  if 'tables' in locals():
16                      tables=numpy.append(tables,[row],axis=0)
17                  else:
18                      tables=numpy.array([row])
19
20          Fs=self.array[1]
21          P=self.array[2]
22          D=self.array[3]
23          fitfunc = lambda C, Fs, P:
                ↪ C[0]+Fs*C[1]+P*C[2]+Fs*P*C[3]+Fs*Fs*C[4]+P*P*C[5]
24          errfunc = lambda params, Fs, P, D: fitfunc(params,
                ↪ Fs, P) - D
25          init_p=numpy.zeros(6)
```

```
1           C, success = leastsq(errfunc, init_p.copy(),
                ↪ args=(Fs, P, D))
2           return round(fitfunc(C, iFs, iP), 2)
3
4   class ptable(object):
5       def __init__(self, objects, tables, config, PorN):
6           self.objects=objects
7           for p in tables:
8               if p.config == config and p.PorN == PorN:
9                   self.poly=p
10          self.config=config
11          self.PorN=PorN
12
13      def contourP(self, P):
14          g=g_list
15          FS_list=[63, 48, 38, 25, 14, 6]
16          FS=FS_list
17          D=numpy.empty([len(FS),len(g)])
18          for x, ig in enumerate(g):
19              for y, iFS in enumerate(FS):
20                  val=self.poly.D(ig,iFS,P)
21                  if val is not None:
22                      val=round(val,1)
23                      if self.PorN == 'P' and (val<2.75 or
                            ↪ val>25) and self.config ==
                            ↪ 'Unreinforced':
24                          D[y][x]=0
25                      elif self.PorN == 'P' and (val<1.25 or
                            ↪ val>25) and self.config == 'T-25␣
                            ↪ 12ft␣lengths':
26                          D[y][x]=0
```

```
                    elif self.PorN == 'N' and (val>-1 or
                        ↪ val<-9) and self.config ==
                        ↪ 'Unreinforced':
                        D[y][x]=0
                    else:
                        D[y][x]=val
                else:
                    D[y][x]=0
        D=numpy.insert(D,0,g,axis=0)
        FS.insert(0,0)
        D=numpy.insert(D,0,FS,axis=1)
        D=numpy.rint(D).astype(int)
        dirs=self.config+'/'+self.PorN
        dirs=dirs.replace(' ','_')
        filename='P_Contour'+'_'+str(P)
        filename='png/'+dirs+'/'+filename.replace(' ','_')
        save_table(filename,D)

    def contourD(self, d):
        g=g_list
        FS_list=[63, 48, 38, 25, 14, 6]
        FS=FS_list
        P=numpy.empty([len(FS),len(g)])
        for x, ig in enumerate(g):
            for y, iFS in enumerate(FS):
                val=round(self.poly.P(ig,iFS,d),1)
                if (val<0.75 or val>10) and self.PorN == 'P':
                    P[y][x]=0
                elif (val>-1 or val<-6) and self.PorN == 'N':
                    P[y][x]=0
                else:
                    P[y][x]=val
```

```
            P=numpy.insert(P,0,g,axis=0)
            FS.insert(0,0)
            P=numpy.insert(P,0,FS,axis=1)
            P=numpy.rint(P).astype(int)
            dirs=self.config+'/'+self.PorN
            dirs=dirs.replace(' ','_')
            filename='D_Contour'+'_'+str(d)
            filename='png/'+dirs+'/'+filename.replace(' ','_')
            save_table(filename,P)

class matrix(object):
    def __init__(self, objects, tables):
        self.objects=objects
        self.tables=tables

    def run(self):
        f=open('house.txt', 'r')
        names=f.readlines()
        f.close()
        houses=[]
        for i in names:
            houses.append(House(i))
        del names
        f=open('Duct_matrix.csv', 'r')
        names=f.readlines()
        f.close()
        quote=[]
        for i in names:
            quote.append(House(i))
        self.matrixp=[]
        self.matrixn=[]
```

```
1           self.matp=[]
2           self.matn=[]
3           line1=['Gauge']
4           line2=['Flatspan']
5           for g in g_list:
6               for FS in FS_list:
7                   line1.append(g)
8                   line2.append(FS)
9           self.matrixp.append(line1)
10          self.matrixp.append(line2)
11          self.matrixn.append(line1)
12          self.matrixn.append(line2)
13          self.matp.append(line1)
14          self.matp.append(line2)
15          self.matn.append(line1)
16          self.matn.append(line2)
17          Ccountp=0
18          Ccountn=0
19          Hcountp=0
20          Hcountn=0
21          for c in config:
22              linep=[c]
23              linen=[c]
24              matlinep=[c]
25              matlinen=[c]
26              for g in g_list:
27                  for FS in FS_list:
28                      cellp=' '
29                      celln=' '
30                      matp=' '
31                      matn=' '
```

```
1                   #for i in quote:
2                   #    if (i.config == c) and (i.gauge ==
                        ↪ g) and (i.flatspan == FS) and
                        ↪ (i.height == 16):
3                   #        if (i.PorN == 'P'):
4                   #            cellp='Q'
5                   #        elif (i.PorN == 'N'):
6                   #            celln='Q'
7                   #for i in houses:
8                   #    if (i.config == c) and (i.gauge ==
                        ↪ g) and (i.flatspan == FS) and
                        ↪ (i.height == 16):
9                   #        if (i.PorN == 'P'):
10                  #            cellp='H '+i.num
11                  #        elif (i.PorN == 'N'):
12                  #            celln='H: '+i.num
13                  for i in self.objects:
14                      if (i.config == c) and (i.gauge ==
                            ↪ g) and (i.flatspan == FS) and
                            ↪ (i.height == 16):
15                              p=round(i.Norm[0][-1],1)
16                              d=round(i.Norm[1][-1],1)
17                              if p>6.1:
18                                  p=round(i.Norm[0][-2],1)
19                                  d=round(i.Norm[1][-2],1)
20                              if (i.PorN == 'P'):
21                                  cellp='P: '+str(p)+'␣
                                        ↪ D: '+str(d)
22                                  matp='T'
23                              elif (i.PorN == 'N'):
```

```
1                                   celln='P: '+str(-1*p)+'␣
                                        ↪ D: '+str(-1*d)
2                                   matn='T'
3                           for t in self.tables:
4                               if c == t.config:
5                                   if (t.PorN == 'P'):
6                                       p=6.0
7                                       d=t.D(g,FS,p)
8                                       if d is not None:
9                                           d=round(d,1)
10                                          if d>2.75 and
                                                ↪ c=='Unreinforced':
11                                              cellp='P: '+str(p)+'␣
                                                    ↪ D: '+str(d)
12                                          elif d>1.25 and c=='T-25␣
                                                ↪ 12ft␣lengths':
13                                              cellp='P: '+str(p)+'␣
                                                    ↪ D: '+str(d)
14                                  elif (t.PorN == 'N'):
15                                      p=-6.0
16                                      d=t.D(g,FS,p)
17                                      if (d is not None) and d <
                                            ↪ -1 and d > -9:
18                                          d=round(d,1)
19                                          celln='P: '+str(p)+'␣
                                                ↪ D: '+str(d)
20                          linep.append(cellp)
21                          linen.append(celln)
22                          matlinep.append(matp)
23                          matlinen.append(matn)
24                  self.matrixp.append(linep)
```

```
            self.matrixn.append(linen)
            self.matp.append(matlinep)
            self.matn.append(matlinen)
        self.save_mat()

    def save_mat(self):
        for i in [['matrixp', self.matrixp], ['matrixn',
            ↪ self.matrixn]]:
            filename=i[0]
            table=i[1]
            filename='png/'+filename.replace(' ','_')+'.csv'
            s=""
            for row in table:
                for cell in row:
                    s+=str(cell)+','
                s=s[:-1]
                s+='\n'
            f=open(filename, 'w')
            f.write(s)
            f.close()

        for i in [['matp', self.matp], ['matn', self.matn]]:
            filename=i[0]
            table=i[1]
            filename='png/'+filename.replace(' ','_')+'.csv'
            s=""
            for row in table:
                for cell in row:
                    s+=str(cell)+','
                s=s[:-1]
                s+='\n'
```

```
1               f=open(filename, 'w')
2               f.write(s)
3               f.close()
4
5       def GaFs(self):
6           for PorN in ['P', 'N']:
7               for c in config:
8                   line=['␣']
9                   for g in g_list:
10                      line.append(g)
11                  array=[line]
12                  for FS in FS_list:
13                      line=[FS]
14                      for g in g_list:
15                          cell='␣'
16                          for duct in self.objects:
17                              if duct.flatspan==FS and
                                    ↪ duct.gauge==g and
                                    ↪ duct.config==c and
                                    ↪ duct.PorN==PorN:
18                                  cell=round(duct.Pe,2)
19                      line.append(cell)
20                  array.append(line)
21              save_table('GaFs_'+c+'_'+PorN, array)
22
23      def GaFsM(self):
24          for i in self.poly:
25              line=['␣']
26              for g in g_list:
```

```
1                   line.append(g)
2               array=[line]
3               for FS in FS_list:
4                   line=[FS]
5                   for g in g_list:
6                       line.append(round(i.Pe(g,FS),2))
7                   array.append(line)
8               save_table('GaFsM_'+i.config+'_'+i.PorN, array)
```

```
1  #!/usr/bin/python2
2  import numpy
3  import matplotlib as mpl
4  mpl.use('Agg')
5  import matplotlib.pyplot as plt
6  from operator import itemgetter
7  from Duct import *
8  from table import *
9
10 def m(Ga): #marker shape for the graph determined by gauge
11     if Ga==18:
12         return 'D'
13     elif Ga==20:
14         return 'H'
15     elif Ga==22:
16         return 'o'
17     elif Ga==24:
18         return '^'
19     elif Ga==26:
20         return 's'
21     else:
22         print 'self.m('+str(Ga)+') broke'
23
24 def co(Fs): #Color of the marker determined by flat span
25     if Fs==6:
26         return 'b'
27     elif Fs==14:
28         return 'g'
29     elif Fs==25:
30         return 'r'
```

```
    elif Fs==38:
        return 'm'
    elif Fs==48:
        return 'y'
    elif Fs==63:
        return 'k'
    else:
        print 'self.co('+str(Fs)+') broke'

def re(C): #Color around the marker determined by reinforcing
    if C=='T-25 6ft lengths':
        return 'g', '2'
    elif C=='T-25 12ft lengths':
        return 'r', '3'
    elif C=='Attached reinforcing 3ft centers':
        return 'm', '4'
    elif C=='Attached reinforcing 6ft centers':
        return 'y', '5'
    elif C=='Trapeze 3ft centers':
        return 'k', '6'
    elif C=='Trapeze 6ft centers':
        return '0.75', '7'
    elif C=='Tie Rod 3ft centers':
        return 'b', '8'
    elif C=='Tie Rod 6ft centers':
        return '0.25', '9'
    else:
        print 'self.re('+str(C)+') broke'
        return None, None
```

```
1  class ratio (object):
2      def __init__(self, table, objects, config):
3          self.config=config
4          self.PorN=table.PorN
5          self.objects=[]
6          self.table=table
7          for duct in objects:
8              if (duct.config == self.config) and (duct.PorN
                   ↪ == self.PorN) and (duct.height == 16):
9                  self.objects.append(duct)
10         if self.PorN=='P':
11             self.P_list=[1,2,4,6,10]
12         else:
13             self.P_list=[-6,-4,-2,-1]
14
15     def run(self):
16         fig=plt.figure()
17         fig.add_axes([0.15, 0.15, 0.5, 0.75])
18         plt.rc('font', size=18.0)
19         self.array=[]
20         for duct in self.objects:
21             for p in P_list:
22                 E=duct.find_deflection(p)
23                 M=self.table.D(duct.gauge, duct.flatspan, p)
24                 if (E is not None) and (M is not None) and
                       ↪ (E>=0):
25                     if (M>2.75 and M<25 and self.PorN=='P')
                           ↪ or (M>1 and M<9 and self.PorN ==
                           ↪ 'N'):
26                         ratio=E/M
```

```
1                           self.array.append([self.config,
                              ↪ self.PorN, duct.gauge,
                              ↪ duct.flatspan, p, round(E,2),
                              ↪ round(M,2), round(ratio,2)])
2                           if 'x' in locals():
3                               x=numpy.append(x,p)
4                               y=numpy.append(y,ratio)
5                           else:
6                               x=numpy.array(p)
7                               y=numpy.array(ratio)
8               if 'x' in locals():
9                   line='Ga: '+str(duct.gauge)+' Fs: '
                      ↪ '+str(duct.flatspan)
10                  plt.plot(x, y, marker=m(duct.gauge),
                      ↪ color=co(duct.flatspan), lw=0,
                      ↪ label=line)
11                  if numpy.size(x)>=2:
12                      xn,yn=spline(x,y)
13                      plt.plot(xn,yn,'-',lw=0.5,color=co(duct.flatspan))
14                  del x, y
15
16          plt.xlabel('Pressure (in. H$_2$O)')
17          plt.xlabel('Pressure (in. H$_2$O)')
18          plt.ylabel('Deflection/Unreinforced Deflection
              ↪ Model')
19          plt.yticks(numpy.arange(0,2,0.5))
20          if self.PorN == 'P':
21              plt.xticks(numpy.arange(0,12,2))
22          else:
23              plt.xticks(numpy.arange(0,8,2))
24          plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
              ↪ borderaxespad=0., fontsize=14)
25          filename='ratio.png'
```

```
1           dirs='png/'+self.config.replace(' ␣
               ↪ ','_')+'/'+self.PorN
2           if not os.path.exists(dirs):
3               os.makedirs(dirs)
4           filename=dirs+'/'+filename.replace(' ␣',' _')
5           if 'xn' in locals():
6               plt.savefig(filename)
7           #else:
8           #     print 'No points on '+self.config+' '+self.PorN
9           plt.close()
10          #print_table(self.array)
11
12  class ratioall (object):
13      def __init__(self, table, objects, spacing):
14          self.PorN=table.PorN
15          self.objects=[]
16          self.table=table
17          self.spacing=spacing
18          for duct in objects:
19              if (duct.PorN == self.PorN) and (duct.height ==
                   ↪ 16) and (duct.config != 'Unreinforced'):
20                  if self.spacing is not None:
21                      if (duct.spacing == self.spacing):
22                          self.objects.append(duct)
23                  else:
24                          self.objects.append(duct)
25          if self.PorN=='P':
26              self.P_list=[1,2,4,6,10]
27          else:
28              self.P_list=[-6,-4,-2,-1]
```

```
1     def run(self):
2         fig=plt.figure()
3         fig.add_axes([0.15, 0.15, 0.5, 0.75])
4         plt.rc('font', size=18.0)
5         self.array=[]
6         self.avg=[]
7         for duct in self.objects:
8             for p in P_list:
9                 E=duct.find_deflection(p)
10                M=self.table.D(duct.gauge, duct.flatspan, p)
11                if (E is not None) and (M is not None) and
                      ↪ (E>=0):
12                    if (M>2.75 and M<25 and self.PorN=='P')
                          ↪ or (M>1 and M<9 and self.PorN ==
                          ↪ 'N'):
13                        ratio=E/M
14                        self.array.append([duct.config,
                              ↪ duct.gauge, duct.flatspan, p,
                              ↪ round(E,2), round(M,2),
                              ↪ round(ratio,2)])
15                        if 'x' in locals():
16                            x=numpy.append(x,p)
17                            y=numpy.append(y,ratio)
18                        else:
19                            x=numpy.array(p)
20                            y=numpy.array(ratio)
21            if 'x' in locals():
22                r, c=re(duct.config)
23                line='Ga: '+str(duct.gauge)+' Fs: '
                      ↪ '+str(duct.flatspan)+' C: '+self.PorN+c
24                if numpy.size(x)>=2:
```

```
1                       plt.plot(x,y,marker=m(duct.gauge),
                            ↪ color=co(duct.flatspan), mec=r,
                            ↪ mew=2, lw=0, label=line, ms=8)
2                       xn,yn=spline(x,y)
3                       plt.plot(xn,yn,'-',lw=0.5,color=co(duct.flatspan))
4                   self.avg.append([duct.config, duct.gauge,
                        ↪ duct.flatspan,
                        ↪ round(numpy.average(y),2)])
5                   del x, y
6
7           plt.xlabel('Pressure (in. H$_2$O)')
8           plt.ylabel('Deflection/Unreinforced Deflection
                ↪ Model')
9           if self.PorN == 'P':
10              plt.yticks(numpy.arange(0,1.5,0.5))
11          else:
12              plt.yticks(numpy.arange(0,2,0.5))
13          if self.PorN == 'P':
14              plt.xticks(numpy.arange(0,12,2))
15          else:
16              plt.xticks(numpy.arange(0,8,2))
17          plt.legend(bbox_to_anchor=(1.05,1), loc=2,
                ↪ borderaxespad=0., fontsize=14)
18          filename='ratio.png'
19          if self.spacing is None:
20              dirs='png/'+self.PorN
21          else:
22              dirs='png/'+self.PorN+str(self.spacing)
23          if not os.path.exists(dirs):
24              os.makedirs(dirs)
25          filename=dirs+filename.replace(' ','_')
26          if 'xn' in locals():
```

```
1            plt.savefig(filename)
2        #else:
3        #    print 'No points on '+self.config+' '+self.PorN
4        plt.close()
5        #self.array.sort(key=lambda x: x[4])
6        self.array=sorted(self.array,
             ↪ key=itemgetter(3,1,2,6))
7        self.array.insert(0, ['Config', 'Gauge', 'Flatspan',
             ↪ 'Pressure', 'E', 'M', 'R'])
8        self.avg=sorted(self.avg, key=itemgetter(1,2,3))
9        self.avg.insert(0, ['Config', 'Gauge', 'Flatspan',
             ↪ 'Average'])
10       #save_table('png/avg'+self.PorN, self.array)
11       print 'Not␣Saving␣png/avg'+self.PorN #Tables were
             ↪ manually cleaned up, so I don't want it
             ↪ overwriting them. I do want it to remind me
             ↪ they aren't being updated.
12       #print_table(self.array)
```

```
1  #!/usr/bin/python2
2  import numpy
3  import matplotlib as mpl
4  mpl.use('Agg')
5  import matplotlib.pyplot as plt
6  from Duct import *
7  from math import pi
8  class height (object):
9      def __init__(self, Ducts, p):
10         self.PorN=p
11         self.Ducts=[]
12         for duct in Ducts:
13             if duct.config=='Unreinforced' and duct.PorN ==
               ↪ self.PorN:
14                 if (duct.flatspan == 25 and duct.gauge ==22)
                   ↪ or (duct.flatspan == 14 and duct.gauge
                   ↪ == 22) or (duct.flatspan == 25 and
                   ↪ duct.gauge == 26) or (duct.flatspan ==
                   ↪ 6 and duct.gauge ==24):
15                     self.Ducts.append(duct)
16         if self.PorN=='P':
17             self.P_list=[1,2,4,6,10]
18         else:
19             self.P_list=[-6,-4,-2,-1]
20
21     def run(self):
22         fig=plt.figure()
23         fig.add_axes([0.15, 0.15, 0.5, 0.75])
24         plt.rc('font', size=18.0)
```

```
1           for duct in self.Ducts:
2               if duct.height == 16:
3                   if duct.gauge == 22 and duct.flatspan == 14:
4                       c='k'
5                   elif duct.gauge == 26:
6                       c='b'
7                   elif duct.flatspan == 6:
8                       c='m'
9                   else:
10                      c='y'
11              elif duct.height == 30:
12                  c='g'
13              elif duct.height == 4:
14                  c='r'
15              dh=(4*duct.height*duct.flatspan+pi*duct.height**2)
                    ↪ /(pi*duct.height+2*duct.flatspan)
16              line='H: '+str(duct.height)+'
                    ↪ Fs: '+str(duct.flatspan)+'
                    ↪ Ga: '+str(duct.gauge)
17              x=duct.Norm[0]
18              y=duct.Norm[1]
19              if duct.name == '12-572AM_N1':
20                  x=numpy.delete(x,-1)
21                  x=numpy.delete(x,-1)
22                  y=numpy.delete(y,-1)
23                  y=numpy.delete(y,-1)
24              xn,yn=spline(x,y)
25              plt.plot(xn,yn,'-',lw=0.5,color=c)
26              plt.plot(x,y,'o',color=c,label=line)
27          plt.xlabel('Pressure (in. H$_2$O)')
28          plt.ylabel('Deflection (inches)')
```

```
1           plt.ylim(ymin=0)
2           if self.PorN == 'P':
3               plt.xlim(xmax=10)
4           else:
5               plt.xlim(xmax=6)
6           plt.legend(bbox_to_anchor=(1.05,1), loc=2,
              ↪ borderaxespad=0., fontsize=14)
7           self.name='Height '+self.PorN
8           #plt.title(self.name)
9           filename=self.name+'.png'
10          filename='png/'+filename.replace(' ','_')
11          plt.savefig(filename)
12          plt.close()
13
14
15          fig=plt.figure()
16          fig.add_axes([0.15, 0.15, 0.5, 0.75])
17          plt.rc('font', size=18.0)
18          for duct in self.Ducts:
19              if duct.height == 16:
20                  if duct.gauge == 22 and duct.flatspan == 14:
21                      c='k'
22                  elif duct.gauge == 26:
23                      c='b'
24                  elif duct.flatspan == 6:
25                      c='m'
26                  else:
27                      c='y'
28              elif duct.height == 30:
29                  c='g'
```

```
1               elif duct.height == 4:
2                   c='r'
3               dh=(4*duct.height*duct.flatspan+pi*duct.height**2)
                    ↪ /(pi*duct.height+2*duct.flatspan)
4               line='H: '+str(duct.height)+'
                    ↪ Fs: '+str(duct.flatspan)+'
                    ↪ Ga: '+str(duct.gauge)
5               x=duct.Norm[0]
6               y=duct.Norm[1]/duct.width
7               if duct.name == '12-572AM_N1':
8                   x=numpy.delete(x,-1)
9                   x=numpy.delete(x,-1)
10                  y=numpy.delete(y,-1)
11                  y=numpy.delete(y,-1)
12              xn,yn=spline(x,y)
13              plt.plot(xn,yn,'-',lw=0.5,color=c)
14              plt.plot(x,y,'o',color=c,label=line)
15          plt.xlabel('Pressure (in. H$_2$O)')
16          plt.ylabel('Deflection/width')
17          plt.ylim(ymin=0)
18          if self.PorN == 'P':
19              plt.xlim(xmax=10)
20          else:
21              plt.xlim(xmax=6)
22          plt.legend(bbox_to_anchor=(1.05,1), loc=2,
                ↪ borderaxespad=0., fontsize=14)
23          self.name='Height W'+self.PorN
24          #plt.title(self.name)
25          filename=self.name+'.png'
26          filename='png/'+filename.replace(' ','_')
27          plt.savefig(filename)
28          plt.close()
```

```
1           fig=plt.figure()
2           fig.add_axes([0.15, 0.15, 0.5, 0.75])
3           plt.rc('font', size=18.0)
4           for duct in self.Ducts:
5               if duct.height == 16:
6                   if duct.gauge == 22 and duct.flatspan == 14:
7                       c='k'
8                   elif duct.gauge == 26:
9                       c='b'
10                  elif duct.flatspan == 6:
11                      c='m'
12                  else:
13                      c='y'
14              elif duct.height == 30:
15                  c='g'
16              elif duct.height == 4:
17                  c='r'
18              dh=(4*duct.height*duct.flatspan+pi*duct.height**2)
                    ↪ /(pi*duct.height+2*duct.flatspan)
19              line='H: '+str(duct.height)+'
                    ↪ Fs: '+str(duct.flatspan)+'
                    ↪ Ga: '+str(duct.gauge)
20              x=duct.Norm[0]
21              y=duct.Norm[1]/duct.height
22              if duct.name == '12-572AM_N1':
23                  x=numpy.delete(x,-1)
24                  x=numpy.delete(x,-1)
25                  y=numpy.delete(y,-1)
26                  y=numpy.delete(y,-1)
27              xn,yn=spline(x,y)
28              plt.plot(xn,yn,'-',lw=0.5,color=c)
29              plt.plot(x,y,'o',color=c,label=line)
```

```
1           plt.xlabel('Pressure (in. H$_2$O)')
2           plt.ylabel('Deflection/Height')
3           plt.ylim(ymin=0)
4           if self.PorN == 'P':
5               plt.xlim(xmax=10)
6           else:
7               plt.xlim(xmax=6)
8           plt.legend(bbox_to_anchor=(1.05,1), loc=2,
              ↪ borderaxespad=0., fontsize=14)
9           self.name='Height H'+self.PorN
10          #plt.title(self.name)
11          filename=self.name+'.png'
12          filename='png/'+filename.replace(' ','_')
13          plt.savefig(filename)
14          plt.close()
15
16          fig=plt.figure()
17          fig.add_axes([0.15, 0.15, 0.5, 0.75])
18          plt.rc('font', size=18.0)
19          for duct in self.Ducts:
20              if duct.height == 16:
21                  if duct.gauge == 22 and duct.flatspan == 14:
22                      c='k'
23                  elif duct.gauge == 26:
24                      c='b'
25                  elif duct.flatspan == 6:
26                      c='m'
27                  else:
28                      c='y'
29              elif duct.height == 30:
30                  c='g'
```

```
1                elif duct.height == 4:
2                    c='r'
3                dh=(4*duct.height*duct.flatspan+pi*duct.height**2)
                     ↪ /(pi*duct.height+2*duct.flatspan)
4                line='H: '+str(duct.height)+'
                     ↪ Fs: '+str(duct.flatspan)+'
                     ↪ Ga: '+str(duct.gauge)
5                x=duct.Norm[0]
6                y=duct.Norm[1]/duct.flatspan
7                if duct.name == '12-572AM_N1':
8                    x=numpy.delete(x,-1)
9                    x=numpy.delete(x,-1)
10                   y=numpy.delete(y,-1)
11                   y=numpy.delete(y,-1)
12               xn,yn=spline(x,y)
13               plt.plot(xn,yn,'-',lw=0.5,color=c)
14               plt.plot(x,y,'o',color=c,label=line)
15           plt.xlabel('Pressure (in. H$_2$O)')
16           plt.ylabel('Deflection/flatspan')
17           plt.ylim(ymin=0)
18           if self.PorN == 'P':
19               plt.xlim(xmax=10)
20           else:
21               plt.xlim(xmax=6)
22           plt.legend(bbox_to_anchor=(1.05,1), loc=2,
                ↪ borderaxespad=0., fontsize=14)
23           self.name='Height FS'+self.PorN
24           #plt.title(self.name)
25           filename=self.name+'.png'
26           filename='png/'+filename.replace(' ','_')
27           plt.savefig(filename)
28           plt.close()
```

```
1           fig=plt.figure()
2           fig.add_axes([0.15, 0.15, 0.5, 0.75])
3           plt.rc('font', size=18.0)
4           for duct in self.Ducts:
5               if duct.height == 16:
6                   if duct.gauge == 22 and duct.flatspan == 14:
7                       c='k'
8                   elif duct.gauge == 26:
9                       c='b'
10                  elif duct.flatspan == 6:
11                      c='m'
12                  else:
13                      c='y'
14              elif duct.height == 30:
15                  c='g'
16              elif duct.height == 4:
17                  c='r'
18              dh=(4*duct.height*duct.flatspan+pi*duct.height**2)
                    ↪ /(pi*duct.height+2*duct.flatspan)
19              line='H: '+str(duct.height)+'
                    ↪ Fs: '+str(duct.flatspan)+'
                    ↪ Ga: '+str(duct.gauge)
20              ASS=duct.width/duct.height #Aspect ratio
21              x=duct.Norm[0]
22              y=duct.Norm[1]*ASS
23              if duct.name == '12-572AM_N1':
24                  x=numpy.delete(x,-1)
25                  x=numpy.delete(x,-1)
26                  y=numpy.delete(y,-1)
27                  y=numpy.delete(y,-1)
28              xn,yn=spline(x,y)
29              plt.plot(xn,yn,'-',lw=0.5,color=c)
```

```
            plt.plot(x,y,'o',color=c,label=line)
        plt.xlabel('Pressure (in. H$_2$O)')
        plt.ylabel('Deflection*Aspect Ratio (in)')
        plt.ylim(ymin=0)
        if self.PorN == 'P':
            plt.xlim(xmax=10)
        else:
            plt.xlim(xmax=6)
        plt.legend(bbox_to_anchor=(1.05,1), loc=2,
           ↪ borderaxespad=0., fontsize=14)
        self.name='Height AM'+self.PorN
        #plt.title(self.name)
        filename=self.name+'.png'
        filename='png/'+filename.replace(' ','_')
        plt.savefig(filename)
        plt.close()

        fig=plt.figure()
        fig.add_axes([0.15, 0.15, 0.5, 0.75])
        plt.rc('font', size=18.0)
        for duct in self.Ducts:
            if duct.height == 16:
                if duct.gauge == 22 and duct.flatspan == 14:
                    c='k'
                elif duct.gauge == 26:
                    c='b'
                elif duct.flatspan == 6:
                    c='m'
                else:
                    c='y'
            elif duct.height == 30:
```

```
1                   c='g'
2               elif duct.height == 4:
3                   c='r'
4               dh=(4*duct.height*duct.flatspan+pi*duct.height**2)
                    ↪ /(pi*duct.height+2*duct.flatspan)
5               line='H: '+str(duct.height)+'
                    ↪ Fs: '+str(duct.flatspan)+'
                    ↪ Ga: '+str(duct.gauge)
6               ASS=duct.width/duct.height
7               x=duct.Norm[0]
8               y=duct.Norm[1]/ASS
9               if duct.name == '12-572AM_N1':
10                  x=numpy.delete(x,-1)
11                  x=numpy.delete(x,-1)
12                  y=numpy.delete(y,-1)
13                  y=numpy.delete(y,-1)
14              xn,yn=spline(x,y)
15              plt.plot(xn,yn,'-',lw=0.5,color=c)
16              plt.plot(x,y,'o',color=c,label=line)
17          plt.xlabel('Pressure (in. H$_2$O)')
18          plt.ylabel('Deflection/Aspect Ratio (in)')
19          plt.ylim(ymin=0)
20          if self.PorN == 'P':
21              plt.xlim(xmax=10)
22          else:
23              plt.xlim(xmax=6)
24          plt.legend(bbox_to_anchor=(1.05,1), loc=2,
                ↪ borderaxespad=0., fontsize=14)
25          self.name='Height AD'+self.PorN
```

```
1           #plt.title(self.name)
2           filename=self.name+'.png'
3           filename='png/'+filename.replace(' ','_')
4           plt.savefig(filename)
5           plt.close()
```