Spring 2021

Projectile generator design for underground coal mine seal testing

Ethan Allan Steward

PROJECTILE GENERATOR DESIGN FOR UNDERGROUND COAL MINE SEAL TESTING

by

ETHAN ALLAN STEWARD

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN EXPLOSIVES ENGINEERING

2020

Approved by:

Dr. Kyle Perry, Advisor
Dr. Gillian Worsey
Dr. Paul Worsey

ABSTRACT

Underground coal mines have several dangers, one of the most hazardous of which is the possibility of an explosion caused by the ignition of methane gas. To reduce ventilation costs, coal mines have the option to close off abandoned areas that are no longer active with structural seals to keep an explosion from propagating into working areas of the mine. Though seals have been designed to resist the overpressure, none have been evaluated for their resistance to the impact of heavy objects. Underground explosions produce high velocity gasses that, traveling through mine openings, may propel objects in the mine at a high velocity causing them to impact and damage the mine seals. Seals damaged in this manner may no longer hold the pressure they were originally rated for and may fail during subsequent explosions or leak explosive gasses into active workings.

To test the effects of impact on mine seals, a projectile generator and two seals have been constructed. The projectile generator is a thick-walled steel pipe with a 4-inch wall, is 8.5 feet in length, and sealed on one end. Projectiles are propelled by a charge of black powder with a wooden wad and are held in place by foam sabots in the bore. This system allows firing of many different potential projectiles that may be found in an underground mine. The velocity of each projectile is measured by an infrared chronometer which can later be used to determine its impact energy. Impact effects on the seals are measured using strain gauges, LIDAR scans before and after impact, high speed cameras, and visual inspection. The focus of this research is on the projectile generator design, initial testing, and analysis of impacts on the mine seals.

ACKNOWLEDGEMENTS

First and foremost, I want to thank God, my *raison d'être*, for the fascinating and complex world in which I get to play and study. As S.M. Lockridge said, “My King is the key of knowledge, he’s the wellspring of wisdom”, and as a student for these last eight and a half years I have very much enjoyed learning about Him and His world.

Next, I want to thank my family for their love, encouragement, and wise advice. You have all been a great source of support for me in this long period of higher education.

I also want to thank Dr. Kyle Perry, my advisor, for giving me this project to work on, it’s been a blast! Thanks are also due the Alpha Foundation for the project’s funding, Strata Mine Services for their assistance in pouring the seals, various friends for volunteering to help, and last but certainly not least the Missouri S&T mine and rock mechanics faculty and staff who gave so much of their time, effort, wisdom, and ideas to make this project a success. I could not have done it without you all.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

Symbol	Description
b	covolume
°C	degrees Celcius
CAD	computer assisted drawing
CNC	computer numerical control
D	outer diameter, inches
ft-lbs.	foot pounds
ft/s	feet per second
FS	factor of safety
g/cm^3	grams per centimeter cubed
HSS	hollow structural section
in^2	inches squared
IR	infrared
kg	kilogram
kips	thousand pounds force
kJ	kilojoules
kPa	kilopascals
lb.	pounds
LIDAR	light detection and ranging
m_g	mass, grams
MN	meganewton

m/s	meters per second
n	moles
P	pressure, pascals
P_i	pressure, psi
psi	pounds per square inch
R	specific gas constant
R_u	universal gas constant
S	yield strength, in psi
t	wall thickness, in inches
T	absolute temperature, degrees Kelvin
V	volume, meters cubed or liters
ϵ	strain
Υ	strain, shear
σ	stress, psi
ς	ratio of outer diameter to inner diameter

1. INTRODUCTION

Underground coal mine explosions are deadly events and a constant danger that miners and engineers must take great care to mitigate. The long process of coal formation also produces methane gas which, especially in deep lying coal veins, cannot escape until mining disturbs the coal. As long as the methane in the strata is at a higher pressure than the surrounding air and has a way of escape, it will continually vent into the mine workings. This results in potentially explosive conditions when the methane-air mixture reaches the lower end of the explosive limit (Bise, 2003). MSHA (the Mine Safety and Health Administration) requires underground coal mines to either ventilate all areas of the mine to eliminate the risk of a methane explosion or block off abandon areas with seals, which are typically made of concrete or other non-flammable material. The required strength of these seals is dependent on the type of atmosphere behind (inby) the seal, either 50 psi (345 kPa) for inert and monitored atmospheres, 120 psi (827 kPa) for non-inert and unmonitored, and over 120 psi (827 kPa) for special cases. Seals must be designed to resist these pressures for four seconds.

Overpressure is not the only danger of an explosion underground. Due to the confined nature of such explosions a long period of high-velocity and high-pressure air is produced behind the shockwave that travels throughout the mine and is capable of propelling any objects in the mine to a very high velocity. Large or heavy objects such as boulders, roof bolts, and equipment may be picked up and thrown into the mine seals in the event of an explosion. These objects may reach velocities over 700 ft/s (213 m/s) based on a simulated explosion event in which 120 psi (827 kPa) was applied to a representative object for four seconds. (Perry, 2018). Though mine seals are designed to withstand high pressure, they are not intentionally designed to withstand impact. This research sought to design, build, and test a projectile generator capable of propelling a variety of typical mine materials to the required velocity and to examine their effects on 50 psi (345 kPa) and 120 psi (827 kPa) MSHA approved mine seals.

2. BACKGROUND

2.1. LITERATURE REVIEW OF MINE SEALS

Methane is liberated from most coal beds continuously after mining and in many mines will accumulate to, and then pass, the lower and upper explosive limits of the 5% to 16% methane to air ratio. Mines close to the surface, or those with breaches that result in air exchange with current workings or the surface, may never pass the upper limit of the explosive ratio and will remain dangerous throughout the life of the mine (NIOSH, 2007). For increased safety, MSHA considers atmospheres to be inert in a range of less than 3% methane or more than 20% methane. If readings are between 4.5% and 17% methane while oxygen is 10% or greater, immediate action must be taken for atmospheres behind seals rated to less than 120 psi (827 kPa).

Historically, old mine workings have been closed off with a wide variety of walls or seals, including seals built with small individual solid blocks, cinder blocks (concrete masonry units or CMU), poured concrete, reinforced concrete, walls of gob (waste rock), timber, and other materials and methods. The Sago Mine disaster on January 2nd, 2006, and its subsequent investigation led MSHA to develop stricter and more robust requirements for coal mine seals. Prior to that event, mine seals such as those in the Sago Mine were only required to resist 20 psi (138 kPa) based on a 1992 USBM study; and before that there were no standards for pressure rating mine seals (Gates et al., 2007). The seals in the Sago Mine were constructed of concrete and fly ash blocks cemented together, made to be simple, inexpensive to construct, and easy to install. MSHA found, through interviews and an investigation, that the seals were destroyed and the individual blocks hurled into the working areas of the mine during the explosion, which initially occurred on the inby, non-ventilated side of the mine. Ignition was likely due to lightning strikes which caused a spark to jump across metal reinforcement in the mine, igniting the methane-air mixture. Unlike other coal mine explosions, the coal dust still present in the mine was not thought to have contributed to the explosion. MSHA estimated that pressure may have reached over 90 psi (655 kPa) behind the seal when it was destroyed, far more than the 20 psi (138 kPa) the seal was rated for (Gates et al., 2007).

After this event, MSHA decided on the three different pressure standards for mine seals, justification for which is provided in the NIOSH report IC 9500 (NIOSH, 2007). Mine seal manufacturers may submit seal designs for approval to MSHA, which keeps a list of all approved seals (MSHA, 2007). Mine seal designs are rated for different size openings and require different thicknesses of concrete or layers of CMU based on their designed pressure. MSHA requires seals to be designed to withstand 50 psi (345 kPa) for four seconds when the atmosphere behind the seal is kept inert, and 120 psi (827 kPa) for four seconds when the atmosphere is not monitored. Currently, no seals have been approved for over 120 psi (827 kPa) (MSHA, 2007).

Mine seal manufacturers submit designs to MSHA for approval upon completion of the engineering design work. Seal designs are not required to be field tested due to the high cost and difficult logistics of testing; engineers use CAD analysis programs and engineer equations to estimate the pressure resistive properties of their designs. Designs may include monolithic blocks with or without reinforcement, CMU blocks, and gob seals created by heaping up waste rock to form a barrier (MSHA, 2007). Seals tested during this research were monolithic concrete blocks, one with reinforcement (referred to as the reinforced seal) and one without reinforcement (referred to as the plug seal).

2.2. LITERATURE REVIEW OF BLACK POWDER

Testing of construction materials is often carried out in weather impact testing. Impact testing of structures is typically conducted using pneumatic (compressed air) cannons to determine their resistance to extreme weather events. This type of testing is meant to evaluate walls, doors, and windows for their resistance to common items being thrown by tornadoes and hurricanes. Though a pneumatic cannon was initially considered for this research, it was quickly realized that any compressed air cannon can only produce modest projectile energy (960 ft-lbs. or 1.3 kJ) compared to a combustion driven projectile generator (>73,800 ft-lbs. or 100 kJ) (FEMA, 2017). Light gas guns are another form of projectile generator that may have been useful for the project, but they are limited to a small diameter projectile in any practical gun, require a large horizontal footprint, and are very expensive.

With these considerations in mind, standard black powder was chosen as the propellant and a muzzle loading design for the projectile generator in order to produce high velocity projectiles in a safe and cost-effective manner. Black powder is relatively inexpensive, produces enough gas pressure to achieve high velocities, and is not capable of producing so much gas pressure as to require an expensive gun design to safely contain it.

Research on black powder has been conducted since the early to mid-1900's, but the field seems fairly arcane due to black powder's obsolescence as a propellant and mining explosive. World militaries replaced most of their black powder arms before WWI with nitrocellulose base propellants, though more than 10 million pounds were produced during that war for use in fuses and older firearms. It is still used in some military hardware in use today, such as the RPG-7 (Shea, 2012), as a reliable ignition mechanism for other charges (Sasse, 1985). The advent of nitroglycerin dynamite eliminated its use in the mining industry. Black powder's obsolescence occurred before a complete understanding of its characteristics was realized. For all its antiquity as humanity's first low explosive black powder still has yet to be fully understood.

Black powder (or gunpowder) is classified as a propellant or low explosive rather than a high explosive. It does not and cannot detonate under most circumstances. Detonation by definition includes the formation of a shock (pressure) wave moving faster than the speed of sound in the detonating material, black power instead only burns, albeit very quickly, producing hot gasses and solid products but no shock wave. Typical black powder has only three components: carbon (C) or charcoal; sulfur (S), and potassium nitrate (KNO_3). Their ratios are approximately 75% potassium nitrate, 15% charcoal, and 10% sulfur by weight and correspond to the stoichiometric formula (Williams, 1975):

$$2\,KNO_3 + 3\,C + S \rightarrow K_2S + N_2 + 3\,CO_2 \tag{1}$$

in which black powder burns to produce potassium sulfide (K_2S), diatomic nitrogen (N_2) and carbon dioxide (CO_2). However, this ratio was not historically agreed upon; numerous alchemists and later scientists (the famous Roger Bacon among them) proscribed significantly different formulas (Davis, 1941).

As shown by Table 2.1, the correct stoichiometric ratio of components was known by the end of the 18th century. The changing ratios may not necessarily reflect error on the part of those medieval scientists. Different ratios give different pressure profiles which may

Table 2.1 Historical Black Powder Compositions by Weight (Davis, 1941)

	Saltpeter	Charcoal	Sulfur
8th century, Marcus Graecus	66.66	22.22	11.11
8th century, Marcus Graecus	69.22	23.07	7.69
c. 1252, Roger Bacon	37.50	31.25	31.25
1350, Arderne (laboratory recipe)	66.6	22.2	11.1
1560, Whitehorne	50.0	33.3	16.6
1560, Bruxelles studies	75.0	15.62	9.38
1635, British Government contract	75.0	12.5	12.5
1781, Bishop Watson	75.0	15.0	10.0

have been more appropriate for whatever use they put it to at the time. For example, higher sulfur content likely made the low-purity early black powder easier to ignite. It should be noted that although sulfur does contribute to the amount of gasses produced, it is not strictly necessary. Sulfur's main purpose is to decrease the ignition temperature, making for an easier ignition and a faster burn (Davis, 1941). When black powder is exposed to a flame, sulfur's low melting point and combustion temperature act as an intermediate step between the ignition flame and the relatively high ignition temperature of potassium nitrate and carbon (Conkling, 2010) (Rose, Hardt, 1979). Table 2.2 displays sulfur's effect on the maximum reaction temperature as a function of its weight percentage.

As shown in Table 2.2, a powder made with 15% sulfur results in a combustion temperature of 1300° C (2372° F), near the maximum possible black powder combustion temperature of 1350° C (2462° F). Per the ideal gas law, changing gas temperatures result

Table 2.2 Sulfur Content vs Max Temperature (Brown, 1989)

Table 1. The Effect of Increasing Sulphur Content on the Burning Rate of Sulphurless Black Powder.

Sulphur [%]		Burning rate [cm/s]	T_{max} [°C]	Enthalpy of reaction, − H (bomb calorimetry) [kJ/g]	[kJ/mol] (of oxidizer)
0%	h	0.20 ± 0.01	900	2.75 ± 0.03	243.5
	p	0.16 ± 0.01			
5%	h	0.39 ± 0.01	1020	3.35 ± 0.05	278.8
	p	0.38 ± 0.01			
10%	h	0.71 ± 0.01	1350	3.21 ± 0.02	252.2
	p	0.56 ± 0.01			
15%	h	0.94 ± 0.01	1300	2.92 ± 0.02	217.2
	p	0.81 ± 0.01			
20%	h	0.95 ± 0.01	1240	2.80 ± 0.04	201.0
	p	0.83 ± 0.01			
30%	h	0.79 ± 0.01	1140	2.59 ± 0.05	166.8
	p	0.77 ± 0.01			

h = hand-packed, p = pressed.

in linear changes in pressure. For safety's sake, the higher temperature of 1350° C will be used as the reaction temperature in this thesis when calculating gas pressure. For contrast, Figure 2.1 displays the combustion temperature curve of various mixtures of potassium nitrate and charcoal and indicates that the removal of sulfur significantly reduces temperature (red lines added to the graph intersect at max temperature reached for sulfur-less black powder with 15% charcoal, about 1100° C on the graph).

The best type of charcoal, what tree it was produced from, and its exact chemical composition were points widely debated and examined in the medieval period and early industrial age. In fact, it is still relevant today and its effects on gas production will be discussed further in the thesis. Different tree species produce varying compositions of charcoal and may be suitable for producing different effects in black powder and other fireworks compositions. Debate still exists over how volatiles in the charcoal affect burn rates; some researchers posit that the amount of volatiles in the charcoal has a great effect on its reactivity and burn rate (Rose, Hardt, 1979). As was typical of early scientific

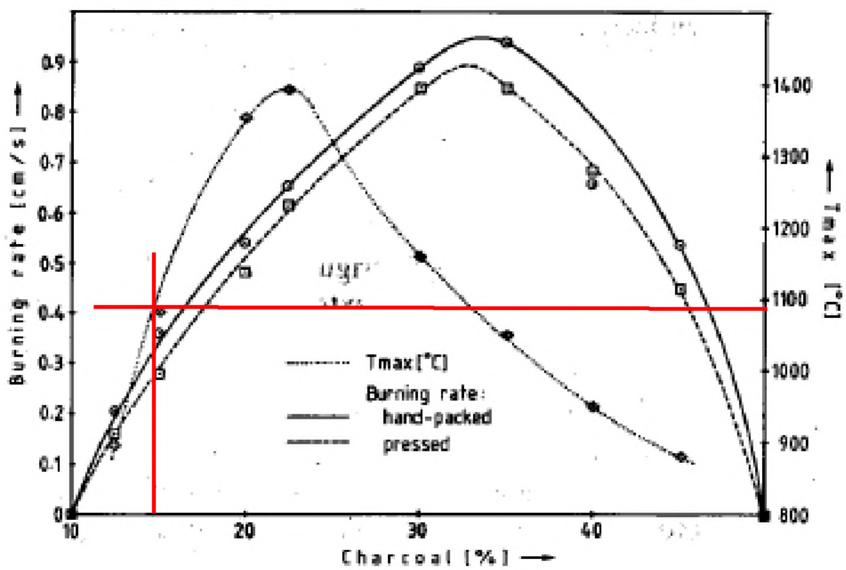


Figure 2.1 Temperature Curve of Sulphur-less Black Powder (Brown, 1989)

research, the manufacture of black powder was an art developed by trial, error, and deadly experience with many milling operations destroying themselves (Lancaster, 2006). The only present-day company in the United States still producing black powder keeps their exact recipe and methods a trade-secret, as the author discovered when attempting to gather more information on the subject.

Also relevant to an understanding of black powder is an examination of its reaction characteristics due to geometry. During deflagration, ignition is generally thought to propagate from one grain to another by means of the molten spray of potassium salts in grained powders. Propagation is slowed by sulfur and a higher sulfur content will slow down the reaction even though it increases sensitivity. Grain size also greatly affects the propagation (flame spread) rate. Large grain powders may have propagation rates as low as 560 ft/s (171 m/s) and small grain powders as high as 2130 ft/s (650 m/s) (Brown, 1989). This is also evident from the use of powder in the muzzle loading firearms: small grain powders are used in smaller bore weapons firing lighter weight projectiles where a fast pressure increase is needed, large grain powders are used for the opposite. Cannon grade powder, the largest grain commercially available for firearms, was the grade used in this study. Small grain powders may have a specific gravity of up to 1.6. The cannon grade powder used in this study had a measured specific gravity of 1. The lower density of larger powders results in a lower energy density per unit of volume.

Reaction products are also of special concern when evaluating black powder's performance. Few chemical mixtures are perfectly homogeneous; each time one substance is mixed with another slight difference in mixing will occur. Gun powder companies have historically found this to be true even up to the present day (Sasse, 1985). An empirical

Potassium nitrate			74.430
Potassium sulfate			0.133
Sulfur			10.093
Charcoal	Carbon	12.398	14.286
	Hydrogen	0.401	
	Oxygen	1.272	
	Ash	0.215	
Moisture			1.058

Their mean results from the analysis of the gaseous products (percentage by volume) and of the solid products (percentage by weight) are shown in the following tables.

Carbon dioxide	49.29	Potassium carbonate	61.03
Carbon monoxide	12.47	Potassium sulfate	15.10
Nitrogen	32.91	Potassium sulfide	14.45
Hydrogen sulfide	2.65	Potassium thiocyanate	0.22
Methane	0.43	Potassium nitrate	0.27
Hydrogen	2.19	Ammonium carbonate	0.08
		Sulfur	8.74
		Carbon	0.08

Figure 2.2 Noble and Abel Experimental Results (Davis, 1941)

measure would be more accurate, and Davis provides an analysis (Figure 2.2) of the reaction products done by Sir Andrew Noble and Frederick Abel in the late 19th century.

Results not shown included the percentage of solids as 55.91%, 42.98% gas, and 1.11% water; and the volume at 274.2 cc at 0° C under a pressure of 760 mm mercury (units of temperature not stated, author assumes to be Celsius). As shown in Figure 2.2 there are a large and varied number of reaction products, and charcoal is composed of more elements than carbon. In modern powders charcoal is approximated as roughly C_9H_4O, a rounded empirical average used by other researchers for the particular black powder used in this study (Sasse, 1984).

2.3. LITERATURE REVIEW OF GUN DESIGN

Gun design has a long and fascinating history. Early firearms were simple iron tubes, open at one end and having a touch-hole drilled or forged into the other. Gunpowder was poured down the tube, a rounded rock or steel ball (or several) driven down after it, and a burning corded match touched to the touch-hole. Black powder was likely first discovered in China with the oldest known writings dated to AD 808 (Lancaster, 2006), and it seems logical that the invention of guns would follow soon after. Firearms were introduced and/or developed in medieval Europe in the 14th century, and by the late 15th they were sailing with Columbus to the Americas (Pegler, 2009).

The so called "cottage industry", as most manufacturing is termed prior to the industrial revolution, did not produce consistent parts because the technology of the time simply wasn't accurate enough and manufacturers were fairly dispersed. Firearms were each individually made and parts fitted by hand. In order to ease loading, the barrel or "bore" was often oversized for the projectile to be fired which allowed gas to escape past the projectile or "blow-by" when fired, reducing accuracy and velocity.

Development did progress rather quickly by the 1600s, essentially modern spring-loaded triggers appeared along with the progression to a flintlock mechanism as a means of initiation, rather than burning match cords. In perhaps a portent of future American culture English colonists themselves were quite well armed in 1609, having according to John Smith "24 peeces [*sic*] of ordinances, 300 muskets, snaphaunces and fire-lockes [*sic*], shot powder and match" (Pegler, 2009). Flintlocks gave way to the percussion cap in the early to mid-1800's, just in time to become well-establish before the American Civil War. Percussion caps used mercury fulminate, an explosive and highly sensitive compound, to set off the main charge of powder in the barrel.

Many modern firearm designs and especially ideas stem from this period; and as is obvious from the much-abbreviated history above, much of the research was trial and error, intuition and ideas. Firearms and traditional "academic" engineering were not always closely related. John Browning, a gun designer who is to firearms what Einstein is to physics, had no formal higher education. Today, as with most engineering disciplines, firearms are designed by engineering teams using CAD software and advanced mechanical engineering degrees. Some of that knowledge will be discussed in the paragraphs below as it pertains to the design of a projectile generator for this thesis.

From an engineering standpoint, guns can be conceptualized essentially as pressure vessels containing a piston which is free to move in one direction. Barlow's formula, which is a common formula used in civil engineering (49 CFR 192.105) for determining the design pressure of steel pipe, has been simplified into the formula:

$$P_i = \frac{2 * S * t}{D} \tag{2}$$

and gives the yield and burst pressure based on the yield and tensile strength of the material, where P_i is maximum internal pressure (psi), S is the yield or tensile strength (psi), t is the wall thickness (inches), and D is the outside diameter of the pipe (inches). Barlow's formula, while widely used for pressure estimation in industry, was not intended to be used in gun barrels but is accurate for thin walled pipes.

The wall ratio equation is a more conservative equation and more accurate for thick walled tubes. It is a combination of von Mises failure criterion and Lame's formulas; specifically used for gun barrels (Carlucci, Jacobson, 2013):

$$P_i = \frac{\varsigma^2 - 1}{\sqrt{3 * \varsigma^4 + 1}} * S \tag{3}$$

and gives the yield and burst pressure. In this formula, P_i is internal pressure (psi), ς is the ratio of the outer diameter to the inner diameter (OD/ID) (unitless), and S is the yield or tensile strength (psi).

Pressure produced by the firing of a cartridge must also be known in order to design safe firearms. The exact calculations involved can be incredibly complex and were beyond the scope of this thesis (Carlucci, Jacobson, 2013). However, a few equations with some assumptions can greatly simplify the process.

The ideal gas law can be used to find the pressure of any ideal gas at close to standard temperatures and pressures and is stated in two forms (Elger et al., 2013):

$$P = \left(\frac{n * R_u * T}{V}\right) \tag{4}$$

$$P = \left(\frac{m * R * T}{V}\right) \tag{5}$$

where

$$R = \frac{R_u}{M} \tag{6}$$

and P is pressure (Pascals), n is the amount of gas in moles, R is the specific gas constant, R_u is the universal gas constant, T is the absolute temperature (degrees Kelvin), and V is the volume in meters cubed. This equation is not particularly accurate as temperatures and pressures increase (Cooper, 2010).

Engineers in the field of ballistics often use an equation developed by the aforementioned Noble and Abel, called the Noble-Abel equation of state:

$$P = \frac{m_g * R * T}{V - m_g * b} \tag{7}$$

where the variables are the same as Equation (4) with the addition of m_g representing the mass of the gas and b representing co-volume (an empirical value unique to each gas), which is often close to $1/1000^{th}$ of the specific volume (Cooper, 2010) (Carlucci, Jacobson, 2013). Cooper gives a simplified version of the Noble-Abel equation of state with:

$$P = \frac{0.0821 * n * T}{V - 0.025 * n} \tag{8}$$

where n is moles, T is temperature Kelvin, and V is volume in liters. This equation has been tailored to fit empirical data from common propellant gasses (Cooper, 2010).

Projectile design and development have progressed greatly since the first person loaded a rock into a tube filled with gunpowder; though the basic ideas of all projectile weapons remain the same. One particular development, used extensively for this project, is the sabot (a French word for a type of shoe, pronounced say-bow) (Carlucci, Jacobson, 2013). Sabots are here defined as any device that allows the firing of projectiles smaller than the bore diameter of a gun. Because sabot materials are generally much lighter weight and/or have a lower mass (they are not intended to cause damage or ever reach the target), the lighter-weight sabot-enclosed projectile can achieve much higher velocity than typical for a gun of that caliber.

A pervasive example of a modern-day sabot is the shotgun sabot; a piece of molded plastic which holds a slug smaller than the bore diameter of the shotgun, allowing hunters to kill deer and other game animals from a far greater range than when using traditional buckshot. In fact, the typical shotgun wad holding shot can be classified as a cup-type sabot, allowing a shotgun to fire many sub-caliber projectiles at once. Many militaries also use sabots in various guns (such as the 120mm cannon on the M1 Abrams tank) in order to fire extremely high velocity projectiles that can defeat heavily armored targets. This project also used sabots for the same purpose; Figure 2.3 is a drawing of one of the steel projectiles used in this project with a wood and foam sabot.

Sabot material selection is a critical part of sabot design. Having no effect on the target, sabot weight is entirely parasitic (Carlucci, Jacobson, 2013), and therefore engineers attempt to design sabots to be as light as possible. However, as with many engineering

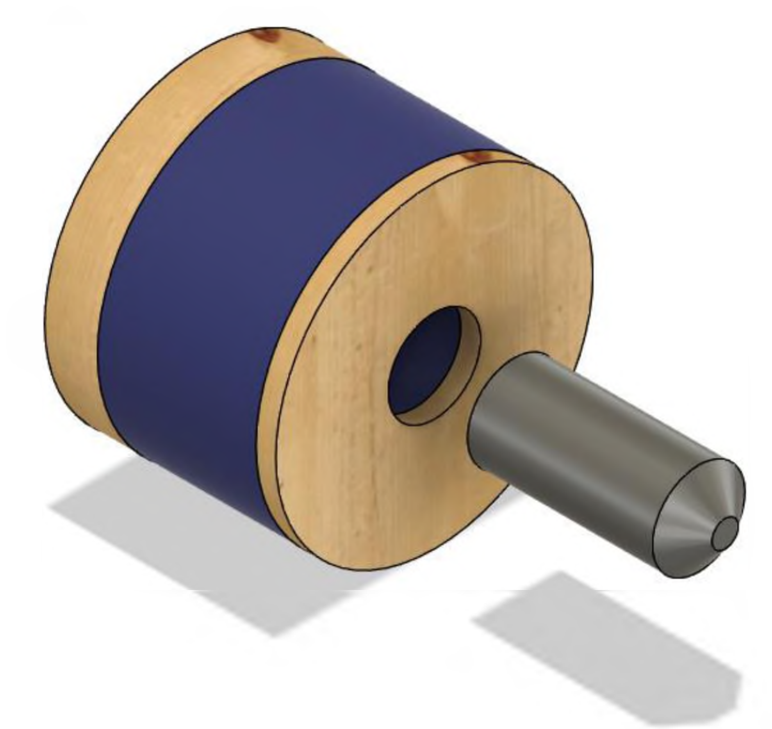


Figure 2.3 Steel Projectile with Sabot

challenges, it presents a dilemma of choice that forces the engineer to decide, based on a multidimensional scale involving weight, manufacturing costs, time, function, and safety, which attributes or combination thereof best meet the needs of the user. As an aside: the US DOD, being somewhat well funded, could afford to spend about $13,370 USD per round of M829A4 fin-stabilized discarding-sabot for the main gun of the M1 Abrams series tank (Exhibit P-40, 2017). This project, being well but somewhat less funded than the DOD, still occasionally had unit costs of well over $100 USD per shot using the some of the cheapest materials and labor available.

3. DESIGN AND CALCULATIONS

3.1. BARREL DESIGN

Researchers decided on a muzzle loading design for the projectile generator due to the simplicity of use and manufacture. Breech loading designs, as typical of most modern firearms and artillery, were considered but rejected. In order to fire a wide variety of projectiles, the projectile generator needed a large bore (the inside diameter of the barrel), of approximately 12 inches (30.5 cm). Locked breech guns, such as modern artillery, require extensive machining of the barrel and many precisely machined and expensive parts to function. This in turn would result in a very large and heavy device that would not be able to be machined on any of the equipment available at Missouri S&T or in the local area. Therefore, a simple steel tube, plugged at one end and loaded from the muzzle, was deemed the best choice as manufacturing and fitting a breech plug was within the capabilities of S&T machine shops and personnel.

Finding a length of steel tube large enough to become the barrel proved difficult. Eventually a 12-inch (30.5 cm) inner diameter, 20-inch (51 cm) outer diameter seamless 4140 steel pipe was found in the US; a cutoff of a piece imported from Germany. This piece became the barrel of the projectile generator, weighing roughly 5,800 pounds (2630 kg) and having a length of 8.5 feet (2.6 m).

Before the proposed barrel could be purchased, it first had to be evaluated to determine if it could safely contain the pressure of firing multiple shots without yielding or failing. To ensure safety and simplify the calculations the barrel was evaluated in the worst-case scenario for a gun: a completely obstructed bore under pressure from the largest charge of powder. The barrel pressure calculations proceeded with the assumption that the barrel would have to withstand the residual pressure of three pounds of cannon grade black powder burning completely and only expanding into the volume of the "chamber" behind the projectile.

According to the provided mill report (Appendix A), the barrel steel has a yield tensile strength of 61,000 psi (421 MPa) and an ultimate tensile strength of 112,000 psi (772 MPa). Both the internal yield pressure and ultimate burst pressure of the pipe were calculated and compared to the maximum pressure the propellant can produce. Internal yield pressure and ultimate burst pressure were both calculated using two different equations for redundancy: Barlow's formula (2), and the wall ratio Equation (3). Barlow's formula:

$$P_i = \frac{2 * 61{,}000 \text{ psi} * 4 \text{ in}}{20 \text{ in}} \tag{9}$$

$$P_i = \frac{2 * 112{,}000 \text{ psi} * 4 \text{ in}}{20 \text{ in}} \tag{10}$$

Calculations with this equation gave a yield pressure of about 24,400 psi (168 MPa) and a burst pressure of about 45,000 psi (310 MPa). Applying the wall ratio equation (Carlucci, Jacobson, 2013):

$$P_i = \frac{\left(\frac{20}{12}\right)^2 - 1}{\sqrt{3 * \left(\frac{20}{12}\right)^4 + 1}} * 61{,}000 \tag{11}$$

$$P_i = \frac{\left(\frac{20}{12}\right)^2 - 1}{\sqrt{3 * \left(\frac{20}{12}\right)^4 + 1}} * 112{,}000 \tag{12}$$

Calculations with Equations 11 and 12 gave a yield pressure of about 22,000 psi (152 MPa) and a burst pressure of about 41,000 psi (283 MPa). Both formulas give somewhat similar results. However, Barlow's formula is only intended for thin walled pipes under relatively low pressures. The wall ratio equation given by Carlucci and Jacobson is specifically for gun barrels and is a derivation of the von Mises criterion for stress (Carlucci, Jacobson, 2013). In Equations 11 and 12, failure is said to occur if the barrel pressure ever exceeds the yield strength of the barrel material. As the wall ratio equation more accurately represents the actual stresses on a thick-walled barrel and gives a lower maximum pressure than Barlow's formula, 22,000 psi was used as the maximum pressure that the barrel should be exposed to.

The next design step after determining the max barrel pressure was to determine how much pressure the cannon grade black powder produces per unit of weight and what chamber volume would keep that pressure below the yield strength of the barrel. Every combustible or explosive substance has a maximum static pressure it is capable of producing (assuming it is contained in a space equal to its volume) based on the amount (moles) of gas produced per unit of volume for a given initial density and constant reaction temperature. Maximum static pressure can be found with a combination of stoichiometry, the modified Nobel-Able equation of state given by Cooper (2010), and several assumptions. The simplest equation commonly given for the combustion of black powder is given in Equation (1). However, that reaction is perhaps too simple. Experimental data has shown that the products of black powder combustion are varied, as shown in Figure 2.2. The author could find no stoichiometric formula in previously published research that accurately represented the reaction, likely because black powder was largely supplanted by smokeless powder in the early 20th century (Sasse, 1985) and only occasional research has been published since.

Noble and Abel's analysis was used to develop a more accurate representation of the reaction, approximating charcoal as C_9H_4O. Their analysis was simplified somewhat to ease calculations; water was eliminated from the reactants which changed the solid to gas mass ratios to about 56.4% and 43.6% respectively. Noble and Abel left it as percent volume in their report; the gas must be converted to a moles per unit of weight to be useful in pressure calculations. Table 3.1 shows this process.

The moles per kilogram of black powder (moles/kg in Table 3.1) were found by using values provided by Noble and Abel and a re-arranging of Equation 7, solving for the moles of gas n, and accounting for the volume percentage of each individual gas:

$$n_{kg} = \frac{P * (V * \%v)}{T * (0.0821 + \frac{0.025}{T})} * 1000 \tag{13}$$

where the units are the same as Equation 7, with the addition of the percentage volume (% v) and multiplying by 1,000 to arrive at the moles of gas per kilogram of black powder (n_{kg}).

Table 3.1 Noble Abel Analysis

	KNO3	C9H5O1	S	-->	K2CO3	K2SO4	K2S	S	CO2	CO	N2	H2	H2S	CH4
grams/mol:	101.1	129.1	32.1		138.2	174.3	110.3	32.1	44.0	28.0	28.0	2.0	34.0	16.0
% mass:	74.8%	14.7%	10.5%		61.2%	15.3%	14.5%	9.0%						
% vol:									49%	12%	33%	1%	3%	4%
moles/kg:									5.43	1.37	3.62	0.12	0.29	0.47
% mass:	74.8%	14.7%	10.5%		35%	9%	8%	5%	23.89%	3.85%	10.14%	0.02%	0.99%	0.76%
												Σ Moles Gas/kg:		11.31

For example, solving for the moles/kg of CO_2:

$$n_{kg} = \frac{1 * (0.2472 * 0.49)}{273.15°K * (0.0821 + \frac{0.025}{273.15°K})} * 1000 \tag{14}$$

yields 5.43 moles of CO_2 per kg of black powder. Using this equation in Table 3.1, the total gas production of black powder was found to be 11.31 moles/kg (25 moles/lb.). Some slight errors were introduced using this method. Several of the trace quantities of solid products were eliminated from the calculations, and moisture content was also ignored. This, combined with the derived nature of the calculations, resulted in about a 3% loss of mass from the original given weight. The error is small and was deemed to be within an acceptable margin. Finally, the maximum pressure produced in the worst-case scenario for the projectile generator could be computed. Using Cooper's equation modified equation of state for pressure (Cooper, 2010):

$$P = \frac{0.0821 * 15.4 \text{ mol} * 1623 °K}{3.71 \text{ liters} - 0.025 * 15.4 \text{ mol}} \tag{15}$$

where 3 lb. (1.4 kg) of black powder produces 15.4 moles of gas in the projectile generator's 226 in^3 (3.71-liter) chamber, the equation yields a pressure of 617 atm or 9,074 psi. The projectile generator has a calculated yield pressure of 22,000 psi (152 MPa), therefore the yield factor of safety in the worst-case scenario is 2.4. Using the burst pressure of 41,000 psi (282 MPa), the projectile generator has an absolute factor of safety of 4.5. In this manner the projectile generator's barrel was proven safe for use, provided a ratio of 3 lb. (1.4 kg) of black powder per 226 in^3 chamber volume (or 2 inch or 5 cm chamber length) is not exceeded while using a disintegrating powder container such as insulation foam. The calculations provided are completely empirical and not quite stoichiometrically balanced; to further understand the problem the author attempted to find a balanced stoichiometric equation, starting with the reaction:

$$KNO_3 + C_9H_5O + S \rightarrow \tag{16}$$
$$K_2CO_3 + K_2SO_4 + K_2S + CO_2 + CO + H_2O + N_2 + H_2 + S$$

H_2O was included in this equation because it is very likely to form during combustion (Cooper, 2010). Again, several minute products of Noble and Abel's analysis have been eliminated for simplicity, such as ammonium carbonate and pure carbon. A spreadsheet was used to find the coefficients of the reaction and then determine the mass percentages of the products as shown in Table 3.2.

Balancing was accomplished by the "guess and check" method because the equation can't be solved algebraically, or rather, it has near infinite solutions. This solution gave the formula shown in Equation 10:

$$80\ KNO_3 + 12.5\ C_9H_5O + 33.7\ S \rightarrow \tag{17}$$
$$26\ K_2CO_3 + 6\ K_2SO_4 + 8\ K_2S + 59\ CO_2 + 27.5\ CO + 5\ H_2O + 40\ N_2 + 26.25\ H_2 + 19.7\ S$$

Normal convention for chemical equations calls for the elimination of fractions and decimals, but to do so in this equation would result in very large and irreducible coefficients. As stated, the number of possible reactions is nearly mathematically infinite;

the goal of this equation was to arrive at a balanced equation that was a close as possible to the reaction in Figure 1.4; a balance more representative of the reaction is likely possible.

The mass ratio of solids to gasses (~57% solids to ~43% gasses) is within one percent of the ratio found by Noble and Abel (Davis, 1941) although the ratios of different gaseous products differ by up to 14%. However, Cooper's Equation 7 is already accounting for common reaction products in the R value and co-volume adjustment, therefore the actual ratios of the different gasses should have little effect on pressure calculation. Using the mass ratios found in Table 3.2, Table 3.3 calculates the moles of gas produced per kilogram of black powder.

Table 3.3 reports 14.6 moles/kg (6.6 moles/lb.) of gas, as opposed to the strictly empirical method in Table 3.1 which found 11.3 moles/kg (5.1 moles/lb.). As an additional comparison, the "standard" black powder reaction equation (Equation 1) yields about 14.8 moles/kg (6.7 moles/lb.). Equation 7 and Table 3.3, with the same volume and temperature as before and 3 lb. (1.4 kg) of black powder, yields 19.9 moles of gas and a pressure of 12,154 psi (84 MPa) compared to 9,074 psi (63 MPa) from Table 3.1, a 34% increase in pressure.

The stoichiometrically balanced equation has infinite solutions and does not seem to represent the actual black powder reaction as closely as the strictly empirical calculation. The simplistic reaction of Equation 1 disregards most of the known products of black powder combustion. For the purposes of this thesis, the Noble-Abel analysis used in Table 3.1 is assumed to be the most accurate representation of the moles of gas produced.

Table 3.2 Stoichiometric Balance of a Complex Black Powder Reaction

	KNO3	C9H5O1	S	-->	K2CO3	K2SO4	K2S	CO2	CO	H2O	N2	H2	S
grams/mol:	101.1	129.1	32.1		138.2	174.3	110.3	44.0	28.0	18.0	28.0	2.0	32.1
moles ratio:	80	12.5	33.7		26	6	8	59	27.5	5	40	26.25	19.70
mass (kg):	8.09	1.61	1.08	0.00	3.59	1.05	0.88	2.60	0.77	0.09	1.12	0.05	0.63
mass %:	75.0%	15.0%	10.0%		33.3%	9.7%	8.2%	24.1%	7.1%	0.8%	10.4%	0.5%	5.9%

Table 3.3 Stoichiometric Balance Pressure Calculation

	KNO3	C9H5O1	S	-->	K2CO3	K2SO4	K2S	CO2	CO	H2O	N2	H2	S
mass %:	75.0%	15.0%	10.0%		33%	10%	8%	24%	7%	0.83%	10%	0.49%	6%
mass (g):	748.5	149.3	100.0		332.6	96.8	81.6	240.3	71.3	8.3	103.7	4.9	58.5
g/mol:	101.1	129.1	32.1		138.2	174.3	110.3	44.0	28.0	18.0	28.0	2.0	32.1
moles:	7.4	1.2	3.1		2.4	0.6	0.7	5.5	2.5	0.5	3.7	2.4	1.8
State:	S	S	S		S	S	S	G	G	G	G	G	S
											Σ Moles Gas/kg:		14.60

3.2. BREECH PLUG DESIGN

Most modern artillery is breech loading, meaning it loads from the rear of the gun, as opposed to nearly all firearms prior to the 19th century which loaded from the muzzle or front end. Breech loading has many advantages: fast loading due to not needing to ram the propellant and projectile down the barrel, access to the rear end of the barrel should errors occur in firing, and easier cleaning. While a breech loading projectile generator would have been preferable due to these advantages, the required design time and complexity of machining the heavy barrel were unfortunately beyond the scope of the project. Consequently, a relatively simple breech plug and plate to seal the bore was designed. Researchers desired a simple design with a high factor of safety to plug the breech end of the projectile generator, but in a way such that the plug could be removed if a projectile became fixed in the barrel due to errors in loading. The largest possible force on the breech face can be calculated by taking the max static pressure of 9,074 psi (63 MPa) and multiplying it by the area of the bore (113 in^2, 713 cm^2), resulting in a force of 1,026 kips (4.6 MN). Researchers chose a 12-inch diameter by 12-inch-long (30.5 cm) cast steel cylinder as a breech plug to be pinned in place with pull-out steel dowels and then capped with a steel breech plate. Removing the breech plate and the plug in particular would be difficult but not impossible if the need arose. Figure 3.1 is a CAD drawing of the proposed breech plug and plate design. Forty-four pins and bolts (twenty-two each) were used to hold the breech plug and plate in place. The dowel pins were arranged in a series of two rows of eleven pins each, with holes drilled through the barrel and two inches into the breech plug and the four-inch-long dowel pins inserted through the barrel wall and into the plug.

Having a shear strength of 102,000 lbs. (0.45 MN), the pull-out dowel pins have a combined strength of 2,244,000 lbs. (10 MN) An additional 22 holes were drilled around the circumference of the breech plate and two inches into the rear of the projectile generator parallel to its axis. Researchers tapped these holes for the threads on the bolts and attached the breech plate to the rear of the projectile generator with the bolts. With a tensile strength of 117,810 lbs. (0.52 MN), twenty-two bolts have a combined strength of 2,591,820 lbs. (11.5 MN) In total, the breech design can support a force of 4,835,820 lbs. (21.5 MN) This design gave the projectile generator breech a factor of safety of 4.7. The barrel and breech plug were shown to be safe using these methods, and the actual factor of safety is much higher as the barrel is not entirely sealed due to the touch hole.

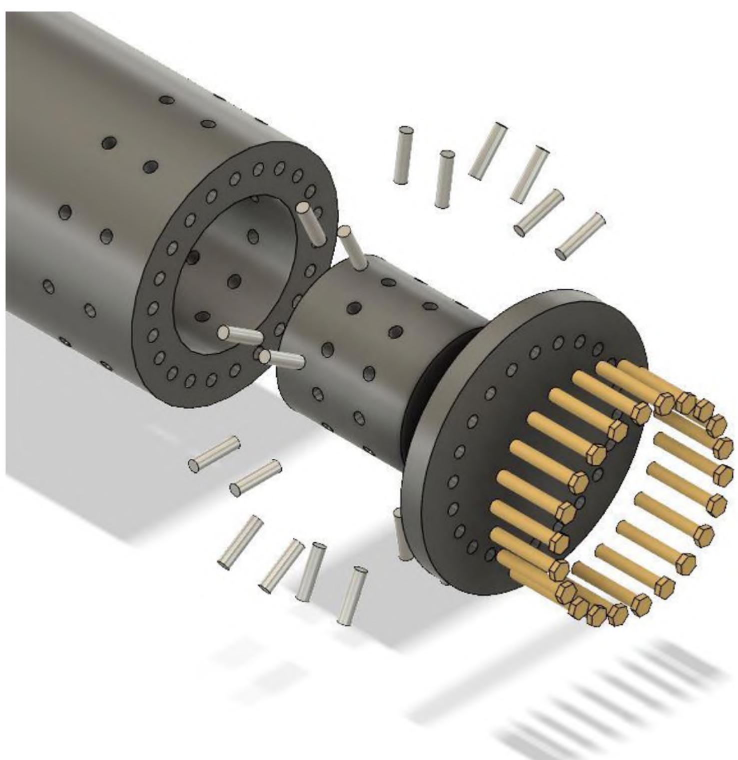


Figure 3.1 Breech Plug and Plate Design

3.3. FRAME DESIGN

The frame design proceeded with two main design considerations: the frame must be simple enough to construct out of locally available materials with the tools available at S&T, and it must be able to withstand the forces of gravity and recoil. With these considerations in mind, researchers chose to build the frame from 1/8-inch thick wall, two inches by two-inch square tubing or, to use industry terms, Hollow Structural Sections or HHS. Figure 3.2 displays the initial CAD drawings of the design:

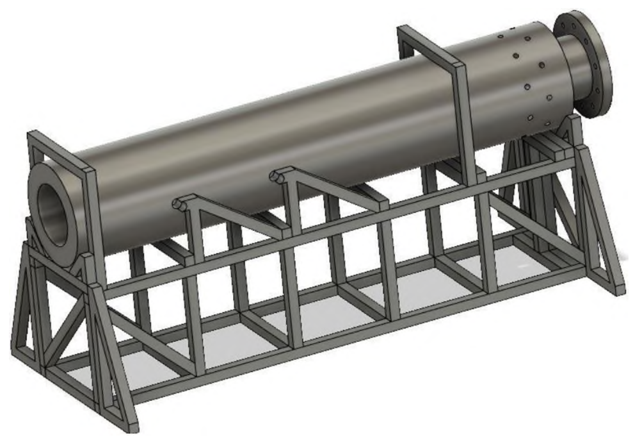


Figure 3.2 Initial Frame Design

A total of 8 HSS were placed as horizontal supports to hold the weight of the projectile generator, each having a bending strength of 4.9 kips (21.8 kN) over the two-foot-wide frame, giving the frame the ability to support a downward force of 39.4 kips (175 kN). The force of gravity due to the projectile generator's weight, breech plug and plate included, is equal to about 7 kips (31 kN), resulting in a factor of safety of 5.6. The frame is slightly overbuilt in this regard. After the initial construction of the frame, the author had several lengths of HSS left over and added them vertically in between the upright sections (supporting the midpoint of the horizontal sections), further increasing strength. Those sections are not shown in Figure 3.2 but are visible in pictures of the completed projectile generator.

Recoil is arrested in this design by four recoil lugs and frames, two on each side, which are the triangular HHS attached to the centerline of the barrel in Figure 3.2. These

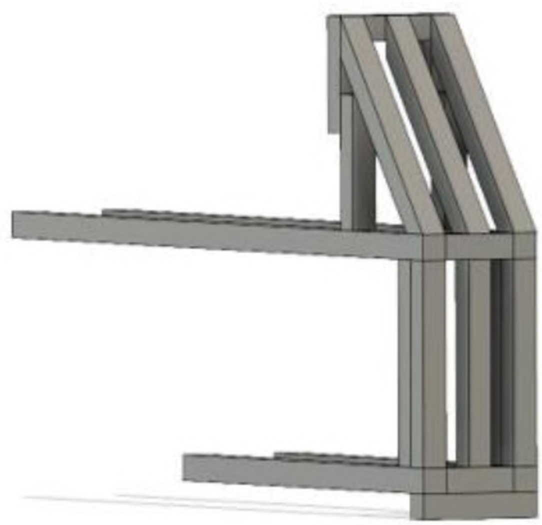


Figure 3.3 Recoil Frame

were each made by welding a steel cube to the triangular HSS along the centerline of the projectile generator between the horizontal supports, drilling a one-inch diameter hole through the two-inch cube and drilling an additional two inches into the barrel. A steel pull-out dowel pin was then inserted through the cube and into the hole in the barrel. Further recoil support was provided by another series of HSS at the rear of the projectile generator as shown in Figure 3.3:

This frame was welded to the rear of the projectile generator butting up against the breech plate. Using the table values for HHS strength (Steel Construction Manual, 2005) and the geometry of the frame, the author calculated the combined strength of the recoil arresting components to be 16.5 kips (73 kN). Per Newton's third law, the recoil force can be computed as the force caused by a 30 lb. (14 kg) projectile accelerated to 700 ft/s (213 m/s), the maximum weight projectile at the highest expected velocity to be fired from the projectile generator. Table 3.4 shows the calculation:

Table 3.4 Recoil Force Calculation

Recoil Force		
Prjt Weight:	30	lbs
Prjt Mass:	0.93243	slug
Velocity:	700	ft/s
Barrel Length:	7.5	ft
Avg Vel:	350	ft/s
Time in Barrel:	0.021	s
Acc:	32667	ft/s^2
Force:	**30459.38**	**lbs**

The recoil calculation is greatly simplified from real ballistic acceleration, assuming constant acceleration through the barrel, but does give a good estimate of the average force experience by the projectile and, inversely, the projectile generator during firing. Table 3.4 gives the average force as 30,500 lbs. or 30.5 kips (136 kN). If the projectile generator were fixed to the ground and unmovable, the design would fail because the recoil frame is only capable of resisting a force of 16.5 kips (73 kN) (FS ~ 0.54). However, as will be shown in the next section, the projectile generator was placed on a surface of clean crushed gravel. Therefore, the static friction force between the gravel and the bottom of the steel frame is the actual maximum force that the frame needs to resist, the rest of the force is translated into movement of the projectile generator rearward (the kinetic friction force, lower than the static, arresting the projectile generator's momentum once it begins to move). Using 0.4 as the coefficient of friction between clean gravel and steel (Fine Software, 2020) and a downforce of 7 kips (31 kN), the author calculated the force of friction to be 2.8 kips (12 kN), or about 1/6th of the resistive force of the recoil frame (FS ~ 6). However, if the projectile generator would sink into the gravel slightly (which it did), the static friction force would be significantly higher and very difficult to predict using engineering calculations. The correct answer is likely in between these two extremes, and projectile generator projectiles only ever achieved about half of the designed muzzle energy, thus keeping the recoil force well below the strength of the recoil frame in any case.

4. CONSTRUCTION, PROJECTILES, AND SITE PREPERATION

The Missouri S&T experimental mine provided a safe location to conduct testing on the seals. Researchers selected a site and cleared it of brush and trees; a gravel layer was laid down to create a firm base for the seals and projectile generator. A concrete pad was poured to create a base for the seals, and a team from Strata Mine Services poured the seals.

Figure 4.1 Mine Seals

Figure 4.1 is a picture of the seals during disassembly of the framework after they had set up for a month. The seal on the left is the unreinforced seal rated to 50 psi (344 kPa), and the seal on the right is the reinforced seal rated to 120 psi (827 kPa) and having two grids of rebar for reinforcement.

Construction of the projectile generator was carried out at S&T's Rock Mechanics building. Researchers assembled and welded the frame, shown in Figure 4.2, and added several features not included in the initial design. Outriggers were attached to provide stability during firing, and angled roll-stops (seen on the top right portion of the frame) were added for increased safety.

Figure 4.2 Frame During Construction

Other parts for the projectile generator were manufactured concurrently. The breech plug is shown in Figure 4.3, just after machining, and in Figure 4.4 it has been fitted to the projectile generator and spot-welded in place. Figure 4.4 also shows the drilling and taping of the holes for the bolts and pins that hold the plug and plate in place. Figure 4.5 is a picture of the completed projectile generator barrel, and Figure 4.6 shows the complete projectile generator, unpainted. Also visible in Figure 4.6 are four additional steel straps, which were heated and hammered into place around the barrel then welded to the frame. These straps ensured that the barrel could not break loose from the frame during transport or firing. Finally, the projectile generator was placed in position at the test site (Figure 4.7).

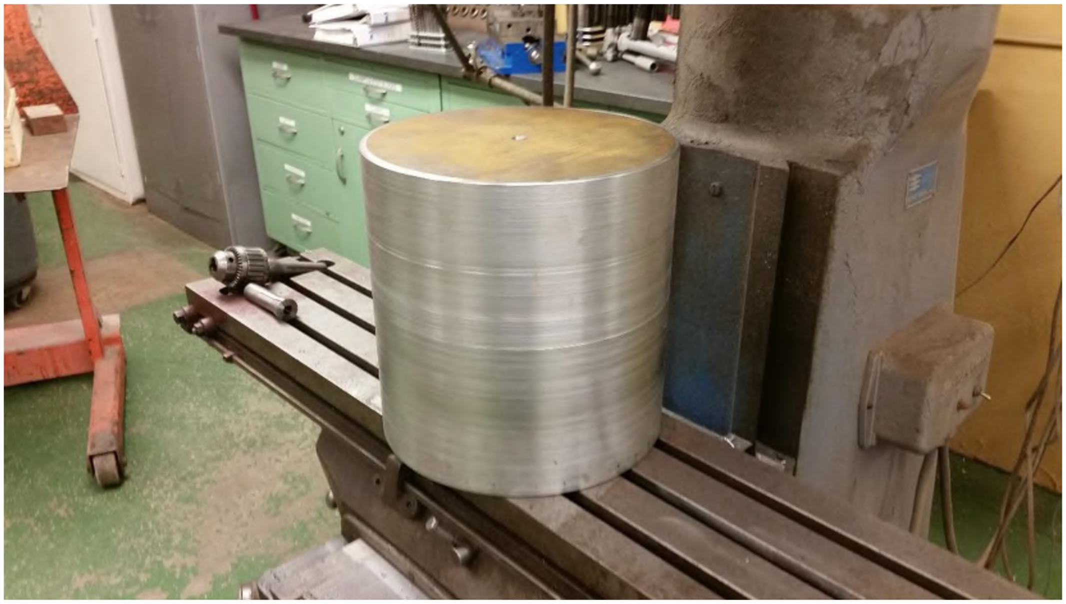


Figure 4.3 Breech Plug

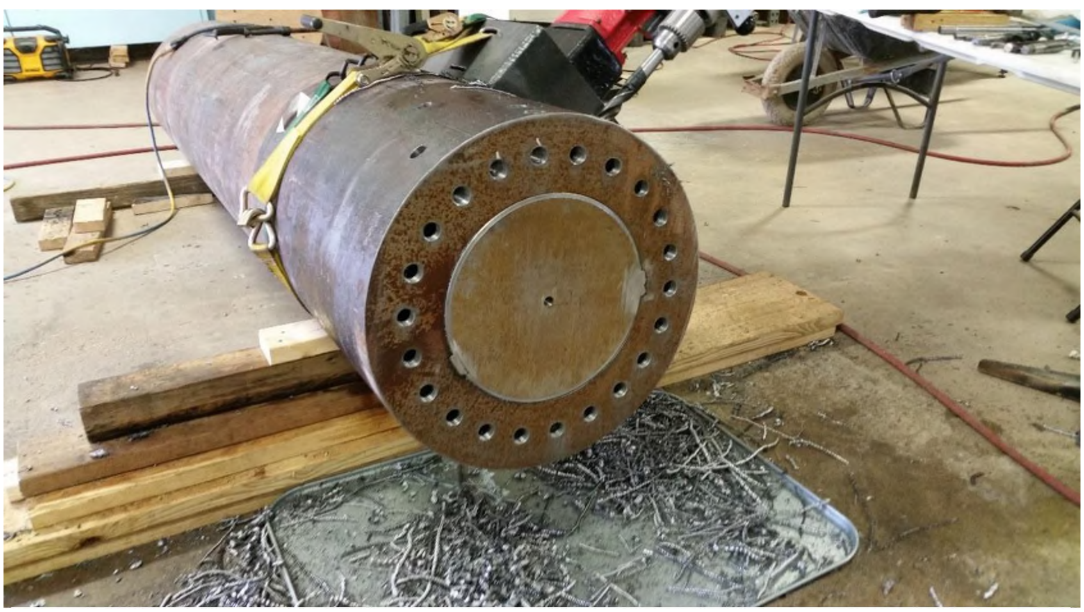


Figure 4.4 Drilling

The projectile generator was specifically designed to fire almost any projectile that fits within a 12-inch diameter circle. To that end, researchers decided on a design very similar to a shotgun wad system in which a wooden “wad” or disk is used to seal the bore

Figure 4.5 Completed Barrel

Figure 4.6 Completed Projectile Generator

during firing and propel projectiles down the barrel. Wads were originally disks cut off from a wood utility pole and lathe turned to the bore diameter, but they were often too cracked along their axis to create an effective gas seal and weighed up to 10 lbs. (4.5 kg)

Figure 4.7 Projectile Generator at Test Site

Figure 4.8 Concrete Projectile with Wad and Powder

by themselves. Researchers then used a water jet cutting CNC machine and a CNC router to cut plywood disks which were stacked together and glued to create the wad. This method worked well, obturating the bore and holding together during firing; it also cut the wad weight by half, allowing higher velocities for the same weight projectile. However, plywood wads shattered in the bore when fired with more than two pounds of black powder due to the increased pressure. Testing determined that charges over two pounds became cost inefficient (due to only moderate velocity gains for additional powder), so researchers used plywood wads and only 1 lb. (2.2 kg) of powder for all subsequent testing (Steward, 2019).

Cannon grade black powder was used as the propellent due to the properties discussed in Section 2. Researchers made the powder charges by double wrapping the desired weight of powder into a squat aluminum foil cylinder, then taping that cylinder into the center of a two-inch-thick piece of foam insulation cut to fit the bore. While protecting the black powder from sparks or electrical shock during loading, the foil could also easily be pierced with a brass rod through the projectile generator's touch hole. After piercing the powder charge, researchers then inserted an electric match through the touch hole and into the powder. The projectile generator was fired from a safe location with an electric blasting machine (Steward, 2019). Foam insulation disintegrates rapidly under high temperature and pressure, and researchers realized that it provided an additional measure of safety to the projectile generator by increasing the chamber volume. When the projectile generator is loaded, the powder charge, contained in a thick foam disk, is pressed to the rear of the projectile generator. The wad is then rammed down the bore until it contacts the foam disk, and the projectile pushed down on top of the wad (Steward, 2019).

To keep projectiles from damaging the bore, researchers cut foam sabots to fit around each projectile. This allowed the use of many different types of projectiles: wood beams, roof bolts, roof bolt plates, concrete and steel cylinders, hand tools, etc. Figure 4.8 shows a 30-pound (13.6 kg) concrete projectile with a foam sabot. Foam added very little to the overall weight of the shot and did not damage the seals or equipment as plastic or metal sabots might have (Steward, 2019).

5. FIRING AND TEST RESULTS

5.1. FIRING AND VISUAL RESULTS

Both seals were subjected to the same battery of tests. These tests included the firing of the following: a 5x5 inch five-foot-long wood beam, a 30 lb. (13.6 kg) concrete slug, four roof bolts plates in a sabot, four roof bolts in a sabot, various hand tools, a section of narrow gauge mine rail, and ten, twenty- and thirty-pound steel penetrators. Figure 5.1 shows a twenty-pound steel penetrator ready to be fired. In total, nine different projectiles

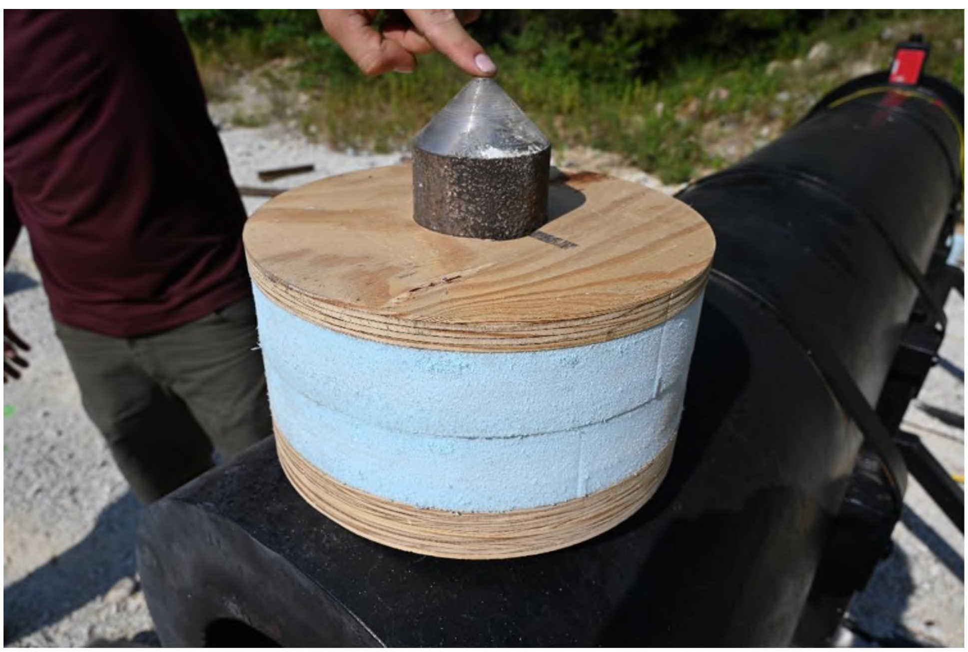


Figure 5.1 20 lb. Steel Penetrator in Sabot

were fired at the reinforced seal and 16 at the unreinforced seal. The unreinforced seal's response to impacts has been evaluated in depth by von Niederhausern (2019) and will only be discussed briefly. One pound of black powder propelled projectiles to velocities of up to 507 ft/s (158 m/s). Two sets of IR beam emitter/receivers, connected to a data acquisition system, reported the time when each beam was broken as a projectile passed through them. Combined with the distance between the sensors, researchers calculated the average velocity of each projectile.

The projectile generator was fired using an electric match from a long lead in line (100 ft, 30 m) behind cover for safety. Figure 5.2 illustrates the test layout. Figure 5.3 displays the strain gauge locations on the back side of the reinforced seal, gauge #4 is at the left side of the seal when viewed from the front. Figure 5.4 shows a projectile, a 30 lb. (13.6 kg) steel penetrator, in flight during a test; the tip of the steel penetrator is just visible

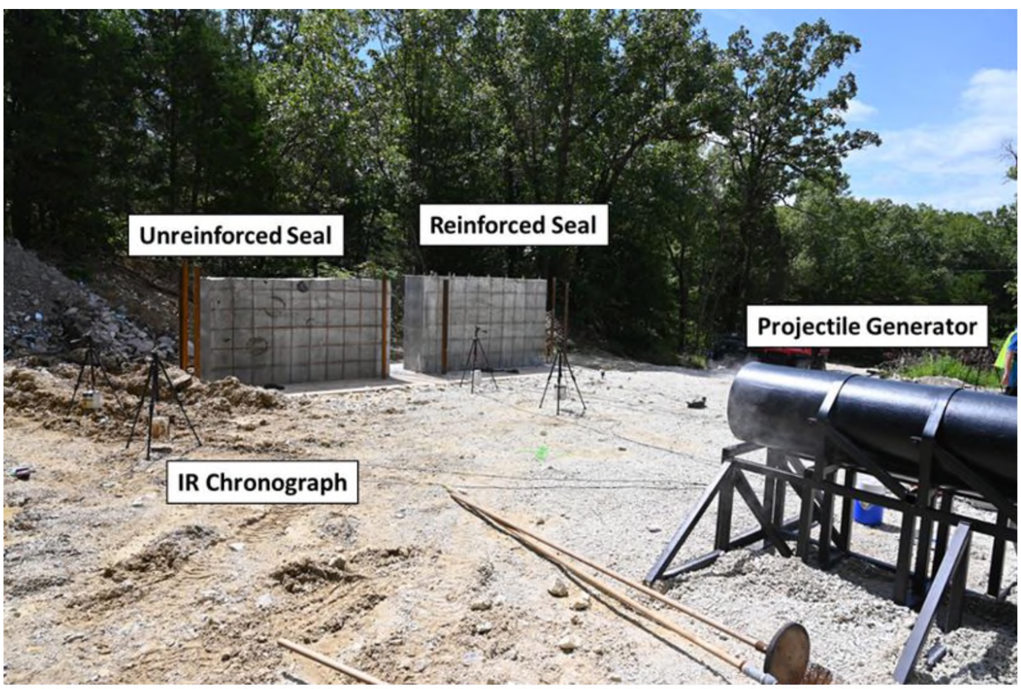


Figure 5.2 Test Layout

at the end of the sabot. Figure 5.5 shows a concrete slug (encased in a yellow cardboard tube) passing through the chronograph during a test shot to establish a velocity baseline. Figure 5.6 displays the impact of a wood 5x5 inch beam on the reinforced seal. Seal damage

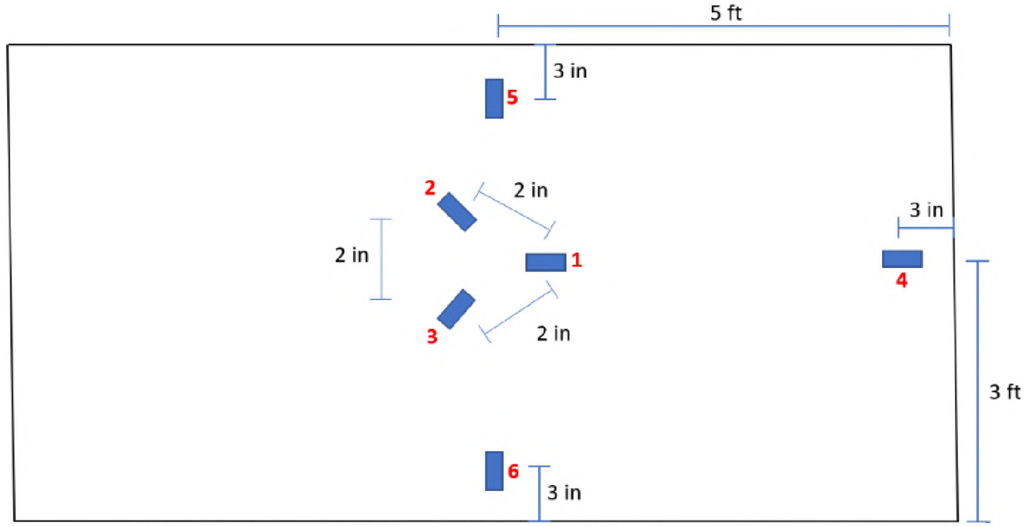


Figure 5.3 Strain Gauge Location on Reinforced Seal

Figure 5.4 Projectile in Flight

was assessed by visual inspection, strain gauge data (six strain gauges were placed on the rear face), and a LIDAR scan of the face after each shot. Damage done to the unreinforced seal was quite severe, as shown in Figure 5.7. Though catastrophic, the damage to the right hand side should not be taken as completely representative of possible in situ effects of


Figure 5.5 Shot Through Chronograph


Figure 5.6 Impact of Wood Projectile

projectile impact. A thirty-pound steel slug impact, visible as the right-most crater, caused the damage and was the last projectile fired at the unreinforced seal. The seal was already weak from multiple impacts, in particular the ten- and twenty-pound slug impacts (left and

Figure 5.7 Unreinforced 50 psi Seal Damage

middle craters). This damage was aggravated by the closeness of the` exposed right side and top of the seal. Pressure wave reflections likely greatly increased the tensile stress on the seal which lead to failure. When in place in a mine, the seal would be poured against the mine roof and walls on either side and would not have experienced the same failure, though it may have experienced failure in a different location (von Niederhausern, 2019). However, disregarding the loss of the corner, the steel slugs caused significant cracking and spalling in addition to the material ejected from the craters on impact. All other projectiles fired at this seal caused no more than surface damage. For a complete study of the impact data from the unreinforced seal, see von Niederhausern's analysis (von Niederhausern, 2019).

Figure 5.8 Reinforced 120 psi Seal Damage

The reinforced seal withstood the same battery of tests with far less obvious damage; likely due to the impact locations of the steel slugs being closer to the center of the seal (away from edges) and further from each other. As with the unreinforced seal, projectiles other than the steel slugs did only slight surface damage. However, an examination of the data does indicate that heavier projectiles, irrespective of the apparent surface damage, did cause significantly larger stresses in the rear face of the seal. Figure 5.8 shows the final state of the reinforced seal after all testing was complete. In this test, the 10 lb. (4.5 kg) steel slug impacted on the right, the 30 lb. (13.5 kg) in the center, and the 20 lb. (9 kg) on the left. The 10 lb. (4.5 kg) slug impact site exhibited slightly less damage than the others, its penetration and crush zone were significantly less. Both the 30 and 20 lb. (13.5, 9 kg) slugs penetrated deeper and had wider crushed and cracked zones, with the 20 lb. (9 kg) slug leaving a slightly deep cavity. Impact locations are shown in

Figure 5.9 30 lb. Steel Slug Impact Site

Figure 5.10 (locations are approximate for the tools, roof bolts, and roof bolt plates). The 30 lb. (13.5 kg) impact site is the middle crater in Figure 5.8 and Figure 5.9; all sites exhibited similar characteristics with a crush zone, a heavily cracked and partially ejected zone, and some small but long cracks going towards free faces. Most of the material in the crushed zone was ejected during impact. The heavily cracked zone extended about an inch outside the crush zone and was also missing some material that was ejected or fell off the seal after impact. Though far less cracking occurred outside of these zones as compared to the unreinforced seal, researchers observed some small cracks traveling between the impact sites and from the impact sites to the edges of the seal. Some very slight cracking was also only removed a small portion of the total volume lost. These scattered impacts are on the left side of the seal in Figure 5.11. Both the 20 and 30 lb. (9 and 13.6 kg) steel projectiles observed at the rear of the seal. No major failures or large cracks were observed during any of the tests and the reinforced seal did not fail as the unreinforced seal did.

LIDAR scans of the face were conducted after every shot and the final scan is shown in Figure 5.11. Table 5.1 displays the amount of volume lost from the reinforced seal. All projectiles other than the steel slugs did little visible damage to the face and penetrated to a depth of about 3.15 inches (80 mm) and removed significant amounts of material, but the 10 lb. (4.5 kg) projectile only removed a small fraction and penetrated to about 2.4 inches (60 mm). This result corresponds with their different impact energies (calculated in Table 5.2) and will be discussed in Section 5.2.

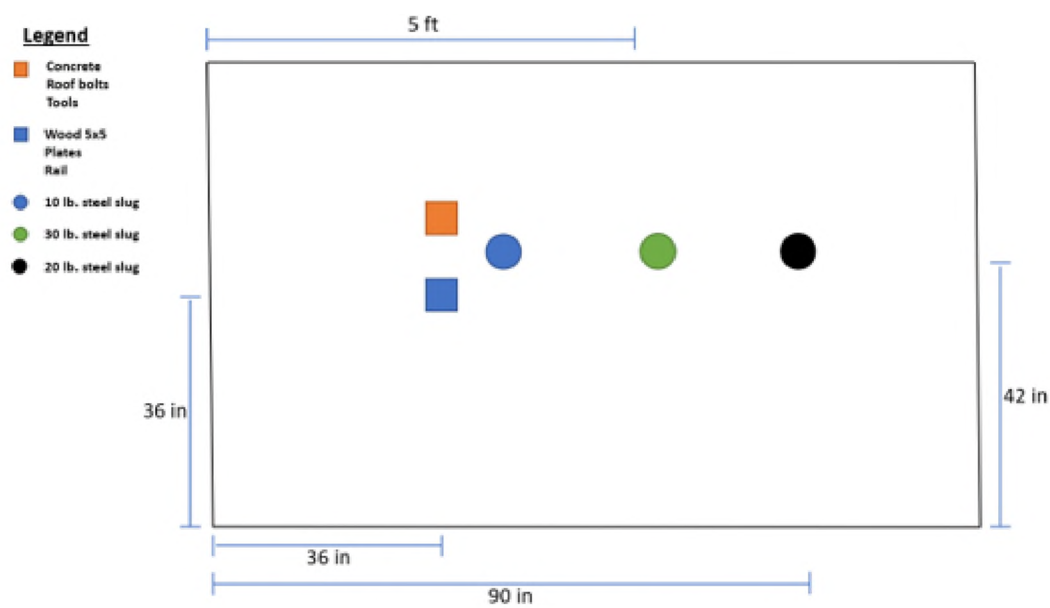


Figure 5.10 Reinforced Seal Impact Locations

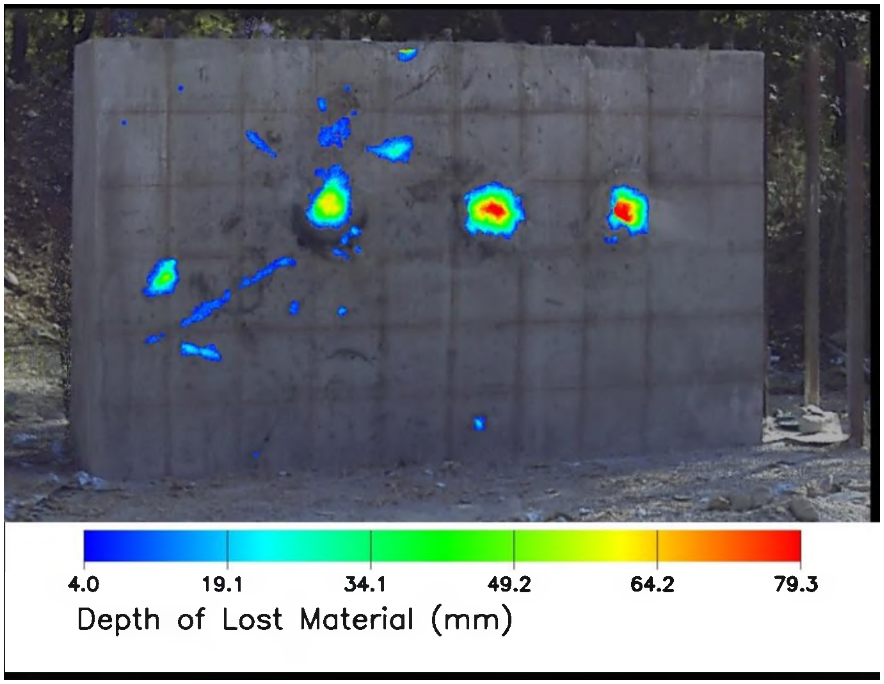


Figure 5.11 Reinforced Seal LIDAR Scan

Table 5.1. Volume Lost from Reinforced Seal

Volume Lost		
10 lb steel:	0.210	liters
20 lb steel:	1.381	liters
30 lb steel:	1.261	liters
All other proj:	0.407	liters
Total Vol:	**3.259**	**liters**

5.2. DATA ANALYSIS, RESULTS, AND DISCUSSION

To process the collected data, the author wrote a program in the Python programming language that takes in the projectile data and strain gauge data and reports projectile velocity, kinetic energy, rear face stress on all channels, and axial stress on a strain rosette centrally located on the rear of the seal. An example of the program's output files can be found at the end of Appendix B. The program analyzes the rosette data using the following equations (Hibbeler, 2014):

$$\epsilon_a = \epsilon_x * cos^2\theta_a + \epsilon_y * sin^2\theta_a + \Upsilon_{xy} * sin\theta_a * cos\theta_a$$
$$\epsilon_b = \epsilon_x * cos^2\theta_b + \epsilon_y * sin^2\theta_b + \Upsilon_{xy} * sin\theta_b * cos\theta_b \quad (18)$$
$$\epsilon_c = \epsilon_x * cos^2\theta_c + \epsilon_y * sin^2\theta_c + \Upsilon_{xy} * sin\theta_c * cos\theta_c$$

where ϵ_a, ϵ_b, and ϵ_c are the strains on the rosette gauges as labeled in Figure 5.12 (Hibbeler, 2014), the angles are between the gauges as labeled θ_a, θ_b, θ_c and ϵ_x, ϵ_y, and Υ_{xy} are the strain in x, y, and shear respectively. These equations solve for any angles regardless of the orientation of the gauges. Principle strain was solved using equation (19) (Hibbeler, 2014), then converted to stress:

$$\epsilon_1, \epsilon_2 = \frac{\epsilon_x + \epsilon_y}{2} +/- \sqrt{\left(\frac{\epsilon_x - \epsilon_y}{2}\right)^2 + \left(\frac{\Upsilon_{xy}}{2}\right)^2} \quad (19)$$

The program solves for the three equations (18) simultaneously and reports strain and stress as the simple stress on each gauge and uses equation (19) to solve the principle strains from the rosette. Maximum stress is solved by multiplying by concrete’s average modulus of elasticity by the largest (most positive) principle strain. Velocity and kinetic energy are also found from the beam break times of the chronograph. Table 5.2 displays the projectile weight, velocity, and kinetic energy. The weight in Table 5.2 is broken down by the projectile weight and the total weight of the shot including the wad and sabot. Kinetic energy is given in ft-lbs. and Joules.

Unfortunately, the cable connecting the data acquisition system to one of the chronograph IR sensors became disconnected during several shots. The sabot holding the rail and its wad became detached in the barrel during firing, resulting in a very low velocity impact. Therefore, the rail was not considered in further data analysis.

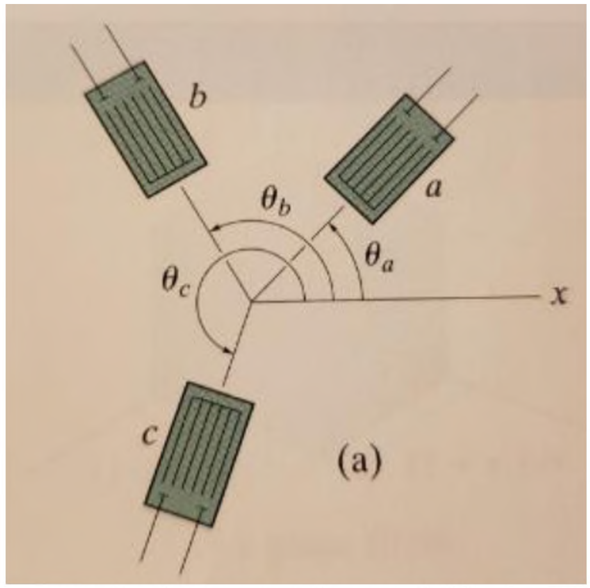


Figure 5.12 Strain Rosette Diagram (Hibbeler, 2014)

Table 5.2. Projectile Data.

	Weight			Kinetic Energy	
Projectile	**Projectile**	**Total**	**Vel (ft/s)**	**ft-lbs**	**Joules**
Concrete	30.5	38	431	87,931	119,234
5x5 wood	31.5	38	423	87,595	118,779
20 lb slug	21	31	477	74,313	100,769
30 lb slug	31	42.5	379	69,143	93,758
Bolts	18	24	430	51,680	70,076
10 lb slug	10	18.5	507	40,000	54,240
Plates	10	15	X	X	X
Rail	27	34	X	X	X
Tools	5	12	X	X	X

Data in Table 5.2 shows some of the interesting properties of the projectile generator. A velocity curve exists, with light weight projectiles having a higher velocity compared to heavier ones (excepting the roof bolt velocity). As is typical of guns, an obvious optimal weight is indicated by the data, even in this small data set. This optimal shot weight is a result of a large number of factors including the burn rate of the powder and length of the bore. The 20 lb. (9 kg) steel slug, though attaining a lower velocity to the 10 lb. (4.5 kg), has nearly double the kinetic energy. The 30 lb. (13.6 kg) slug has slightly lower kinetic energy than the 20 lb. (9 kg), and the 5x5 inch wood post and the concrete slug had nearly the same total weight, velocity, and the highest kinetic energies of the projectiles. A total shot weight of about 40 pounds (18 kg) is likely the optimal weight and the most energy efficient for the powder, barrel diameter and length.

Principle stresses and max tensile stress from each impact are shown in Table 5.3, the impacts are in order of greatest to least measured tensile stress and the sensor that registered the max value is also listed. Strain was converted to stress using a modulus of elasticity of $4*10^6$ psi (27.6 GPa), which is average for concrete. Principle stresses are σ_1 and σ_2; positive numbers represent tension. Due to the large amount of data gathered from each shot not all graphs can be shown in this section; they are available in Appendix B. Figures 5.13 through 5.16 are graphs produced by the analysis program for the concrete projectile; 5.13 and 5.14 display micro-strain on strain gauges 1 and 2, 5.15 displays a bar chart comparison of all strain gauge stress from that shot (CH3 corresponding with strain gauge 1, CH4 with 2, etc.), and Figure 5.16 displays the principal stresses.

Analysis of the data revealed surprising results. As shown in Tables 5.2 and 5.3, the projectiles with the highest kinetic energy did not necessarily cause the greatest stress on the rear face of the seal. Though the wood 5x5 and concrete projectiles had the most kinetic energy and caused the most stress, the 20 lb. (9 kg) and 30 lb. (13.6 kg) steel slugs had only slightly less and caused, on average, about 125 psi (862 kPa) less stress. All projectiles except the steel slugs caused no visible damage or only very slight surface damage, the steel projectiles did cause cratering. This may explain why they caused less

Table 5.3 Stress Data

Stress (psi)				
Projectile	**Max Tensile**	**Sensor**	**σ1**	**σ2**
Concrete	238.5	2	447	-138.8
5x5 wood	197	3	287.8	17.6
10 lb slug	167.3	2	256	-67.3
Roof Bolts	113.8	4	232.9	-136.7
Tools	101.4	5	133.1	-31.2
30 lb slug	100.1	1	101	-238.5
20 lb slug	87.1	3	156.3	-21.1
Plates	70.6	4	60.1	-30.4

rear face stress; some of their kinetic energy was absorbed in crushing the concrete at the impact area; and that crush period also resulted in a lower pressure impulse into the seal. Note that the 10 lb. (4.5 kg) steel slug created a smaller impact crater and caused significantly more rear face stress than the 20 or 30 lb. (9, 13.6 kg) slugs (~70 psi, 482 kPa), while attaining only just over half their kinetic energy. As shown in Table 5.1, the 10 lb. (4.5 kg) slug also only crushed and removed a small fraction of concrete compared to the 20 and 30 lb. (9, 13.6 kg) slugs.

Concrete has a tensile strength generally ranging from 400 to 700 psi (2,758 to 4,826 kPa). As shown in Table 5.2, the lower end of that range was only exceeded once by the principal stress caused by the concrete projectile. Most induced stresses were at least within 25% of the 400-psi (2758 kPa) tensile strength. Due to the location of the impacts and sensors, stresses high enough to causes cracking on the rear face may not have been recorded. The strain rosette was centrally located on the reinforced seal (Figure 5.3) during all shots; principle stresses at other locations may have reached the failure point. Stresses within the seal likely reached levels above the concrete's tensile strength between the impact points and the rear face, especially for steel slugs. The data recorded by the strain gauges was in most cases noisy to draw solid conclusions about wave reflections, other than that there were many. Also important in consideration of the data is the types of pressure waves created in the rear of the seal.

Due to the placement of the strain gauges on the rear of the seals, they will experience both transverse and longitudinal waves and, as those waves reflect off the various surfaces and bottom of the seal, they will also experience wave superposition and interference. This greatly complicates analysis, especially when trying to determine strains and stresses other than those directly measured by the strain gauges. Axial stress (in this case, through the seal perpendicular to the rear face and the strain gauges) is an important component of concrete failure and causes scabbing when high pressure waves in a solid material meet a low impedance zone such as air (Cooper, 2010). Several attempts were made to solve for axial stress, but due to the complex nature of the waves and the single strain rosette for data, these attempts did not result in axial stress values that the author deemed legitimate (they varied from 550 psi to 1,700 psi, 3,792 to 11,721 kPa) because no scabbing was observed on the rear face. Therefore, conclusions about the stress state of the concrete were only drawn from the strain gauge readings and the principles strains and stresses derived from them.

Finally, a comparison of the data from the 120 psi (827 kPa) reinforced seal against the 50 psi (345 kPa) unreinforced seal (von Niederhausern, 2019) shows that the two seals reacted somewhat similarly when comparing the relative stresses of the different projectiles, though they experienced velocity and impact energy differences. Both seals generally experienced high rear-face stress from the concrete impacts relative to other projectiles, except for the hand tools shot at the 50 psi (345 kPa) seal which may have

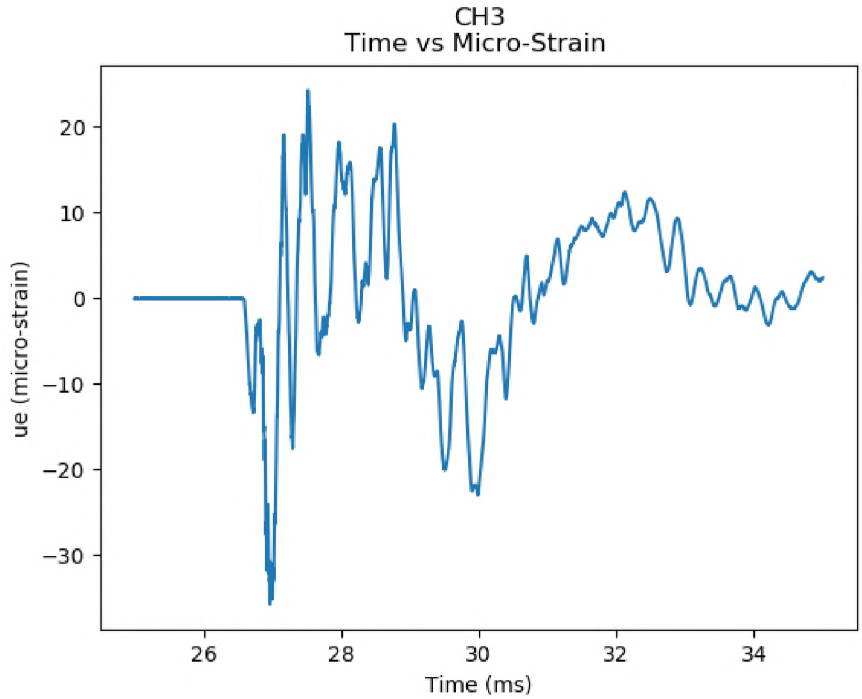


Figure 5.13 Strain Gauge #1 Strain, Concrete Projectile

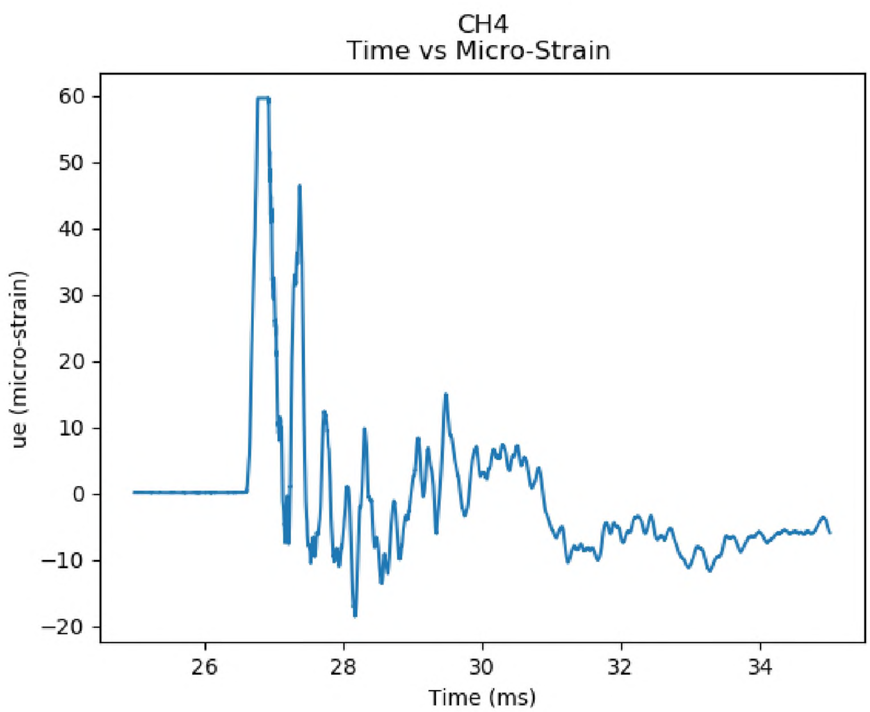


Figure 5.14 Strain Gauge #2 Strain, Concrete Projectile

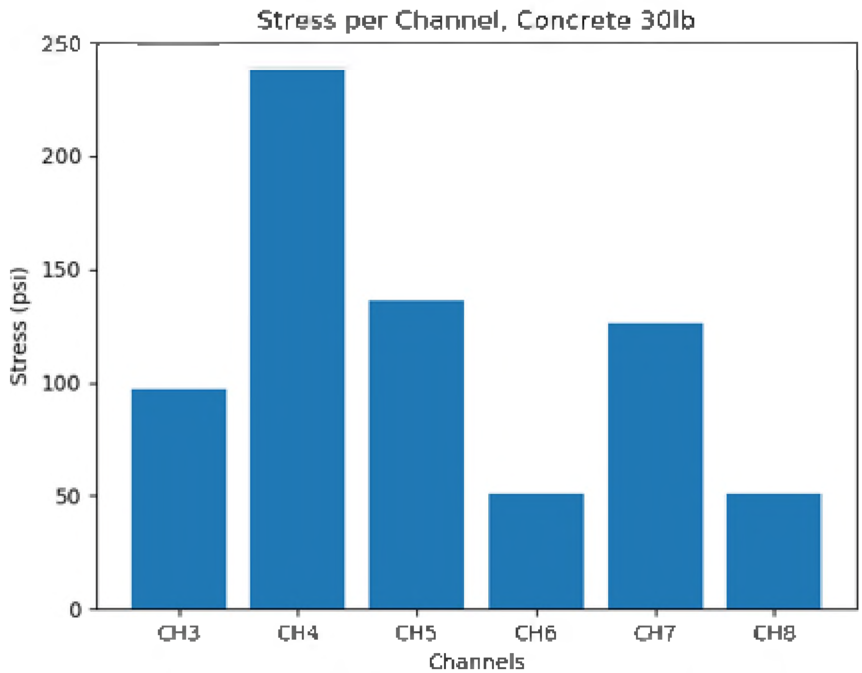


Figure 5.15 Stress per Channel, Concrete Projectile

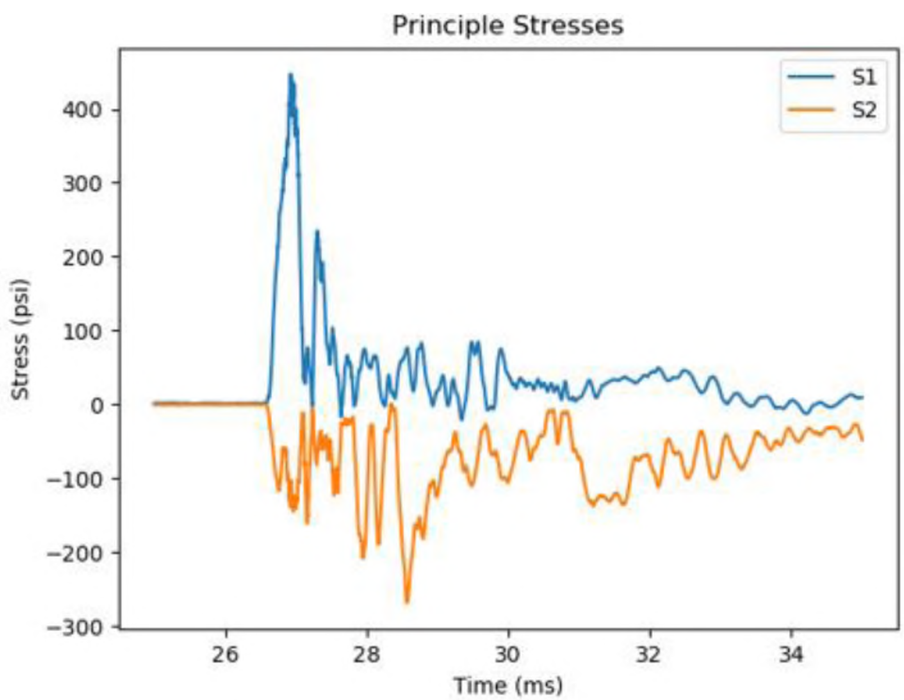


Figure 5.16 Principle Stresses, Concrete Projectile

achieved a higher velocity than when shot at the 120 psi (827 kPa) seal (the velocity data was not collected for that shot), and the wood projectile which achieved far higher velocity in the test on the 120 psi (827 kPa) seal. The 30 lb. (13.6 kg) concrete projectile caused the second highest stress in the 120 psi (827 kPa) seal and third highest in the 50 psi (345 kPa) seal. The steel projectiles caused almost all the surface damage and cracking in both seals, but not the highest rear-face tensile stress.

6. CONCLUSION

6.1. DISCUSSION OF CONCLUSIONS

The projectile generator has proven to be very effective and useful in propelling a large assortment of various materials that could become projectiles in the event of an explosion in an underground mine. Though the initial target velocity of 700 ft/s (213 m/s) has yet to be reached, the projectile generator did accelerate projectiles up to sufficient velocities and kinetic energies necessary to cause failure; catastrophic failure in the case of the unreinforced seal. Challenges to the project included the design of the projectile generator for safety and simplicity at a low cost, overcoming the parasitic weight of the wad, building sabots that successfully protected the barrel from damage, data loss, and analysis of the complicated pressure waves induced in the seals.

Conclusions reached from this research:

- The projectile generator can produce muzzle energies of at least 88,000 ft-lbs. (119,000 J)
- The projectile generator can accelerate projectiles to at least 507 ft/s (154.5 m/s)
- The projectile generator can safely contain the pressure produced by 3 lb. of black powder (1.4 kg) per 2-inch (5 cm) length of chamber, provided disintegrating foam or other material is used as the powder container
- Traditional, simplified black powder reaction equations do not accurately represent the complex combustion products and may over-estimate the actual gas production by up to 31%. Based on historical experimental data, black powder made in a 75/15/10 ratio

of potassium nitrate, charcoal, and sulfur produces about 5.13 moles/lb. (11.3 moles/kg) of gas vs 6.7 moles/lb. (14.8 moles/kg) for the simplified equation

- Mine seals can be damaged by materials commonly found in mines when propelled to velocities possible during an underground explosion
- Hard, dense, non-deforming projectiles can cause cratering, cracking, and catastrophic failure of mine seals
- Soft, deforming, or disintegrating projectiles can cause stress greater than the tensile strength of concrete while not causing immediately visible damage to mine seals
- Small, dispersed objects (hand tools, roof bolt plates, roof bolts) do not pose a significant threat to mine seals

The data shows that dense, hard objects of sufficient mass (the steel slugs) cause significant surface damage by cratering and spalling the impact face but cause less stress in the rear face of the reinforced seal. Common steel objects found in a mine may include drill steel, I-beams, and equipment parts. Projectiles that deform or disintegrate upon impact (concrete, wood) cause little to no visible surface damage but may still generate rear-face stress over the yield strength of concrete. Common frangible objects found in a mine may include large rocks and boulders, concrete chunks, and wood beams. Significant stress, cracking, cratering, or ejection of material at the impact face was observed in both seals when struck with both deforming and non-deforming projectiles. Consequently, this study has shown that the materials found in an underground coal mine can cause significant structural damage to coal mine seals when propelled to velocities possible in a methane-coal dust explosion.

6.2. FUTURE WORK

This area of research has many avenues for further work:

- Additional impact testing, with higher energy projectiles
- Internal stress analysis and the consequences of wave reflections
- Response of the seals in-situ when surrounded by and attached to rock
- Test different seal designs and types
- Failure rates and mechanisms when the seal is struck by high velocity projectiles while also in a pressurized, explosive environment
- Modeled and full-scale methane and coal dust explosion analysis to estimate pressures and projectile velocities
- Analysis of how far objects in a mine may travel during an explosion and develop a standard distance for clearing areas inby the seal. The current standard of 50 ft (15.2 m) may not be adequate
- Further analysis of the effect of piling waste rock (gob) on the inby side of the mine to protect seals from projectiles and diffuse pressure waves (Perry, 2010)

6.3. DESCRIPTION OF APPENDICIES

A. Appendix A is a page from the mill test report provided with the steel pipe that became the barrel. The yield and tensile strengths used for calculating max barrel pressure are found under the "Tensile Test Results" heading and are given in psi.

B. Appendix B contains that data collected from each shot on the reinforced seal, displayed in graph form. Graphs labeled "#.1" show the principle stresses as calculated by the analysis program and graph the max tensile stress measured on each channel. Graphs labeled "#.2" show the data from each sensor, displayed as micro-strain vs time.

C. Appendix C displays the text of the program used to analyze the sensor data. This program was written in the Python 3.8 programming language as two separate scripts. The first sets up a file and naming system to organize multiple projects and testing dates, then calls the second script to perform the actual analysis of data. The second script performs all analysis and uses the sympy library to solve the principle stress equations, creates the graphs shown in Appendix B using the matplotlib library, and also writes data and relevant information in text files, an example of which is shown at end of the appendix. The settings and other constants needed for the analysis are also found in an example setup text file at the end of the appendix.

APPENDIX A.

MILL TEST REPORT

Vallourec Deutschland GmbH (A01)		MATERIAL TEST REPORT (A02)
Werk Rath-Pilger Rather Kreuzweg 106 40472 DÜSSELDORF GERMANY	vallourec	3.1 EN 10204:2004
		No. : 19005RP16 (A03) Page: 4 / 5 Date: 06.09.2016

TENSILE TEST RESULTS

(B07.1) Heat	(C00.1) Test Piece	(C10) Diameter	(C11) YS	(C12) TS	(C13) Elong.	(C14.1) Ratio	(C15) Red.Area				
		Inch	$R_{p0.2}$ psi	R_m psi	2" %	R/R_m -	Z %				
183883	5429	00.49	61349	112545	21.5	0.55	54				
186343	5431	00.49	61059	113125	21.5	0.54	52				

ADDITIONAL TEST S 2.2

HARDNESS TEST RESULTS

Location (C01.1) MIDDLE CONCERNING WALL

(B07.1) Heat	(C00.1) Test Piece	(C31) Value	(C31) Value	(C31) Value							
		HB	HB	HB							
183883	5429	253.0	248.0	237.0							
186343	5431	228.0	235.0	230.0							

ADDITIONAL TEST S 2.1

(D55)

OTHER TESTS ON PIPE

Test	Conditions	Test rate	Result
HEAT TREATMENT	NORMALIZED	5 MIN 1688 DEGREES F AIR	
APPEARANCE AND DIMENSIONS (D01)		EACH PIPE/ TUBE	SATISFACTORY
MERCHANDISE IS FREE OF MERCURY CONTAMINATION AND NO WELD REPAIR WAS PERFORMED..			

(A04, B06)

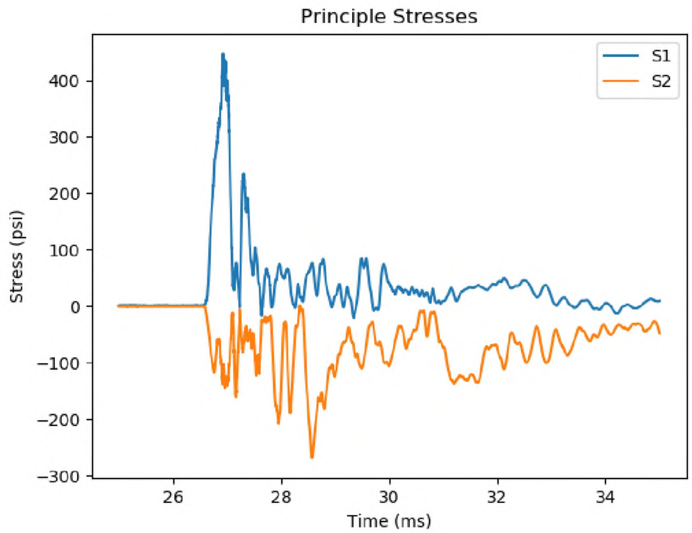
MARKING, IDENTIFICATION

VALLOUREC

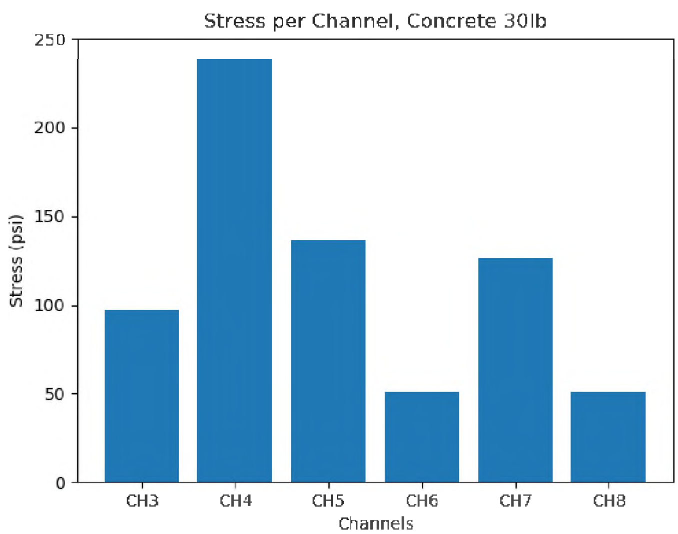
DIE STAMPED ON ONE SIDE ROUND NOSE TOOLS VALLOUREC LOGO MANUFACTURER'S NAME MANUFACTURER'S MARK COUNTRY OF ORIGIN GERMANY HEAT-NO. DIE STAMPED BOTH SIDES PIPE NO. (TALLY-NR.) DIE STAMP SURROUNDED WHITE PAINT STENCILED ON BOTH SIDES VALLOUREC LOGO MANUFACTURER'S NAME MANUFACTURER'S MARK TERMS OF DELIVERY A519 GRADE4140MOD. WA PIPE NO. (TALLY-NR.) SPECIALTY CUSTOMER ORDER-NO. 108331 ORDER-NO. USER W42183 PORT OF DESTINATION CLEVELAND / OHIO 20 X 4 HEAT-NO. TUBE/PIPE WEIGHT LBS NORMALIZED LONGITUDINAL COLOUR STRIPE RED

APPENDIX B.

STRESS ANALYSIS DATA



a.



b.

Figure B.1 Concrete Stresses

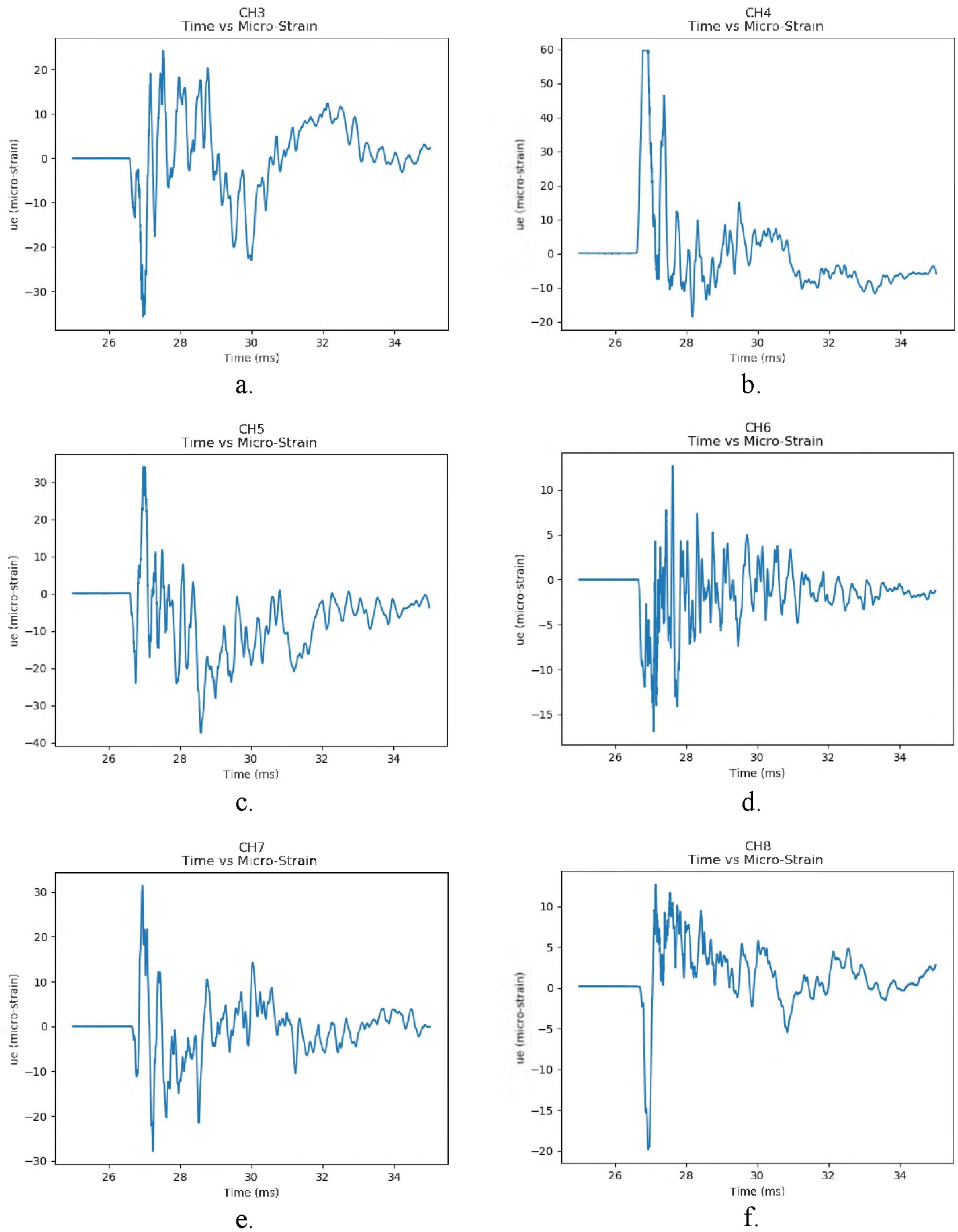
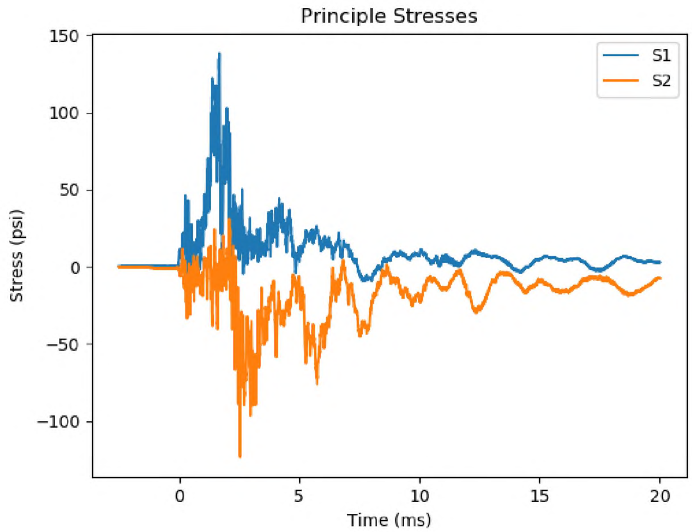
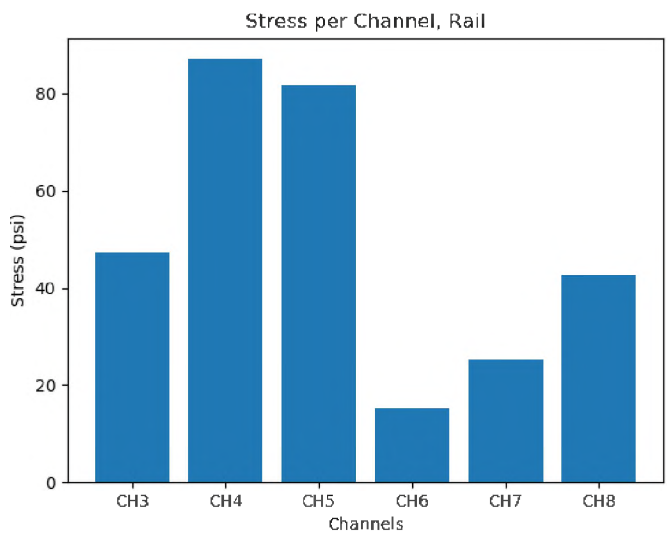


Figure B.2 Concrete Sensor Data



a.



b.

Figure B.3 Rail Stresses

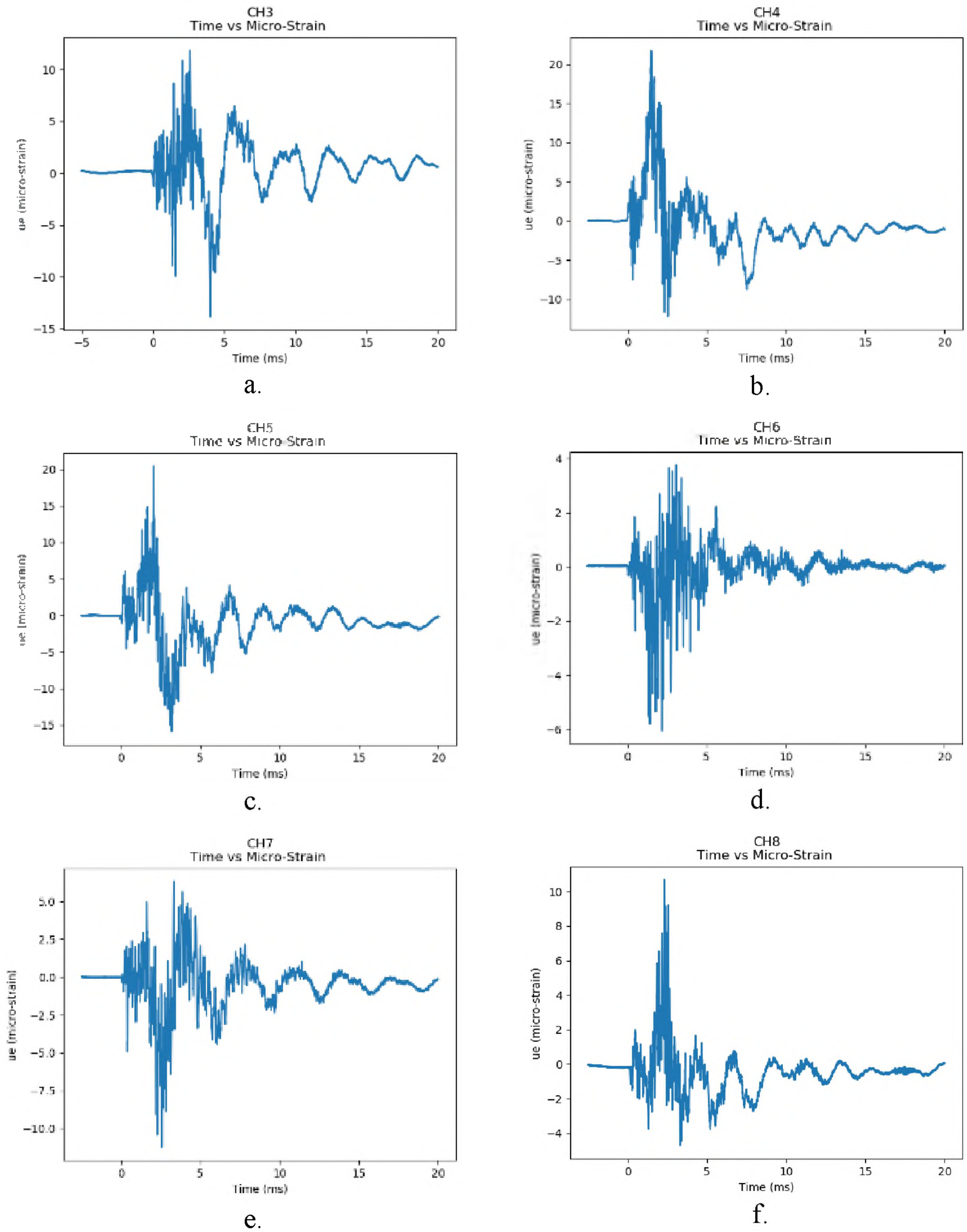
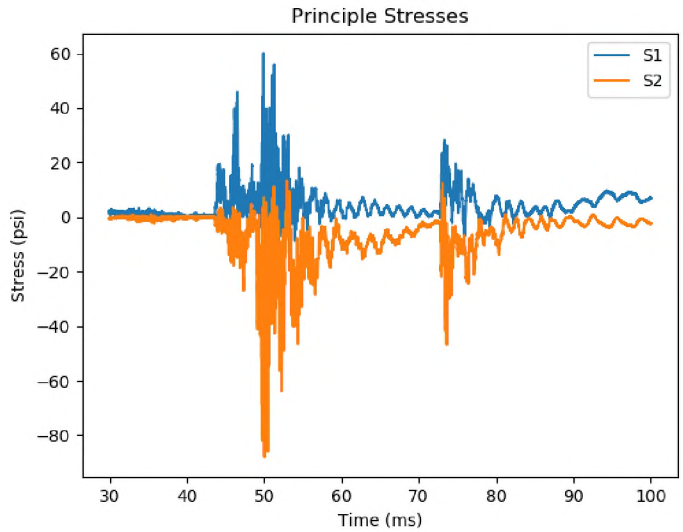
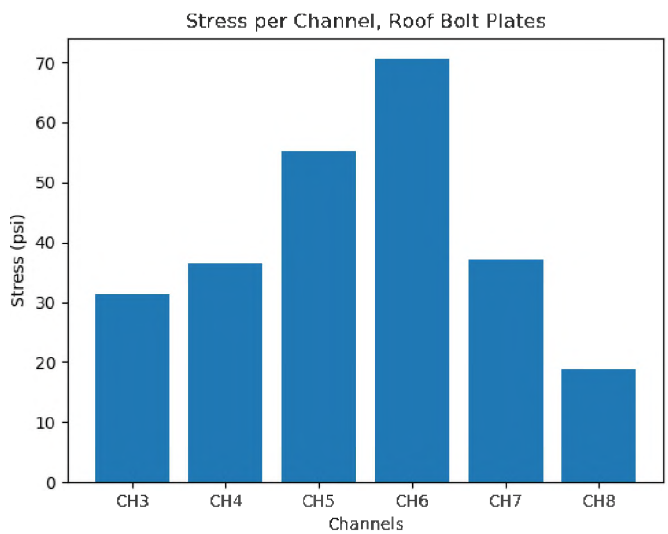


Figure B.4 Rail Sensor Data



a.



b.

Figure B.5 Roof Bolt Plate Stresses

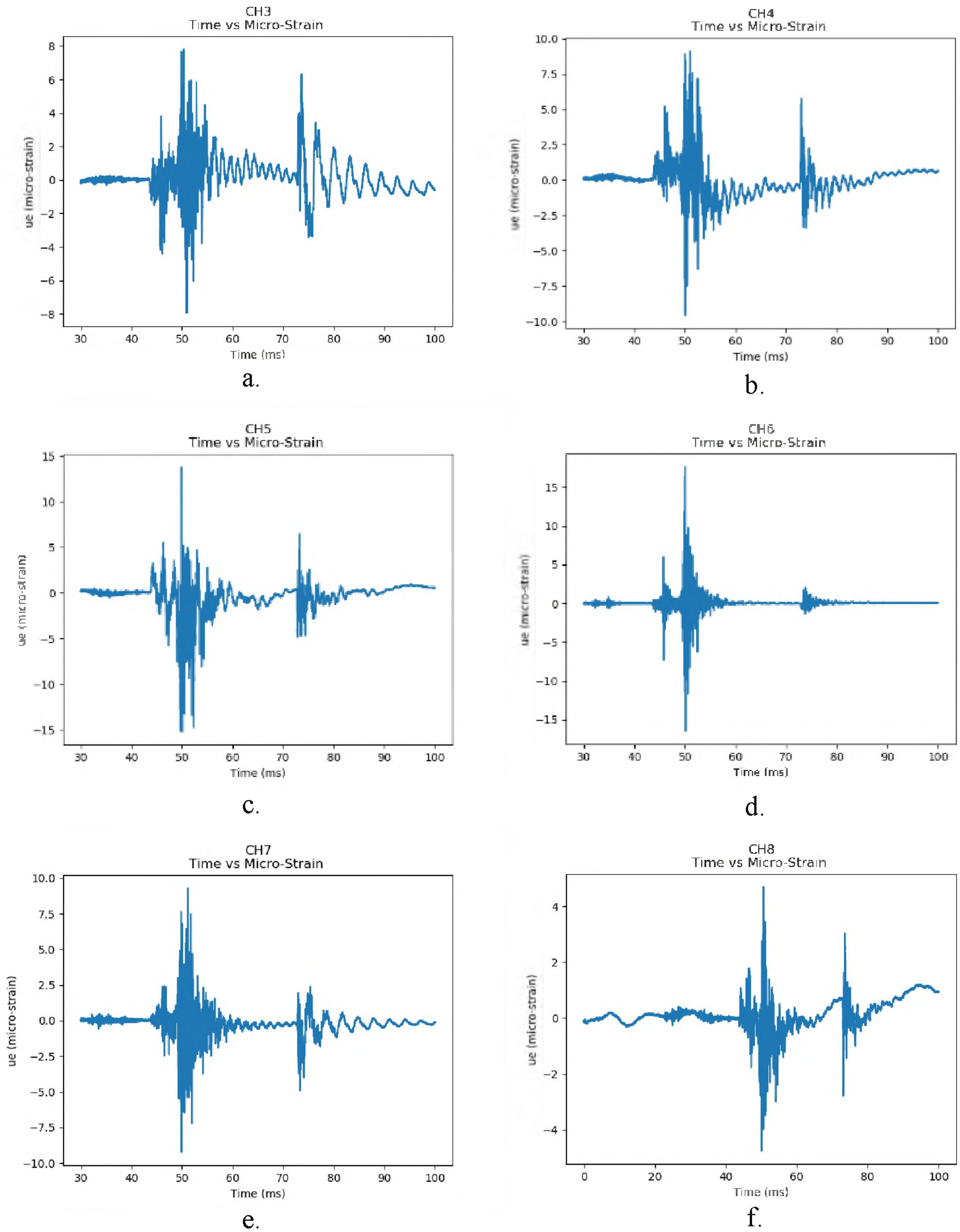
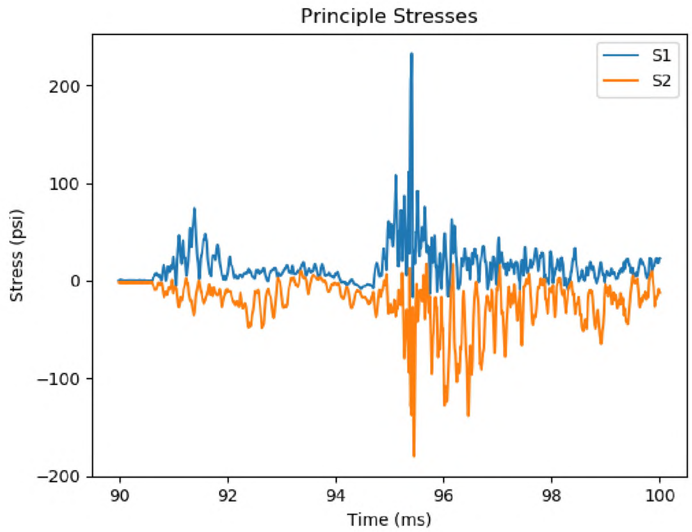
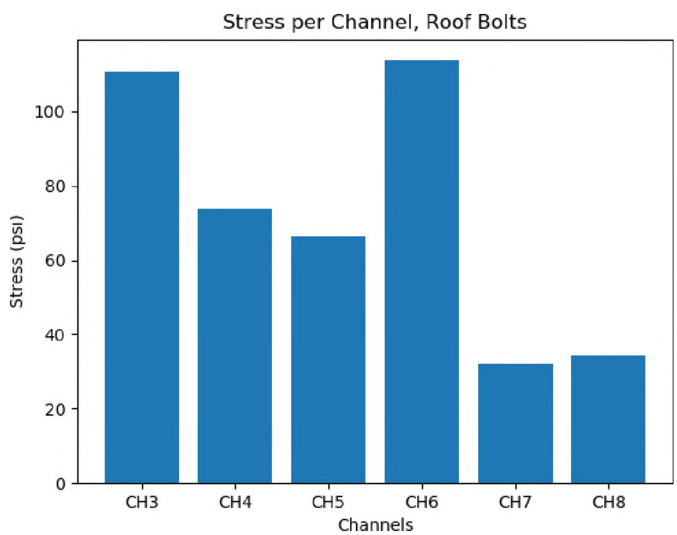


Figure B.6 Roof Bolt Plate Data



a.



b.

Figure B.7 Roof Bolt Stresses

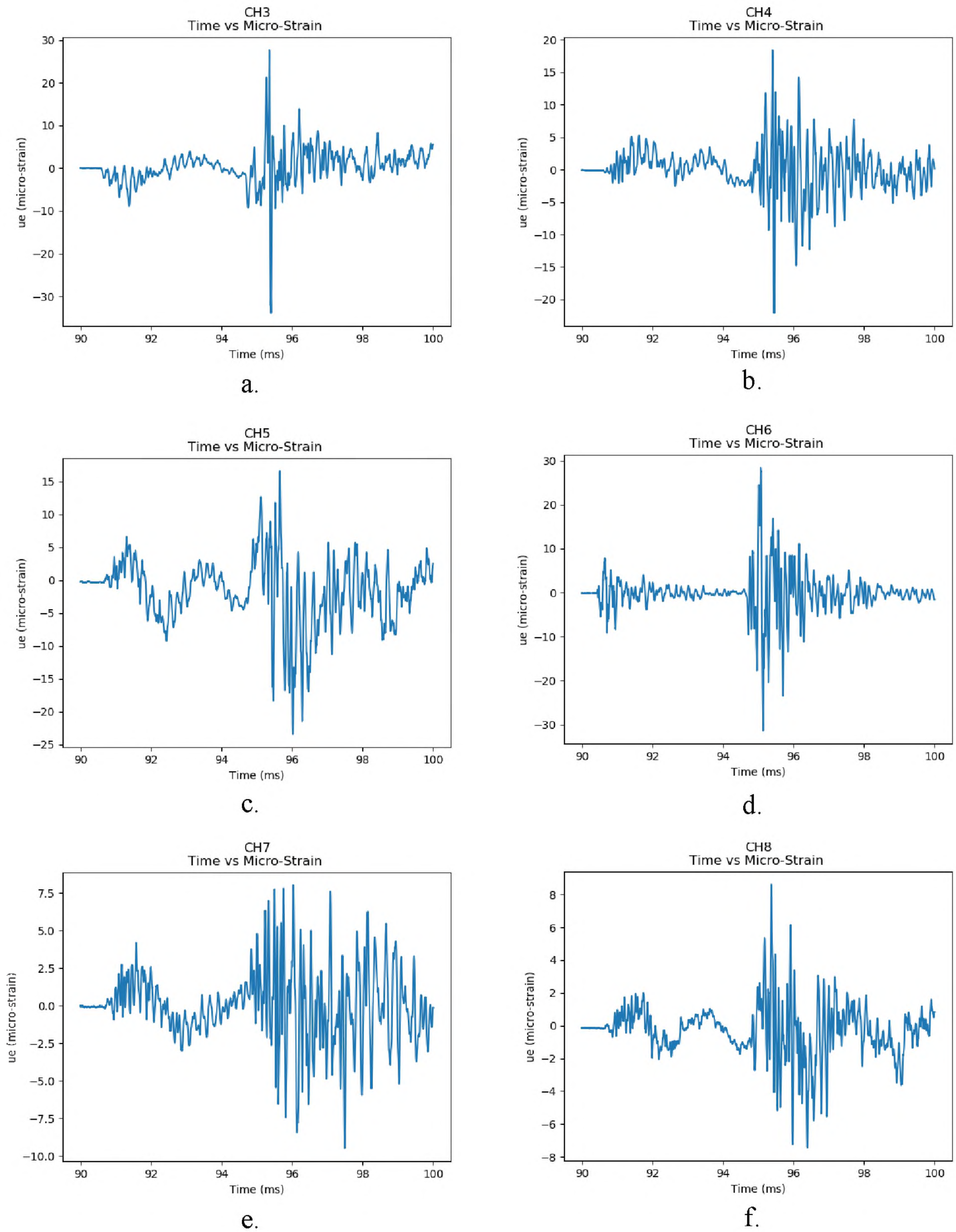
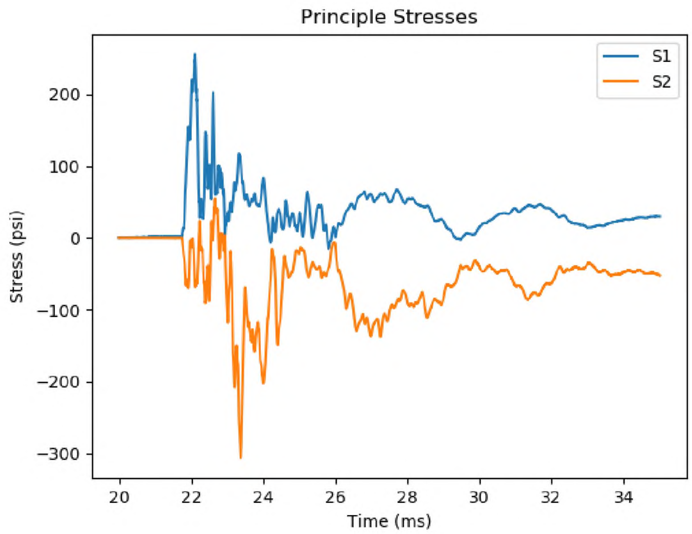
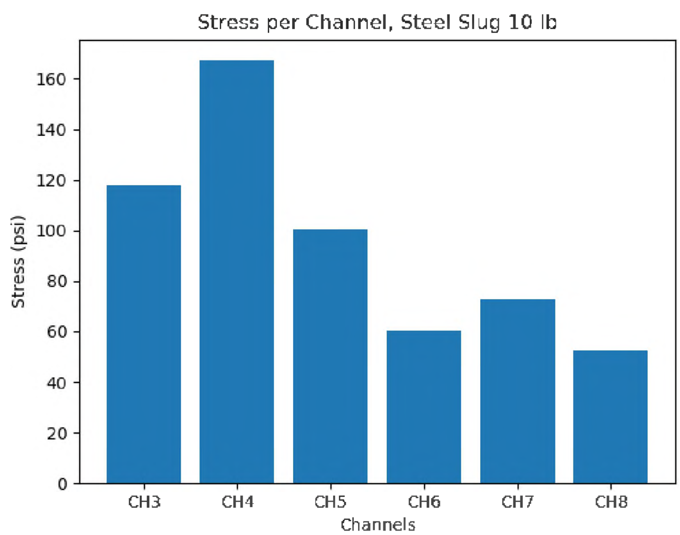


Figure B.8 Roof Bolt Data

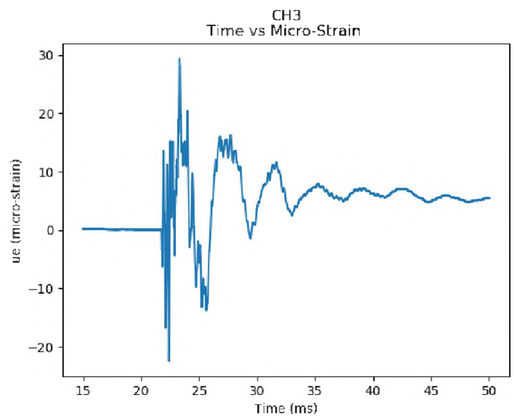


a.

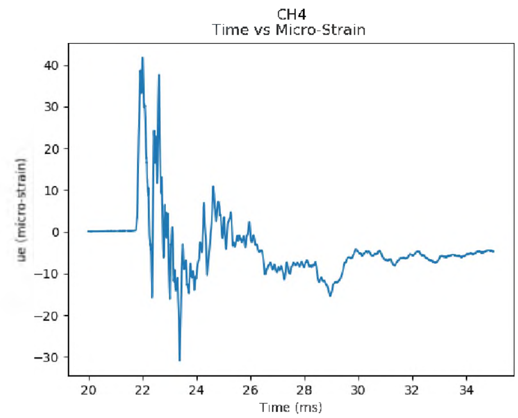


b.

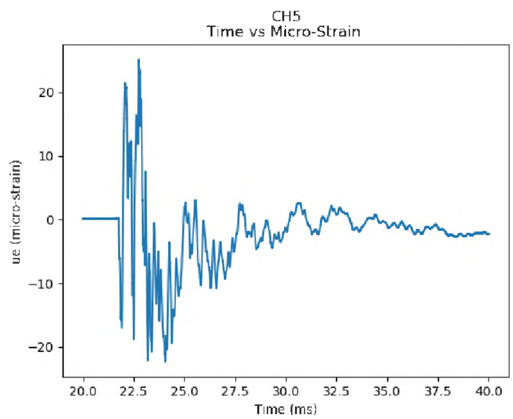
Figure B.9 Steel Slug 10 lb. Stresses



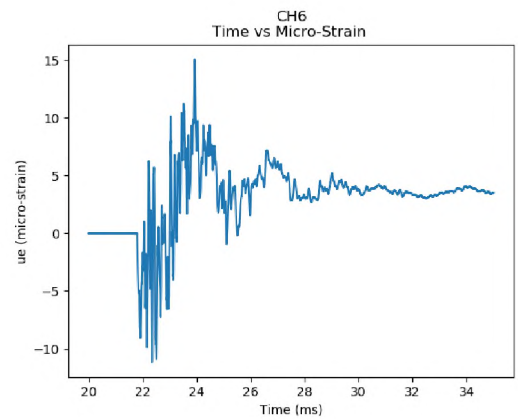
a.



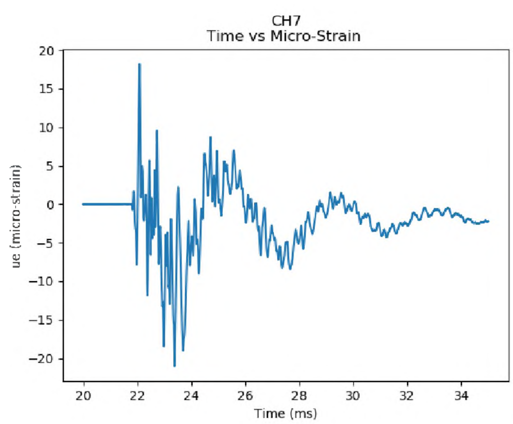
b.



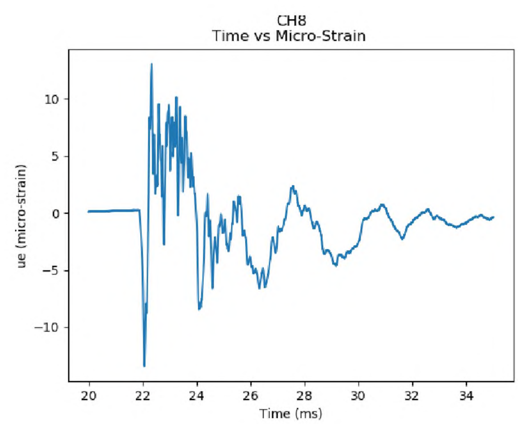
c.



d.

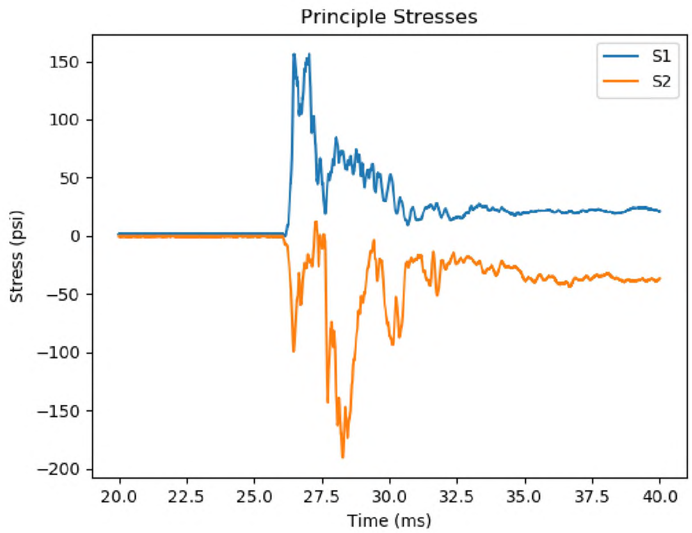


e.

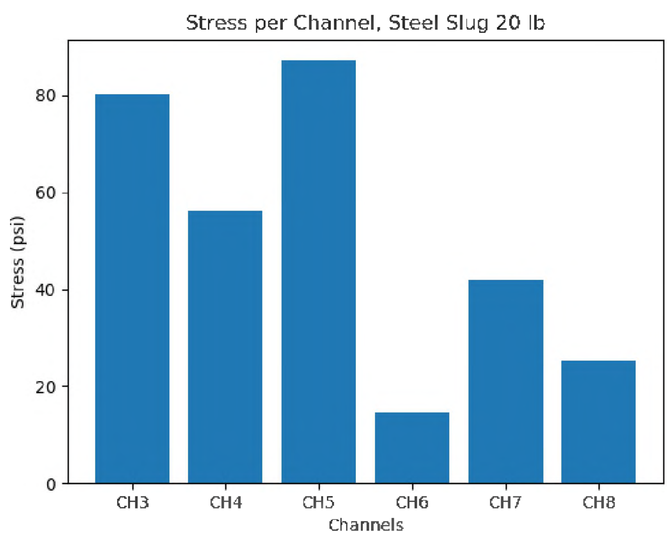


f.

Figure B.10 Steel Slug 10 lb. Data



a.



b.

Figure B.11 Steel Slug 20 lb. Stresses

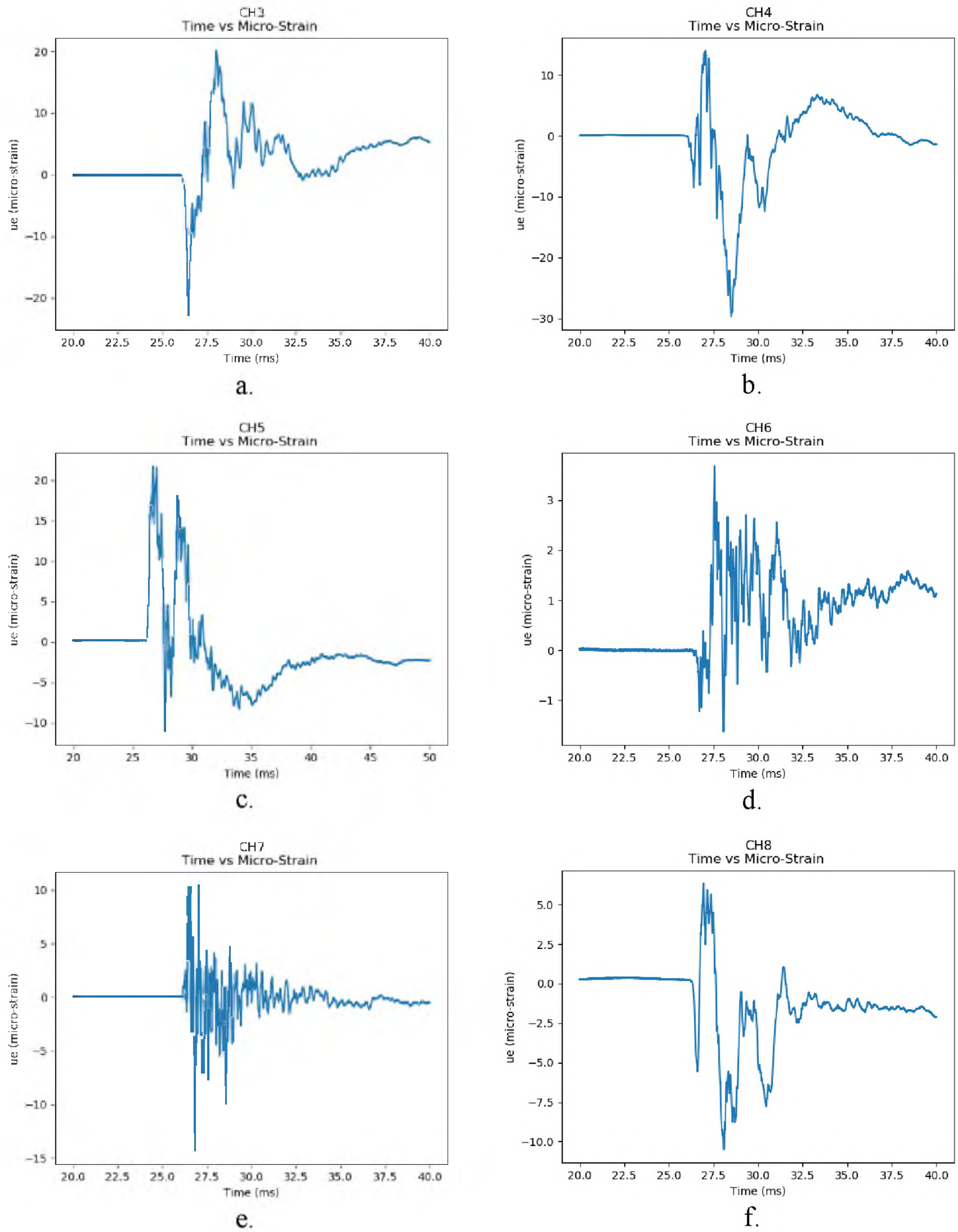
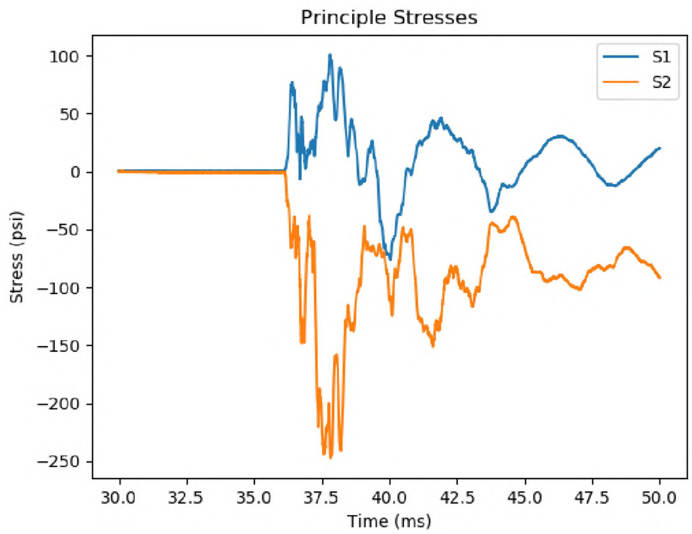
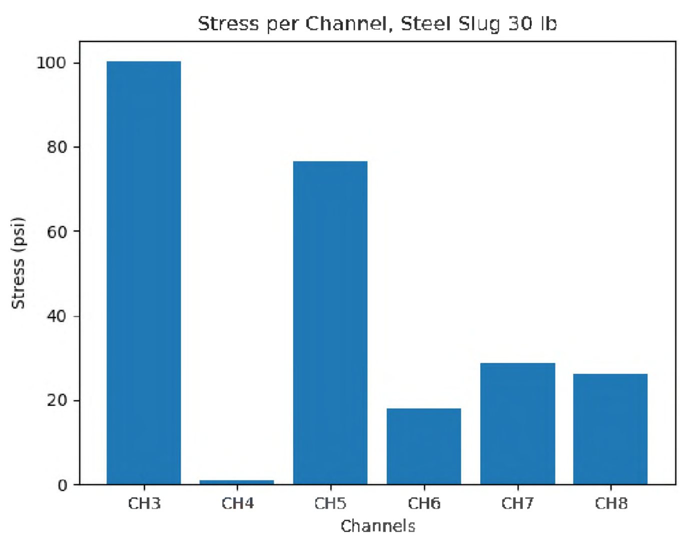


Figure B.12 Steel Slug 20 lb. Data



a.



b.

Figure B.13 Steel Slug 30 lb. Stresses

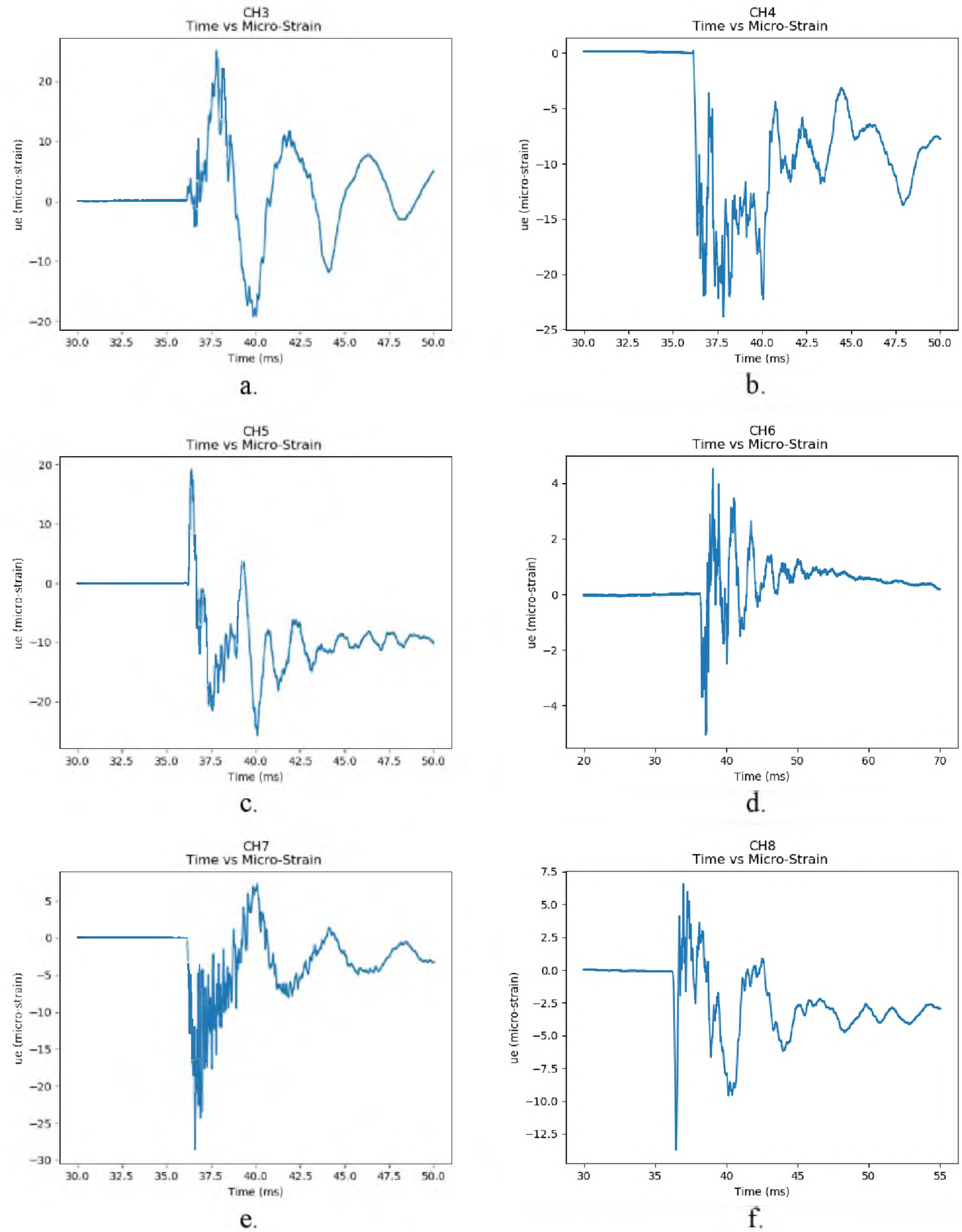
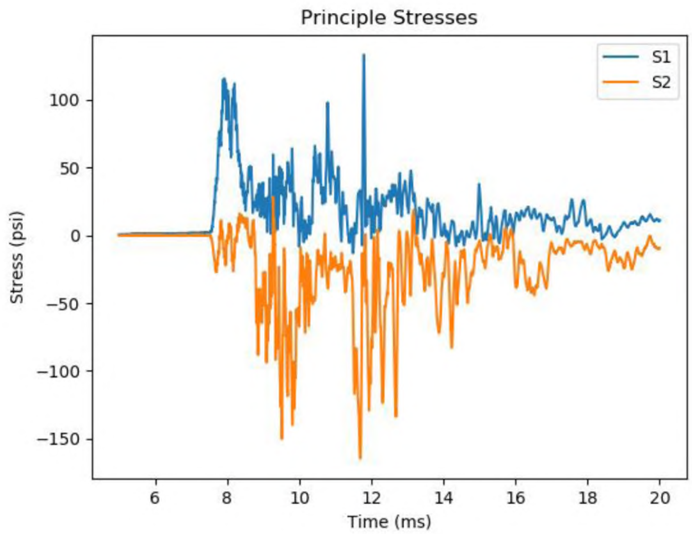
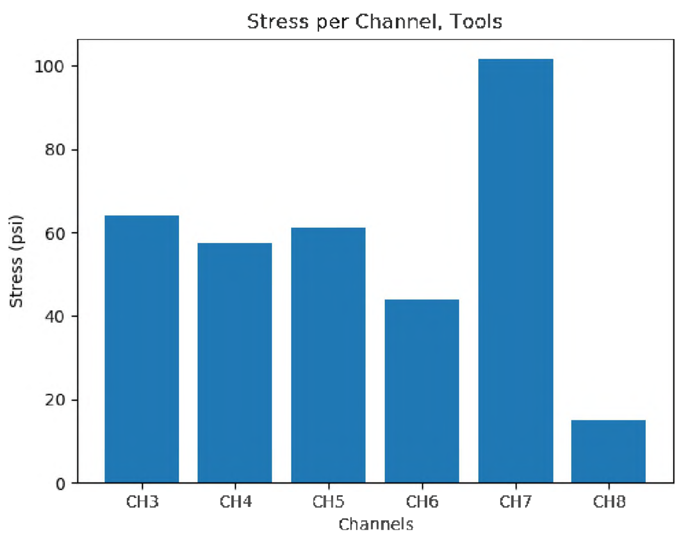


Figure B.14 Steel Slug 30 lb. Stresses



a.



b.

Figure B.15 Tools Stresses

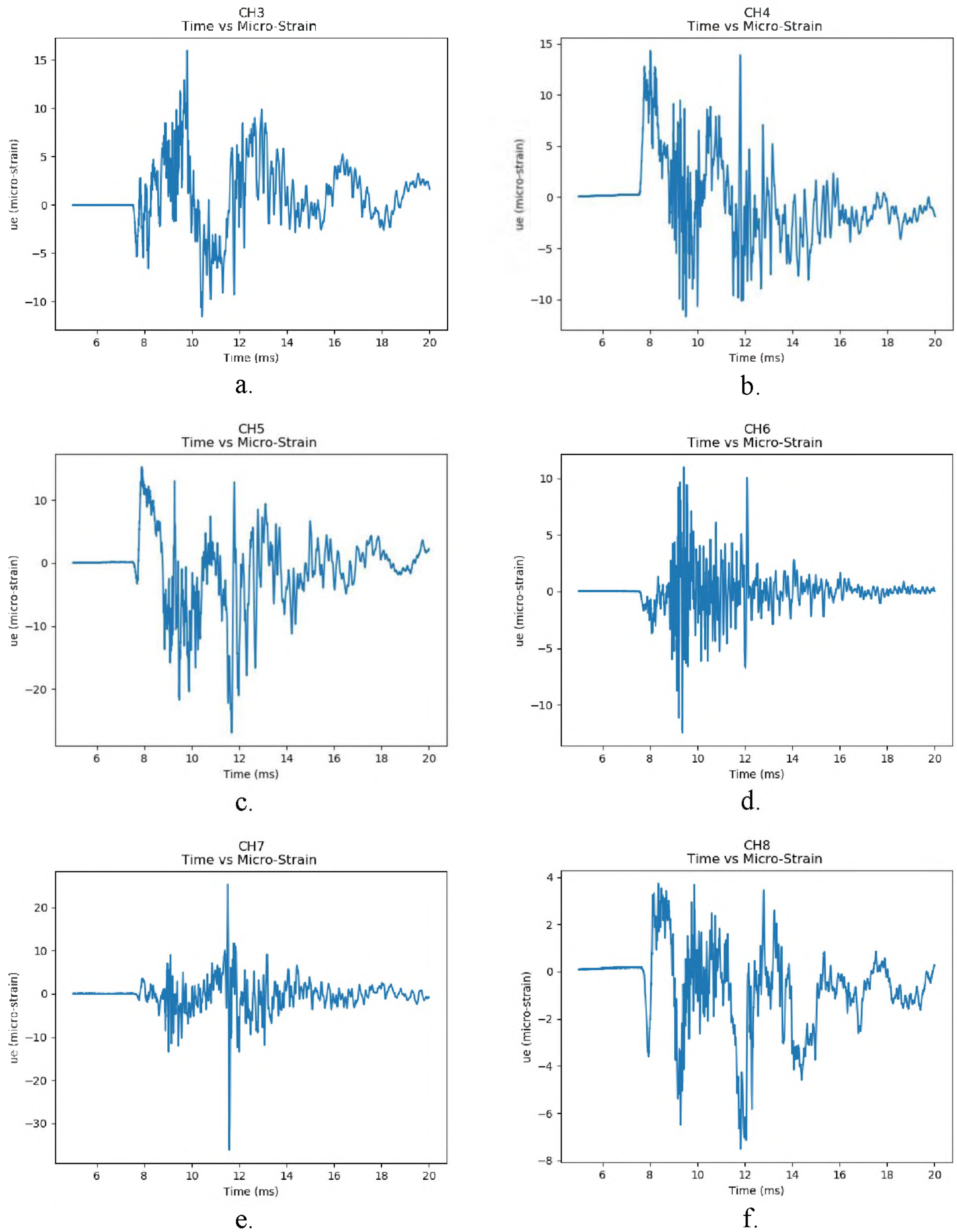
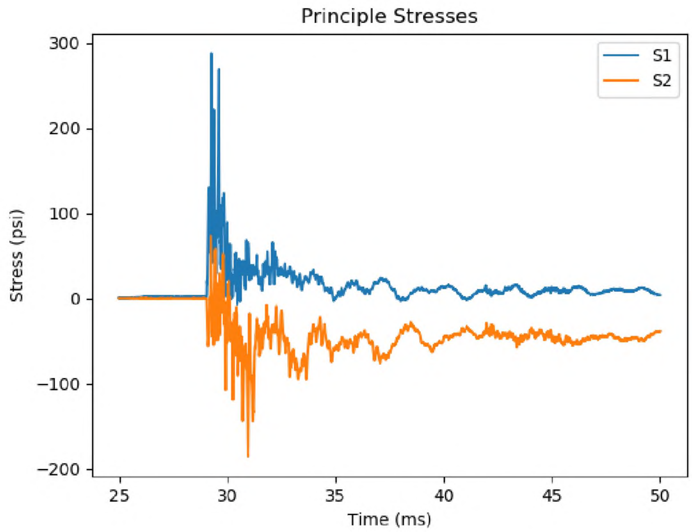
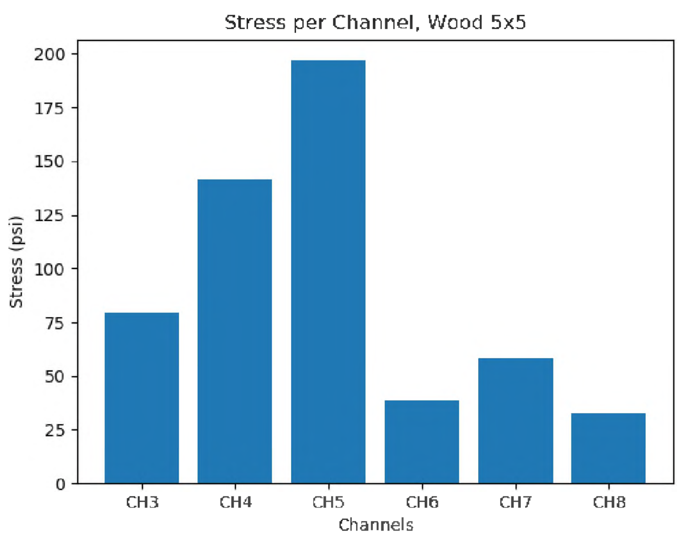


Figure B.16 Tools Data

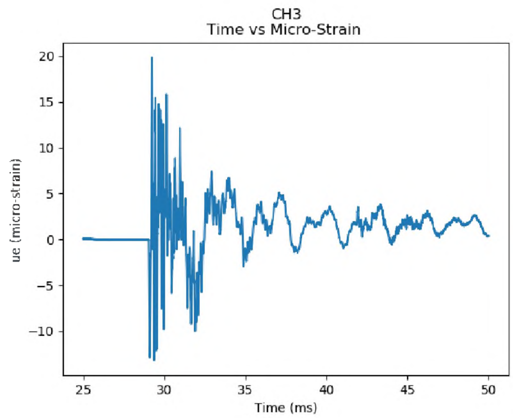


a.

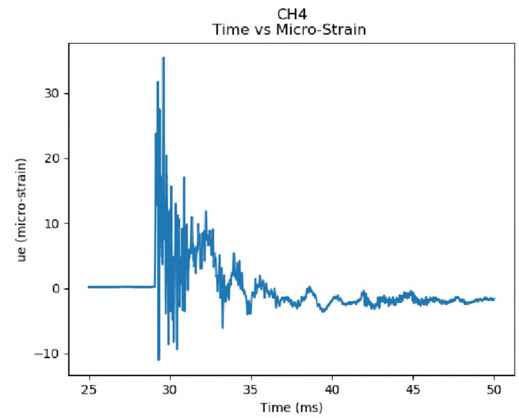


b.

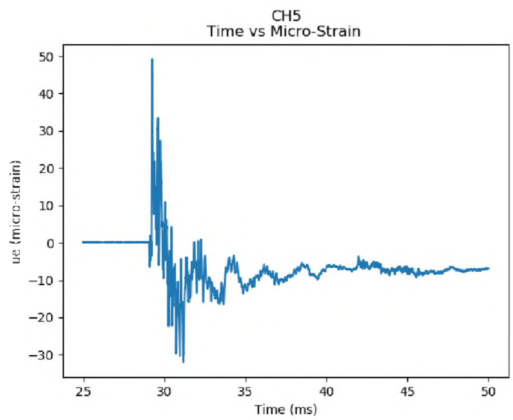
Figure B.17 Wood 5x5 Stresses



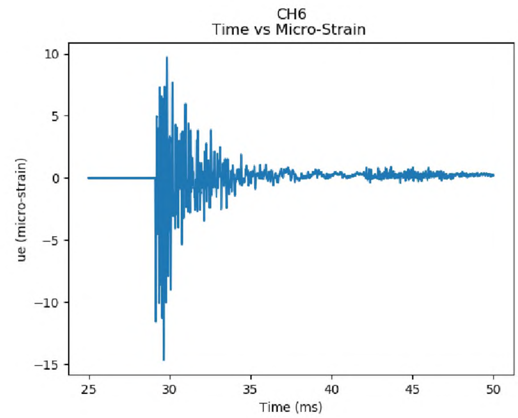
a.



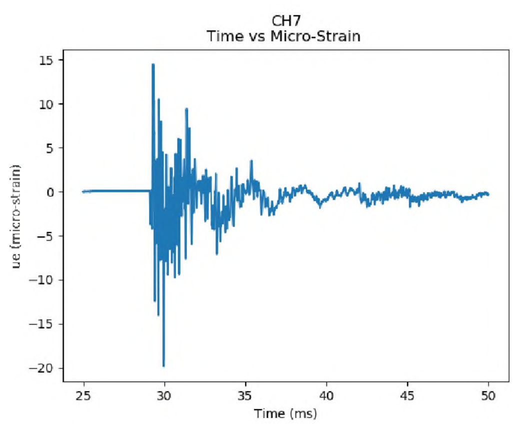
b.



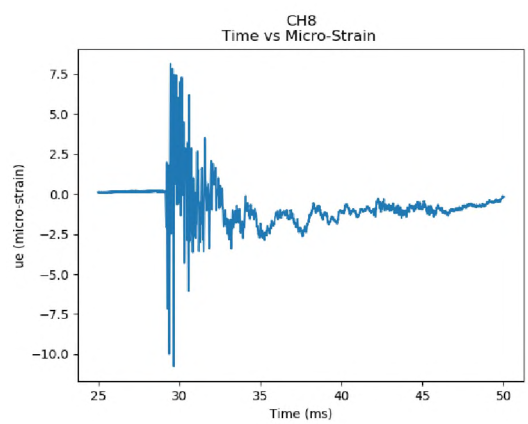
c.



d.



e.



f.

Figure B.18 Wood 5x5 Stress

APPENDIX C.

STRESS ANALYSIS PROGRAM

```
#Project Main Menu and File System Creator
from tkinter import *
from PIL import ImageTk, Image
from tkinter import filedialog
import tkinter as tk
from tkinter import messagebox
import os
import Cannon_CSV_Data_Processor as DataP
import sqlite3

new_filepath = "0"


def new_project():
    gui = Toplevel()
    root.withdraw()
    gui.title("New Project")
    gui.iconbitmap('MST_Icon.ico')
    gui.geometry("+860+400")

    set_location = Button(gui, text="Set Project Location", command=lambda:
get_new_filepath(gui, textbox), bg="black", fg="white")
    textbox = tk.Text(gui, height=1, width=35, bg="white", fg="black")
    file_label = Label(gui, text="Filepath:", fg="black")
    name_label = Label(gui, text="Enter a project name: ")
    E_name = Entry(gui, width=30, borderwidth=5)
    create_project = Button(gui, text="Create Project", command=lambda:
create_project_file(E_name, gui), bg="black", fg="white")
    menu_button = Button(gui, text="Main Menu", command=lambda: destroy(gui),
bg="black", fg="white")


    set_location.grid(row=0, column=0, padx=5, pady=5)
    file_label.grid(row=1, column=0, padx=5, pady=5)
    textbox.grid(row=2, column=0, padx=5, pady=5)
    name_label.grid(row=3, column=0, padx=5, pady=5)
    E_name.grid(row=4, column=0, padx=5, pady=5)
    create_project.grid(row=5, column=0, padx=5, pady=5)
    menu_button.grid(row=6, column=0, padx=5, pady=5)
    #get_button.grid(row=7, column=0)

    name = E_name.get()
    print(name)

    return


def get_new_filepath(gui, textbox):
    global new_filepath
    new_filepath = filedialog.askdirectory(initialdir="/", title="Select File
Location")
    textbox.insert(tk.END, new_filepath)
    gui.update_idletasks()
```

```
    return


def create_project_file(E_name, gui):
    name = E_name.get()
    if name == "":
        messagebox.showwarning("Awww shucks", "Please name the project.")
        return

    if new_filepath == "0":
        messagebox.showwarning("Awww shucks", "Please set new project
location.")
        return

    project_file = new_filepath + "/" + name + " Project"

    try:
        os.mkdir(project_file)

    except FileExistsError:
        messagebox.showwarning("Nuts!", "File name already exists.")
        return

    database = sqlite3.connect(project_file + "/" + name + " Project" +
"_Database.db")

    r = database.cursor()

    r.execute("""CREATE TABLE shot (
                    Shot_Date text,
                    Projectile_Material text,
                    Projectile_Weight_lbs text,
                    Powder_Weight_lbs text,
                    Kinetic_energy_ft_lbs text,
                    Avg_Velocity_ft_s text,
                    Max_Stress_psi text,
                    Max_Axial_Stress_psi text,
                    shot_os_directory text
                )""")

    database.commit()
    database.close()

    messagebox.showinfo("Msg", "New project created at " + new_filepath +
".\n\nPlease return to Main Menu to select "
                                                                          "the
project.")

    return


def current_project():
    filepath = filedialog.askdirectory(initialdir="/", title="Select project
folder")
```

```
    if filepath == "":
        messagebox.showwarning("Warning", "No file path selected")
        return

    else:
        select_create_test(filepath)

    return


def select_create_test(filepath):
    gui = Toplevel()
    root.withdraw()
    gui.title("Current Project")
    gui.iconbitmap('MST_Icon.ico')
    gui.geometry("+860+400")

    last_file_pos = filepath.rfind("/", 0)
    last_file = filepath[last_file_pos + 1:]

    project_name = Label(gui, text="Project: " + last_file, fg="black")
    old_test = Button(gui, text="Analyze Test Data", command=lambda:
select_test(filepath),
                      bg="black", fg="white")
    new_test = Button(gui, text="Make New Test Folder", command=lambda:
name_new_test_folder(filepath), bg="black", fg="white")

    menu_button = Button(gui, text="Main Menu", command=lambda: destroy(gui),
bg="black", fg="white")

    project_name.grid(row=0, column=0, padx=5, pady=5)
    old_test.grid(row=1, column=0, padx=5, pady=5)
    new_test.grid(row=2, column=0, padx=5, pady=5)
    menu_button.grid(row=6, column=0, padx=5, pady=5)

    return


def select_test(filepath):
    test_filepath = filedialog.askdirectory(initialdir=filepath, title="Select
test folder/date")
    DataP.Analysis_gui(test_filepath)

    return


def name_new_test_folder(filepath):
    gui = Toplevel()
    gui.title("New Test Folder")
    gui.iconbitmap('MST_Icon.ico')
    gui.geometry("+860+400")

    month_list = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sept", "Oct", "Nov", "Dec"]
```

```
    click1 = StringVar()
    click1.set(month_list[0])

    month_label = Label(gui, text="Enter the month:")
    month_drop = OptionMenu(gui, click1, *month_list)
    select_day = Label(gui, text="Enter the day (1-31):")
    E_day = Entry(gui, width=20, borderwidth=5)
    description_label = Label(gui, text="Enter test descriptor:")
    E_description = Entry(gui, width=20, borderwidth=5)
    go_back = Button(gui, text="Create Folder", command=lambda:
make_new_folder(gui, filepath, click1, E_day,

E_description), fg="white", bg="black")

    month_label.grid(row=1, column=0, padx=5, pady=5)
    month_drop.grid(row=2, column=0, padx=5, pady=5)
    select_day.grid(row=3, column=0, padx=5, pady=5)
    E_day.grid(row=4, column=0, padx=5, pady=5)
    description_label.grid(row=5, column=0, padx=5, pady=5)
    E_description.grid(row=6, column=0, padx=5, pady=5)
    go_back.grid(row=7, column=0, padx=5, pady=5)

    return


def make_new_folder(gui, filepath, click1, E_day, E_description):

    day = E_day.get()
    month = click1.get()
    descriptor = E_description.get()

    test_file = filepath + "/" + day + "_" + month + "_" + descriptor

    os.mkdir(test_file)
    gui.destroy()

    return



def destroy(x):
    x.destroy()
    root.deiconify()
    return
def title_window():
    root.destroy()
    return


#Title Window
root = tk.Tk()
root.iconbitmap('MST_Icon.ico')
root.title("Cannon Data Processor")
root.configure(background='green')
```

```
root.geometry("+650+300")
load = Image.open('Cannon_Shot.jpg')
render = ImageTk.PhotoImage(load)
img = Label(root, image=render)
img.grid(row=0, column=0)
label = Label(root, text="Welcome to the MST Cannon Data Processor",
bg='green', fg='white')
label.grid(row=1, column=0)
root.after(2000, title_window)
root.mainloop()


root = tk.Tk()
root.title("Cannon Data Processor")
root.iconbitmap('MST_Icon.ico')
root.geometry("+800+300")

Menu_frame = LabelFrame(root)
Title_label = Label(Menu_frame, text="Main Menu", fg="green", bg="white")
new_project_button = tk.Button(Menu_frame, text="Create New Project",
command=new_project, fg="white", bg="black")
create_file_button = tk.Button(Menu_frame, text="Select Project",
command=current_project, fg="white", bg="black")
exit_button = Button(Menu_frame, text="Exit Program", command=root.quit,
bg="black", fg="white")

load = Image.open('Chrono_Resize.jpg')
load = load.resize((300, 200), Image.ANTIALIAS)
render = ImageTk.PhotoImage(load)
img = Label(root, image=render)

Menu_frame.grid(row=0, column=0, padx=5, pady=5)
Title_label.grid(row=0, column=0, padx=5, pady=15)
create_file_button.grid(row=1, column=0, padx=5, pady=5)
new_project_button.grid(row=2, column=0, padx=5, pady=5)
exit_button.grid(row=3, column=0, padx=5, pady=5)
img.grid(row=4, column=0, padx=5, pady=5)

mainloop()
```

```
#This code is the data processor module "Cannon_CSV_Data_Processor.py" for
#Project_Main_Menu.py
#This program was written by Ethan Steward for the Alpha Foundation project:
#Analysis of Coal Mine Seal Integrity from Explosively Driven Projectiles
print("Imported Cannon_CSV_Data_Processor.py")
from tkinter import *
from tkinter import filedialog
import tkinter as tk
#from PIL import ImageTk, Image
import csv
import math
from matplotlib import pyplot as plt
import os
from tkinter import messagebox
import numpy as np
import sympy as sp
from sympy import Eq, solve_linear_system, Matrix
import math
import sqlite3


filepath = "0"
main_file = "0"


def Analysis_gui(test_date_filepath):
    #Main Window
    io_gui = tk.Tk()
    io_gui.title("Cannon Data Processor")
    io_gui.iconbitmap('MST_Icon.ico')
    io_gui.geometry("+650+300")

    Output_frame = LabelFrame(io_gui, text="Output:")
    Entry_frame = LabelFrame(io_gui, text="Input:")
    Folder_frame = LabelFrame(Entry_frame)

    dist_label = Label(Entry_frame, text="Enter IR sensor separation distance
(in):", padx=5, pady=5)
    E_dist = Entry(Entry_frame, width=20, borderwidth=5)

    mass_label = Label(Entry_frame, text="Enter test projectile mass (lbs):")
    E_mass = Entry(Entry_frame, width=20, borderwidth=5)

    sum_mass_label = Label(Entry_frame, text="Enter shot weight (include all
projectile components):")
    E_sum_mass = Entry(Entry_frame, width=20, borderwidth=5)

    powder_weight_label = Label(Entry_frame, text="Enter powder charge weight
(lbs):", padx=5, pady=5)
    E_powder_weight = Entry(Entry_frame, width=20, borderwidth=5)

    x_y_position = Label(Entry_frame, text="Enter hit location (in, x/y) from
bottom left:")
    E_x= Entry(Entry_frame, width=10, borderwidth=5)
```

```
    E_y=Entry(Entry_frame, width=10, borderwidth=5)

    new_shot_button = tk.Button(Folder_frame, text="New Shot Folder",
                                command=lambda:
make_new_shot_folder(test_date_filepath), fg="white", bg="black")

    filepath_button = tk.Button(Folder_frame, text="Select Shot",
command=lambda: findfilepath(test_date_filepath),
                                fg="white", bg="black")

    material_list = ["steel", "concrete", "wood", "rock", "other"]

    material = tk.StringVar(Entry_frame)
    material.set(material_list[0])

    material_label = Label(Entry_frame, text="Select projectile material:")
    material_drop = OptionMenu(Entry_frame, material, *material_list)
    material_drop.config(width=6)

    target_label = Label(Entry_frame, text="Describe Target: ")
    E_target = Entry(Entry_frame, width=20, borderwidth=5)

    textbox = tk.Text(Output_frame, height=30, width=35, padx=5, pady=5,
bg="black", fg="#49fe00")

    analyze_button = tk.Button(Entry_frame, text="Run Analysis",
command=lambda: analyze_data(test_date_filepath,

textbox, Output_frame, E_dist, E_mass,

E_sum_mass, E_powder_weight, E_x, E_y,

material, E_target), fg="white",

bg="black")

    back_button = Button(io_gui, text="Back", command=lambda: go_back(io_gui),
bg="black", fg="white")
    exit_button = Button(io_gui, text="Exit Program", command=io_gui.quit,
bg="black", fg="white")

    Output_frame.grid(row=0, column=1, padx=5, pady=5)
    Entry_frame.grid(row=0, column=0, padx=5, pady=5)

    Folder_frame.grid(row=1, column=0, padx=1, pady=1)

    new_shot_button.grid(row=0, column=0, padx=5, pady=1)
    filepath_button.grid(row=0, column=1, padx=5, pady=1)

    dist_label.grid(row=2, column=0)
    E_dist.grid(row=3, column=0, padx=10, pady=10)

    mass_label.grid(row=4, column=0)
    E_mass.grid(row=5, column=0, padx=10, pady=10)
```

```
    sum_mass_label.grid(row=6, column=0)
    E_sum_mass.grid(row=7, column=0, padx=10, pady=10)

    powder_weight_label.grid(row=8, column=0)
    E_powder_weight.grid(row=9, column=0)

    x_y_position.grid(row=10, column=0, padx=10, pady=10)
    E_x.grid(row=11, column=0, padx=4, pady=4)
    E_y.grid(row=12, column=0, padx=4, pady=4)

    material_label.grid(row=13, column=0, padx=5, pady=5)
    material_drop.grid(row=14, column=0)

    target_label.grid(row=15, column=0, pady=5)
    E_target.grid(row=16, column=0, pady=5)

    analyze_button.grid(row=17, column=0, pady=5)

    textbox.grid(row=0, column=1, padx=5, pady=5)

    back_button.grid(row=1, column=0)
    exit_button.grid(row=1, column=1)


    textbox.insert(tk.END, "Program Running...")

    return


def analyze_data(test_date_filepath, textbox, Output_frame, E_dist, E_mass,
E_sum_mass, E_powder_weight, E_x, E_y,
                 material, E_target):

    if filepath == "0":
        textbox.insert(tk.END, "\nFile path not set")
        return

    x = 1
    CH1_time = CH1_2_function(x)
    textbox.insert(tk.END, "\nCH1 file found")
    Output_frame.update_idletasks()

    x = 2
    CH2_time = CH1_2_function(x)
    textbox.insert(tk.END, "\nCH2 file found")
    Output_frame.update_idletasks()

    x = 3
    c = get_sensor_conversion(x, textbox)
    CH3data = CH3_8_function(x, c)
    textbox.insert(tk.END, "\nCH3 file found")
    Output_frame.update_idletasks()
```

```
x = 4
c = get_sensor_conversion(x, textbox)
CH4data = CH3_8_function(x, c)
textbox.insert(tk.END, "\nCH4 file found")
Output_frame.update_idletasks()

x = 5
c = get_sensor_conversion(x, textbox)
CH5data = CH3_8_function(x, c)
textbox.insert(tk.END, "\nCH5 file found")
Output_frame.update_idletasks()

x = 6
c = get_sensor_conversion(x, textbox)
CH6data = CH3_8_function(x, c)
textbox.insert(tk.END, "\nCH6 file found")
Output_frame.update_idletasks()

x = 7
c = get_sensor_conversion(x, textbox)
CH7data = CH3_8_function(x, c)
textbox.insert(tk.END, "\nCH7 file found")
Output_frame.update_idletasks()

x = 8
c = get_sensor_conversion(x, textbox)
CH8data = CH3_8_function(x, c)
textbox.insert(tk.END, "\nCH8 file found")
textbox.insert(tk.END, "\nGraphs created")
Output_frame.update_idletasks()

Time_Diff = CH2_time - CH1_time

dist = E_dist.get()
mass = E_mass.get()
dist = float(dist)
mass = float(mass)

sum_mass = E_sum_mass.get()
sum_mass = float(sum_mass)

Avg_vel = float((dist / Time_Diff) * (1 / 12) * (1000))
KE_imp = 0.5 * (mass / 32.174) * Avg_vel ** 2
KE_met = KE_imp * 1.356
sum_KE_imp = 0.5 * (sum_mass / 32.174) * Avg_vel ** 2
sum_KE_met = sum_KE_imp * 1.356
Avg_vel = str(round(Avg_vel, 2))
KE_imp = str(round(KE_imp, 2))
KE_met = str(round(KE_met, 2))
sum_KE_imp = str(round(sum_KE_imp, 2))
sum_KE_met = str(round(sum_KE_met, 2))
mass = str(round(mass, 2))
sum_mass = str(round(sum_mass, 2))
```

```
    #Note that channel 3 is always sensor 1, CH4 = Sens2, etc.
    CH3max = find_strain_max(CH3data)
    CH4max = find_strain_max(CH4data)
    CH5max = find_strain_max(CH5data)
    CH6max = find_strain_max(CH6data)
    CH7max = find_strain_max(CH7data)
    CH8max = find_strain_max(CH8data)

    CH3min = find_strain_min(CH3data)
    CH4min = find_strain_min(CH4data)
    CH5min = find_strain_min(CH5data)
    CH6min = find_strain_min(CH6data)
    CH7min = find_strain_min(CH7data)
    CH8min = find_strain_min(CH8data)

    CH3_xyz = get_sensor_location(1, textbox, Output_frame)
    CH4_xyz = get_sensor_location(2, textbox, Output_frame)
    CH5_xyz = get_sensor_location(3, textbox, Output_frame)
    CH6_xyz = get_sensor_location(4, textbox, Output_frame)
    CH7_xyz = get_sensor_location(5, textbox, Output_frame)
    CH8_xyz = get_sensor_location(6, textbox, Output_frame)

    CH3_dist = find_sensor_distance(CH3_xyz, E_x, E_y)
    CH4_dist = find_sensor_distance(CH4_xyz, E_x, E_y)
    CH5_dist = find_sensor_distance(CH5_xyz, E_x, E_y)
    CH6_dist = find_sensor_distance(CH6_xyz, E_x, E_y)
    CH7_dist = find_sensor_distance(CH7_xyz, E_x, E_y)
    CH8_dist = find_sensor_distance(CH8_xyz, E_x, E_y)

    CH3_stress = find_simple_stress(CH3max, CH3min, textbox)
    CH4_stress = find_simple_stress(CH4max, CH4min, textbox)
    CH5_stress = find_simple_stress(CH5max, CH5min, textbox)
    CH6_stress = find_simple_stress(CH6max, CH6min, textbox)
    CH7_stress = find_simple_stress(CH7max, CH7min, textbox)
    CH8_stress = find_simple_stress(CH8max, CH8min, textbox)

    #finds order of sensors from closest to furthest from impact site
    dist_list = [CH3_dist, CH4_dist, CH5_dist, CH6_dist, CH7_dist, CH8_dist]
    stress_list = [CH3_stress, CH4_stress, CH5_stress, CH6_stress, CH7_stress,
CH8_stress]

    dist_ord = sort_dist(dist_list)
    stress_ord = sort_stress(stress_list)

    max_pos = stress_list.index(max(stress_list))
    #stress channels start at channel 3, hence max_pos + 3
    max_CH = max_pos + 3
    max_CH = str(max_CH)
    max_stress = max(stress_list)
    max_stress = str(max_stress)
    max_list = []
    max_list.insert(0, max_CH)
    max_list.insert(1, max_stress)
```

```
    powder_weight = E_powder_weight.get()
    material = material.get()
    target = E_target.get()

    data_list = [[Avg_vel, KE_imp, KE_met, sum_KE_imp, sum_KE_met,
powder_weight, mass, sum_mass, material, target],
                 [CH3max, CH3min, CH3_dist, CH3_stress], [CH4max, CH4min,
CH4_dist, CH4_stress],
                 [CH5max, CH5min, CH5_dist, CH5_stress], [CH6max, CH6min,
CH6_dist, CH6_stress],
                 [CH7max, CH7min, CH7_dist, CH7_stress], [CH8max, CH8min,
CH8_dist, CH8_stress]]

    create_basic_charts(stress_list)

    rosette_stress_data = Full_Rosette_Scan(textbox, Output_frame)
    stress_data = rosette_stress_data[2]
    write_rosette_graphs(stress_data)

    max_axial = [rosette_stress_data[0], rosette_stress_data[1]]

    write_output_files(test_date_filepath, data_list, dist_ord, textbox,
max_list, stress_ord, max_axial)

    last_file_pos = filepath.rfind("/", 0)
    last_file = filepath[last_file_pos + 1:]
    textbox.insert(tk.END, "\n\nAll values calculated\n\nWrote output file: "
+ last_file + " Output.txt"
                            "\n\nData sent to database\n\nAnalysis complete")

    return


def CH1_2_function(f):
    f = str(f)

    file = "/CH" + f + ".txt"

    with open(filepath + file, 'r') as csv_file:
        file_read = csv.reader(csv_file, delimiter=',')

        xdata = []
        ydata = []

        for row in file_read:
            x = row[0]
            y = row[1]

            xdata.append(x)
            ydata.append(y)

    for i in range(0, len(xdata)):

        xdata[i] = float(xdata[i])
```

```
    for i in range(0, len(ydata)):

        ydata[i] = float(ydata[i])

    file_data = list(zip(xdata, ydata))

    i = 0
    z = 49
    baseline = 0

    #establishes the normal sensor signal from a # of datapoints starting at
    # the beginning of the data. Var avgsig represents the average unactivated
sensor signal
    while i <= z:
        baseline = baseline + file_data[i][1]
        i += 1

    #average signal found from average of z datapoints
    avgsig = baseline/(z+1)

    r = 0
    z = 0
    #USER: set voltage_variance (voltage_var) just above maximum noise level
of sensor.
    voltage_var = .15
    file_time = 0
    volts = 0

    #loop compares all datapoints in y (2nd) column (voltage) against the
average sensor
    # signal IOT to determine when sensor voltage drops (infrared beam broken
by projectile)
    while r <= len(file_data) - 1:

        z = file_data[r][1] - avgsig
        z = abs(z)
        r += 1

        #if voltage variance is found above voltage_var variable, records time
        if z > voltage_var:
            file_time = file_data[r][0]
            volts = file_data[r][1]
            break

        if r == len(file_data) - 1:
            print("CH" + f + ".txt not found")
            break

    return file_time


def CH3_8_function(f, c):
```

```
    f = str(f)

    file = "/CH" + f + ".txt"
    #convert c from mv/ue to ue
    c = (1000/c)

    with open(filepath + file, 'r') as csv_file:
        file_read = csv.reader(csv_file, delimiter=',')

        xdata = []
        ydata = []

        for row in file_read:
            x = row[0]
            y = row[1]

            xdata.append(x)
            ydata.append(y)

    #Converts data from string to float
    for i in range(0, len(xdata)):
        xdata[i] = float(xdata[i])

    #Converts ydata (volts) to microstrain (ue)
    for i in range(0, len(ydata)):
        ydata[i] = float(ydata[i])
        ydata[i] = ydata[i] * c

    write_graphs(xdata, ydata, f)

    file_data = list(zip(xdata, ydata))

    return file_data


def create_basic_charts(stress_list):
    channels = ["CH3", "CH4", "CH5", "CH6", "CH7", "CH8"]

    plt.ioff()
    plt.clf()
    last_file_pos = filepath.rfind("/", 0)
    last_file = filepath[last_file_pos + 1:]

    plt.bar(channels, stress_list)
    plt.xlabel('Channels')
    plt.ylabel('Stress (psi)')
    plt.title('Stress per Channel, ' + last_file)
    plt.savefig(filepath + '/Stress per Channel ' + last_file + '.png')


    return


def findfilepath(test_date_filepath):
```

```
    global filepath
    global main_file
    filepath = filedialog.askdirectory(initialdir=test_date_filepath,
title="Select File")
    print(filepath)
    last_file_pos = filepath.rfind("/", 0)
    main_file = filepath[0:last_file_pos]
    print(main_file)
    print(test_date_filepath)
    messagebox.showinfo("Input Data", "Copy and paste the CSV files (8) into
the new shot folder.\nPlease ensure that"
                                      " the setup file for this test date is
inside the test folder.")

    return


def find_sensor_distance(pos, E_x, E_y):
    x_pos = E_x.get()
    y_pos = E_y.get()

    x_pos = float(x_pos)
    y_pos = float(y_pos)

    dist = math.sqrt((pos[0]-x_pos)**2 + (pos[1]-y_pos)**2 + (pos[2]-0)**2)
    dist = round(dist, 1)

    return dist


def find_strain_max(data):
    b = data[0][1]

    for i in range(0, len(data)):
        if b < data[i][1]:
            b = data[i][1]

    return b


def find_strain_min(data):
    b = data[0][1]

    for i in range(0, len(data)):
        if b > data[i][1]:
            b = data[i][1]

    return b


def find_times(a_data, b_data, c_data):

    max_start = max(a_data[0][0], b_data[0][0], c_data[0][0])
    ms = str(max_start)
```

```
    print("max start time: " + ms + " ms")

    min_end = min(a_data[-1][0], b_data[-1][0], c_data[-1][0])
    me = str(min_end)
    print("min end time: " + me + " ms")

    min_max_times = [max_start, min_end]

    return min_max_times
def index_2d(list, v):
    for i, x in enumerate(list):
        if v in x:
            return i, x.index(v)


def Full_Rosette_Scan(textbox, Output_frame):
    x = 3
    c = get_sensor_conversion(x, textbox)
    a_data = CH3_8_function(x, c)

    x = 4
    c = get_sensor_conversion(x, textbox)
    b_data = CH3_8_function(x, c)

    x = 5
    c = get_sensor_conversion(x, textbox)
    c_data = CH3_8_function(x, c)

    stress_para = get_stress_parameters(textbox)
    Tavg = stress_para[0]
    ModEavg = stress_para[1]
    PR = stress_para[2]

    angles = get_rose_angles()

    ta = angles[0]
    tb = angles[1]
    tc = angles[2]

    min_max_times = find_times(a_data, b_data, c_data)

    #find (row, column) of start time
    a_start = index_2d(a_data, min_max_times[0])
    b_start = index_2d(b_data, min_max_times[0])
    c_start = index_2d(c_data, min_max_times[0])
    #find row of start time
    a_row_start = a_start[0]
    b_row_start = b_start[0]
    c_row_start = c_start[0]


    a_end = index_2d(a_data, min_max_times[1])
    b_end = index_2d(b_data, min_max_times[1])
    c_end = index_2d(c_data, min_max_times[1])
```

```
    a_row_end = a_end[0]
    b_row_end = b_end[0]
    c_row_end = c_end[0]

    data_range = a_row_end-a_row_start
    dr = str(data_range)

    axial_strain_list = [0, 0, 0, 0]
    axial_stress_list = [0, 0, 0, 0]
    Stress_S1 = []
    Stress_S2 = []
    Stress_Axial = []
    time = []

    textbox.insert(tk.END, "\nRosette Scan Start.")
    textbox.insert(tk.END, "\nAnalysis may take several minutes. \nPLEASE
STAND BY...")
    Output_frame.update_idletasks()

    a1 = 0
    a2 = 0
    a3 = 0

    for i in range(0, data_range):
        # This loop uses sympy to solve the 3 equations for strain in x, y,
and shear (v)
        a = a_data[a_row_start + i][1]
        b = b_data[b_row_start + i][1]
        c = c_data[c_row_start + i][1]

        if i > 0.05 * data_range and a1 == 0:
            print("Progress: 5%")
            a1 = 1

        if i > 0.50 * data_range and a2 == 0:
            print("Progress: 50%")
            a2 = 1

        if i > 0.75 * data_range and a3 == 0:
            print("Progress: 75%")
            a3 = 1

        s = a / (math.cos(math.radians(ta)) ** 2)
        s = round(s, 4)
        d = (math.sin(math.radians(ta)) ** 2) / (math.cos(math.radians(ta)) **
2)
        d = round(d, 4)
        f = (math.sin(math.radians(ta)) * math.cos(math.radians(ta))) /
(math.cos(math.radians(ta)) ** 2)
        f = round(f, 4)

        g = b / (math.sin(math.radians(tb)) ** 2)
        g = round(g, 4)
        h = (math.cos(math.radians(tb)) ** 2) / (math.sin(math.radians(tb)) **
```

```
2)
        h = round(h, 4)
        j = (math.sin(math.radians(tb)) * math.cos(math.radians(tb))) /
(math.sin(math.radians(tb)) ** 2)
        j = round(j, 4)

        k = c / (math.sin(math.radians(tc)) * math.cos(math.radians(tc)))
        k = round(k, 4)
        n = (math.cos(math.radians(tc)) ** 2) / (math.sin(math.radians(tc)) *
math.cos(math.radians(tc)))
        n = round(n, 4)
        m = (math.sin(math.radians(tc)) ** 2) / (math.sin(math.radians(tc)) *
math.cos(math.radians(tc)))
        m = round(m, 4)

        eq1 = sp.Function('eq1')
        eq2 = sp.Function('eq2')
        eq3 = sp.Function('eq3')

        x, y, v = sp.symbols('x y v')

        eq1 = Eq(x + y * d + v * f, s)
        eq2 = Eq(y + x * h + v * j, g)
        eq3 = Eq(v + x * n + y * m, k)

        row1 = [1, d, f, s]
        row2 = [1, h, j, g]
        row3 = [1, n, m, k]

        system = Matrix((row1, row2, row3))
        sol = (solve_linear_system(system, x, y, v))
        Ex = sol[x]
        Ex = round(Ex, 2)
        Ey = sol[y]
        Ey = round(Ey, 2)
        Yxy = sol[v]
        Yxy = round(Yxy, 2)
        Sx = (Ex * 10 ** (-6)) * ModEavg
        Sy = (Ey * 10 ** (-6)) * ModEavg
        Sxy = (Yxy * 10 ** (-6)) * ModEavg
        Savg = (Sx + Sy) / 2

        E1 = ((Ex + Ey) / 2) + math.sqrt((((Ex - Ey) / 2) ** 2) + (Yxy / 2) **
2)
        E2 = ((Ex + Ey) / 2) - math.sqrt((((Ex - Ey) / 2) ** 2) + (Yxy / 2) **
2)
        E1 = round(E1, 2)
        E2 = round(E2, 2)
        S1 = (E1 * 10 ** (-6)) * ModEavg
        S2 = (E2 * 10 ** (-6)) * ModEavg

        Emin = min(E1, E2)
        Ez = -(Emin) / PR
        Ez = round(Ez, 2)
```

```
        Sz = (Ez * (10 ** (-6))) * ModEavg

        Stress_S1.append(S1)
        Stress_S2.append(S2)
        Stress_Axial.append(Sz)
        time.append(a_data[a_row_start + i][0])

        Avg_Strain = (Ex + Ey) / 2
        Avg_Stress = (Avg_Strain * (10 ** (-6))) * ModEavg

        # Theta = (math.degrees(math.atan(Yxy / (Ex - Ey)))) / 2
        # Theta = round(Theta, 2)

        # # max shear strain
        # Y = (math.sqrt((((Ex - Ey) / 2) ** 2) + ((Yxy / 2) ** 2))) * 2
        # Ystress = stress_para[1] * (Y * 10 ** (-6))

        if E1 > axial_strain_list[0]:
            axial_strain_list[0] = E1
            axial_strain_list[1] = E2
            axial_strain_list[2] = Ez
            axial_strain_list[3] = Avg_Strain

            axial_stress_list[0] = S1
            axial_stress_list[1] = S2
            axial_stress_list[2] = Sz
            axial_stress_list[3] = Avg_Stress

    stress_data = [Stress_S1, Stress_S2, Stress_Axial, time]

    # textbox.insert(tk.END, "\nRosette Data Scan Complete")
    # Output_frame.update_idletasks()
    return axial_strain_list, axial_stress_list, stress_data


def find_simple_stress(strain_max, strain_min, textbox):

    stress_para = get_stress_parameters(textbox)

    x = strain_max
    x = float(x)
    #convert strain to stress
    stress = stress_para[1] * (x * 10**(-6))
    stress = round(stress, 2)

    return stress


def get_rose_angles():

    Setup_file = open("Setup.txt", "r")
    Setup_string = Setup_file.read()
    Setup_file.close()
```

```
    z = Setup_string.find("Theta a:")
    x = Setup_string.find(',', z)
    y = len("Theta a:")
    a = Setup_string[(z + y):x]

    z = Setup_string.find("Theta b:")
    x = Setup_string.find(',', z)
    y = len("Theta b:")
    b = Setup_string[(z + y):x]

    z = Setup_string.find("Theta c:")
    x = Setup_string.find(',', z)
    y = len("Theta c:")
    c = Setup_string[(z + y):x]

    a = float(a)
    b = float(b)
    c = float(c)

    angles = [a, b, c]

    return angles


def get_sensor_location(sensor, textbox, Output_frame):
    #This function attempts to open the setup file, which stores the location
data of the sensors and other data
    #If Setup.txt is found, function gets the thickness of the plug and the
x/y location of the sensors
    try:
        Setup_file = open(main_file + "/Setup.txt", "r")
        if sensor == 1:
            textbox.insert(tk.END, "\nSetup.txt found")
            Output_frame.update_idletasks()

    except FileNotFoundError:
        textbox.insert(tk.END, "\nSetup.txt not found")
        return

    try:
        Setup_file = open(main_file + "/Setup.txt", "r")
        Setup_string = Setup_file.read()
        Setup_file.close()
        Thick_loc = Setup_string.find("\"")
        Thick = Setup_string[(Thick_loc + 1):(Thick_loc + 5)]
        Thick = float(Thick)

        Sensor_loc = []

        sensor = str(sensor)
        current_sensor = ("Sensor " + sensor)
        loc_index = Setup_string.find(current_sensor)
        x = Setup_string[(loc_index + 11):(loc_index + 16)]
        y = Setup_string[(loc_index + 18):(loc_index + 23)]
```

```
            x = float(x)
            y = float(y)
            Sensor_loc.insert(0, x)
            Sensor_loc.insert(1, y)
            Sensor_loc.insert(2, Thick)

    except FileNotFoundError:
        textbox.insert(tk.END, "\nLocation data not found")
        return

    return Sensor_loc


def get_sensor_conversion(Channel, textbox):
    #function finds the sensor on the given channel in the setup file, then
finds the conversion ratio of that sensor
    #returns the conversion ratio as mV/microstrain
    try:
        Channel = str(Channel)
        Setup_file = open(main_file + "/Setup.txt", "r")
        Setup_string = Setup_file.read()
        Setup_file.close()
        CH = "CH" + Channel
        z = Setup_string.find(CH)
        x = Setup_string[(z + 4):(z + 8)]
        Conversion_Index = Setup_string.find("SN" + x)
        c = Setup_string[(Conversion_Index + 7):(Conversion_Index + 11)]
        c = float(c)

    except FileNotFoundError:
        textbox.insert(tk.END, "\nConversion data not found")
        return

    return c


def get_stress_parameters(textbox):

    try:
        Setup_file = open(main_file + "/Setup.txt", "r")
        Setup_string = Setup_file.read()
        Setup_file.close()

        # z = Setup_string.find("Tensile max:")
        # x = len("Tensile max:")
        # Tmax = Setup_string[(z + x):(z + x + 5)]
        #
        # z = Setup_string.find("Tensile min:")
        # x = len("Tensile min:")
        # Tmin = Setup_string[(z + x):(z + x + 5)]
        #
        # z = Setup_string.find("Mod of Elas max:")
        # x = len("Mod of Elas max:")
        # ModEmax = Setup_string[(z + x):(z + x + 9)]
```

```
        #
        # z = Setup_string.find("Mod of Elas min:")
        # x = len("Mod of Elas min:")
        # ModEmin = Setup_string[(z + x):(z + x + 9)]

        #gets avg tensile strength and modulus of elasticity from Setup.txt
        z = Setup_string.find("Tensile avg:")
        x = Setup_string.find(',', z)
        y = len("Tensile avg:")
        Tavg = Setup_string[(z + y):x]

        z = Setup_string.find("Mod of Elas avg:")
        x = Setup_string.find(',', z)
        y = len("Mod of Elas avg:")
        ModEavg = Setup_string[(z + y):x]

        z = Setup_string.find("Poisson's Ratio avg:")
        x = Setup_string.find(',', z)
        y = len("Poisson's Ratio avg:")
        PR = Setup_string[(z + y):x]

        Tavg = float(Tavg)
        ModEavg = float(ModEavg)
        PR = float(PR)

        stress_para = [Tavg, ModEavg, PR]

    except FileNotFoundError:
        textbox.insert(tk.END, "\nStress data not found")
        return

    return stress_para


def get_test_date(textbox):

    try:
        Setup_file = open(main_file + "/Setup.txt", "r")
        Setup_string = Setup_file.read()
        Setup_file.close()
        date_loc = Setup_string.find("Test Date:")
        date = Setup_string[(date_loc + 10):(date_loc + 20)]

        return date

    except FileNotFoundError:
        textbox.insert(tk.END, "\nTest Date not found")
        return


def go_back(gui):
    gui.destroy()
    return
```

```
def make_shot_folder(gui, filepath, E_description):

    descriptor = E_description.get()
    os.mkdir(filepath + "/" + descriptor)
    gui.destroy()

    return


def make_new_shot_folder(filepath):
    gui = Toplevel()
    gui.title("New Shot Folder")
    gui.iconbitmap('MST_Icon.ico')
    gui.geometry("+860+400")

    description_label = Label(gui, text="Enter test descriptor:")
    E_description = Entry(gui, width=20, borderwidth=5)
    go_back = Button(gui, text="Create Shot Folder", command=lambda:
make_shot_folder(gui, filepath, E_description),
                     fg="white", bg="black")

    description_label.grid(row=1, column=0, padx=5, pady=5)
    E_description.grid(row=2, column=0, padx=5, pady=5)
    go_back.grid(row=3, column=0, padx=5, pady=5)

    return


def sort_dist(dist_list):

    labeled_dist = [["CH3", 0], ["CH4", 0], ["CH5", 0], ["CH6", 0], ["CH7",
0], ["CH8", 0]]

    for i in range(0, len(dist_list)):
        CH = "CH" + str(i + 3)
        dist = dist_list[i]
        labeled_dist[i][0] = CH
        labeled_dist[i][1] = dist

    labeled_dist.sort(key=lambda x: x[1])

    return labeled_dist


def sort_stress(stress_list):
    labeled_stress = [["CH3", 0], ["CH4", 0], ["CH5", 0], ["CH6", 0], ["CH7",
0], ["CH8", 0]]

    for i in range(0, len(stress_list)):
        CH = "CH" + str(i + 3)
        stress = stress_list[i]
        labeled_stress[i][0] = CH
        labeled_stress[i][1] = stress
```

```
    labeled_stress = sorted(labeled_stress, key=lambda v: v[1], reverse=True)

    return labeled_stress


def write_graphs(xdata, ydata, f):

    plt.ioff()
    plt.clf()
    last_file_pos = filepath.rfind("/", 0)
    last_file = filepath[last_file_pos + 1:]

    plt.plot(xdata, ydata)
    plt.xlabel('Time (ms)')
    plt.ylabel('ue (micro-strain)')
    plt.title('CH' + f + '\nTime vs Micro-Strain ')
    plt.savefig(filepath + '/CH' + f + ' Time vs Micro-Strain_' + last_file +
'.png')

    return


def write_rosette_graphs(RSD):

    plt.ioff()
    plt.clf()
    last_file_pos = filepath.rfind("/", 0)
    last_file = filepath[last_file_pos + 1:]

    S1 = RSD[0]
    S2 = RSD[1]
    Sz = RSD[2]
    time = RSD[3]

    plt.plot(time, S1)
    plt.plot(time, S2)
    #plt.plot(time,  Sz)
    plt.xlabel('Time (ms)')
    plt.ylabel('Stress (psi)')
    plt.title('Principle Stresses')
    plt.legend(['S1', 'S2'])
    plt.savefig(filepath + '/Principle Stresses_' + last_file + '.png')

    return


def write_output_files(test_date_filepath, data_list, dist_ord, textbox,
max_list, stress_ord, max_axial):

    #function finds file name of the last selected file, then writes program
results to "last_file Output.txt"
    last_file_pos = filepath.rfind("/", 0)
    last_file = filepath[last_file_pos+1:]
```

```
    #print(last_file)

    date = get_test_date(textbox)

    Output_file = open(filepath + '/' + last_file + " Output.txt", "w")
    Output_file.write(last_file + " Output File")
    Output_file.write("\nTest Date: " + date)
    Output_file.write("\n\nProjectile Data")
    Output_file.write("\nAverage Impact Velocity: " + data_list[0][0] + "
ft/s")
    Output_file.write("\nKinetic Energy of test projectile: " +
data_list[0][1] + " ft-lbs, "
                          + data_list[0][2] + " Joules")
    Output_file.write("\nSum KE of projectile, wad, and sabot: " +
data_list[0][3] + " ft-lbs, "
                          + data_list[0][4] + " Joules")
    Output_file.write("\nProjectile weight: " + data_list[0][6] + " lbs")
    Output_file.write("\nProjectile material: " + data_list[0][8])
    Output_file.write("\nTotal shot weight: " + data_list[0][7] + " lbs")
    Output_file.write("\nPowder weight: " + data_list[0][5] + " lbs")
    Output_file.write("\nTarget: " + data_list[0][9])
    Output_file.write("\nMax stress: " + max_list[1] + " psi on Channel " +
max_list[0])

    Output_file.write("\n\nSensor order, from closest to furthest from impact
site:")
    for i in range(0, len(dist_ord)):
        x = str(dist_ord[i][0])
        Output_file.write("\n" + x)

    Output_file.write("\n\nStress order, highest to lowest:")
    for i in range(0, len(stress_ord)):
        x = str(stress_ord[i][0])
        y = str(stress_ord[i][1])
        Output_file.write("\n" + x + ": " + y + " psi")

    Output_file.write("\n\nRosette max axial strain data:")
    Output_file.write(("\nMax axial strain Ez: " + str(max_axial[0][2])))
    Output_file.write(("\nPrinciple strain E1: " + str(max_axial[0][0])))
    Output_file.write(("\nPrinciple strain E2: " + str(max_axial[0][1])))
    Output_file.write(("\nAverage strain: " + str(max_axial[0][3])))

    Output_file.write("\n\nRosette max axial stress data:")
    Output_file.write(("\nMax axial stress Sz: " + str(round(max_axial[1][2],
2))) + " psi")
    Output_file.write(("\nPrinciple stress S1: " + str(round(max_axial[1][0],
2))) + " psi")
    Output_file.write(("\nPrinciple stress S2: " + str(round(max_axial[1][1],
2))) + " psi")
    Output_file.write(("\nAverage stress: " + str(round(max_axial[1][3], 2)))
+ " psi")

    Output_file.close()
```

```
    for i in range(1, len(data_list)):
        x = i + 2
        x = str(x)

        Output_file = open(filepath + '/' + last_file + " Output.txt", "a")
        Output_file.write("\n\nChannel " + x + " Data")

        mx = data_list[i][0]
        mx = round(mx, 2)
        mx = str(mx)
        Output_file.write("\nMax Strain (ue): " + mx)

        mn = data_list[i][1]
        mn = round(mn, 2)
        mn = str(mn)
        Output_file.write("\nMin Strain (ue): " + mn)

        dist = data_list[i][2]
        dist = round(dist, 2)
        dist = str(dist)
        Output_file.write("\nDistance from hit location (in): " + dist)


        stress = data_list[i][3]
        stress = str(stress)
        Output_file.write("\nStress (psi): " + stress)

        Output_file.close()

    #Write to database
    last_file_pos = test_date_filepath.rfind("/", 0)
    last_file = test_date_filepath[last_file_pos + 1:]
    project_directory = test_date_filepath.rstrip(last_file)
    for file in os.listdir(project_directory):
        if file.endswith(".db"):
            data_file = os.path.join(project_directory, file)

    database = sqlite3.connect(data_file)
    r = database.cursor()

    axial = str(max_axial[1][2])

    r.execute("INSERT INTO shot VALUES (:date, :material, :proj_mass,
:pwdr_mass, :KE, :avg_vel, :max_S, :max_axial, :shot_os_directory)",
              {
                  'date': date,
                  'material': data_list[0][8],
                  'proj_mass': data_list[0][6],
                  'pwdr_mass': data_list[0][5],
                  'KE': data_list[0][1],
                  'avg_vel': data_list[0][0],
                  'max_S': max_list[1],
                  'max_axial': axial,
                  'shot_os_directory': filepath
```

```
          })

database.commit()
database.close()

return
```

Example Setup text document (.txt)

Plug Thickness: "36.0"

Test Date:18OCT2019

Sensor Locations (inches, x,y) bottom left corner (front view):

Sensor 1: "57.00, 36.00"

Sensor 2: "62.12, 37.12"

Sensor 3: "62.12, 33.88"

Sensor 4: "03.00, 36.00"

Sensor 5: "60.00, 69.00"

Sensor 6: "60.00, 03.00"

Strain Gauge rosette angles (degrees):

Theta a:0.0,

Theta b:135.0,

Theta c:225.0,

Strain Gauge on each Channel (DO edit this, no spaces/tabs):

CH3:8228

CH4:8266

CH5:8269

CH6:8226

CH7:8268

CH8:8227

Strain Gauge Serial Number and mV to ue conversion (Don't edit this unless new sensor used):

SN8266:42.2

SN8269:50.0

SN8226:52.0

SN8268:44.8

SN8227:51.4

SN8229:40.5

SN8267:50.8

SN8228:42.4

Concrete stress values (psi) (edit to change concrete properties):

Tensile avg:550.0,

Mod of Elas avg:4000000.0,

Poisson's Ratio avg:0.15,

Tensile max:700.0,

Tensile min:400.0,

Mod of Elas max:6000000.0,

Mod of Elas min:2000000.0,

Example Output File (.txt)

Steel Slug 10 lb Output File

Test Date: 18OCT2019

Projectile Data

Average Impact Velocity: 507.34 ft/s

Kinetic Energy of test projectile: 40000.21 ft-lbs, 54240.29 Joules

Sum KE of projectile, wad, and sabot: 74000.4 ft-lbs, 100344.54 Joules

Projectile weight: 10.0 lbs

Projectile material: steel

Total shot weight: 18.5 lbs

Powder weight: 1 lbs

Target: Reinforced Seal

Max stress: 167.25 psi on Channel 4

Sensor order, from closest to furthest from impact site:

CH3

CH4

CH5

CH7

CH6

CH8

Stress order, highest to lowest:

CH4: 167.25 psi

CH3: 117.77 psi

CH5: 100.58 psi

CH7: 72.88 psi

CH6: 60.22 psi

CH8: 52.33 psi

Rosette max axial strain data:

Max axial strain Ez: 24.80

Principle strain E1: -3.72

Principle strain E2: -10.36

Average strain: -7.04

Rosette max axial stress data:

Max axial stress Sz: 99.20 psi

Principle stress S1: -14.88 psi

Principle stress S2: -41.44 psi

Average stress: -28.16 psi

Channel 3 Data

Max Strain (ue): 29.44

Min Strain (ue): -22.53

Distance from hit location (in): 39.5

Stress (psi): 117.77

Channel 4 Data

Max Strain (ue): 41.81

Min Strain (ue): -30.89

Distance from hit location (in): 41.5

Stress (psi): 167.25

Channel 5 Data

Max Strain (ue): 25.14

Min Strain (ue): -22.34

Distance from hit location (in): 42.0

Stress (psi): 100.58

Channel 6 Data

Max Strain (ue): 15.05

Min Strain (ue): -11.16

Distance from hit location (in): 53.4

Stress (psi): 60.22

Channel 7 Data

Max Strain (ue): 18.22

Min Strain (ue): -21.03

Distance from hit location (in): 48.5

Stress (psi): 72.88

Channel 8 Data

Max Strain (ue): 13.08

Min Strain (ue): -13.43

Distance from hit location (in): 56.0

Stress (psi): 52.33

BIBLIOGRAPHY

1. Bise, C. J. (2003). *Mining Engineering Analysis*. Littleton, CO: Society for Mining, Metallurgy, and Exploration.
2. Perry, K. A. (2018) *Analysis of Coal Mine Seal Integrity from Explosively Driven Projectiles.* Rolla, MO: Missouri University of Science and Technology.
3. FEMA. (2017). *Windows, Doors, and Opening Protection*. (https://www.fema.gov/media-library-data/20130726-1707-25045-9020/chapter10.pdf)
4. Williams, F. (1975). The Role of Black Powder in Propelling Charges (United States, Picatinny Arsenal, Dover, NJ). NTIS.
5. Davis, T. L. ((1941) (2016)). The Chemistry of Powder and Explosives. Pickle Partners Publishing.
6. Conkling, J. A. (2010). *Chemistry of Pyrotechnics: Basic Principles and Theory* (2nd ed.). Boca Raton, FL: CRC.
7. Sassé, R. A. (1985). A Comprehensive Review of Black Powder. Aberdeen Proving Ground, MD: US Army Ballistic Research Laboratory.
8. Lancaster, R. (2006). *Fireworks: Principles and Practice* (4th ed.). New York: Chemical Pub. Co.
9. Brown, M. E., & Rugunanan, R. A. (1989). A Temperature-Profile Study of the Combustion of Black Powder and its Constituent Binary Mixtures. Grahamstown, South Africa: Chemistry Department, Rhodes University.
10. Rose, J. E., & Hardt, A. P. (1979). *Black Powder - A Modern Commentary*.
11. Shea, D. (2012, December 28). RAFFICA SPECIAL: The RPG-7 System. Retrieved from http://www.smallarmsreview.com/display.article.cfm?idarticles=1581
12. Mill Test Report (Rep. No. 19005RP16). (2016). Dusseldorf, Germany: Vallourec Deutschland GmbH.
13. 49 CFR 192.105. Design Formula for Steel Pipe. (Accessed 2020 from https://www.law.cornell.edu/cfr/text/49/192.105)
14. Pegler, M. (2009). *Sniper: A History of the Us Marksman*. Oxford: Osprey.

15. Cooper, P. W. (2010). Explosives Engineering. Wiley-VCH Inc.

16. Elger, D. F., Williams, B. C., Crowe, C. T., & Roberson, J. A. (2013). Engineering Fluid Mechanics (10th ed.). Erscheinungsort nicht ermittelbar: Wiley.

17. HSS Sections Beam and Column Load Tables. (1998). Steel Tube Institute of North America.

18. Steel Construction Manual (13th ed.). (2005). American Institute of Steel Construction.

19. https://www.finesoftware.eu/help/geo5/en/table-of-ultimate-friction-factors-for-dissimilar-materials-01/. (accessed 2020).

20. Steward, E.A., & Perry, K.A. (2019) Projectile Generator Design for Underground Coal Mine Seal Testing.

21. von Niederhausern, B.A. (2019) Projectile Impact Effects on a 50 psi Plug Type Coal Mine Seal.

22. Hibbeler, R. C. (2014). *Mechanics of Materials* (9th ed.). Harlow: Pearson.

23. Exhibit P-40, Budget Line Item Justification: FY 2018 Army, Exhibit P-40, Budget Line Item Justification: FY 2018 Army (2017).

24. Gates, Richard A., Robert L. Phillips, John E. Urosek, Clete R. Stephan, Richard T. Stoltz, Dennis J. Swentosky, Gary W. Harris, Joseph R. O'Donnell, Jr., and Russell A. Dresch, Report of Investigation - Fatal Underground Coal Mine Explosion - Sago Mine, Wolf Run Mining Co. Rep. No. 46-08791. Arlington: Mine Safety and Health Administration (2007).

25. https://www.law.cornell.edu/cfr/text/30/75.335

26. https://arlweb.msha.gov/Seals/Approved/120PSI/120M-18.0.pdf

27. https://arlweb.msha.gov/Seals/Approved/50PSI/50M-01.1.pdf

28. MSHA. Final Rule on Sealing of Abandon Areas Source Page. (2007). Arlington, VA: MSHA. doi:https://arlweb.msha.gov/Seals/SealsSingleSource2007.asp

29. NIOSH. Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines (Rep. No. IC 9500). (2007). Pittsburgh, PA: NIOSH.

30. Perry, K. A. (2010). *Development of Modern Sealing Techniques for Underground Coal Mining* (Doctoral dissertation, University of Kentucky, 2010). KY.

VITA

Ethan Allan Steward graduated from Northwestern Area High School in Mellette, SD, in 2011. He then attended the South Dakota School of Mines and Technology beginning the same year, graduating with a bachelor's in mining engineering and commissioning as an engineer officer in the US Army National Guard in 2017. In the fall of that year he attended the Missouri University of Science and Technology and received an M.S. in Explosives Engineering in December 2020.