Fall 2019

Projectile impact effects on a 50 psi plug type coal mine seal

Bruce Albert von Niederhäusern

Department:

PROJECTILE IMPACT EFFECTS ON A 50 PSI PLUG TYPE COAL MINE SEAL

by

BRUCE ALBERT VON NIEDERHÄUSERN

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN EXPLOSIVES ENGINEERING

2019

Approved by:

Dr. Kyle Perry, Advisor
Dr. Braden Lusk
Dr. Catherine Johnson

ABSTRACT

The current lack of projectile damage criteria for mine seals is assessed to determine if criteria should be developed. Impact experiments were performed on a full-scale 50 psi mine seal to determine failure modes, dynamic stress state, and assess risk to existing operational mine seals. Test projectiles were selected from common underground coal mine items propelled by means of a black powder projectile generator. Projectile velocities attained in testing ranged from approximately 100 to 650 ft/s (30 to 200 m/s). Dynamic strain was measured on the non-impact face to derive the state of stress and estimate failure criteria. Penetration models were used to estimate depth of penetration and compared to empirical data. Finite element analysis impact simulation was compared with empirical data and a critical impact velocity for rigid body projectiles of 20 to 30 lb (9 to 14 kg) was estimated to be in the range of 550 to 600 ft/s (168 to 183 m/s). Damage to the mine seal was observed and compared with numerical simulation to assess the need for defined projectile damage criteria to mine seals.

ACKNOWLEDGMENTS

First and foremost, I am grateful to my God, the source of all knowledge and opener of ways who has brought me through all challenges. Also, to the manifestation of God's love in my life: my wife, Becky, for supporting and enabling me to put in the many hours required to complete a project of this scope. I am also grateful to my family and friends for their enduring support and encouragement.

I am also grateful for my advisor, Dr. Kyle Perry, for providing me the opportunity to support this research and all the university staff and students that have contributed in a multitude of ways that ultimately resulted in this product. Additionally, I would like to thank Strata Worldwide, Mike Fabio, and the Alpha Foundation who provided material, expertise, and funding to make this project possible.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Page

LIST OF TABLES

Page

NOMENCLATURE

Symbol	Description
DDT	Deflagration to detonation transition
d_p	Projectile diameter
e.g.	Exampli gratia (for example)
f_c'	Unconfined compressive strength
ft	Foot or feet
ft/s	Feet per second
F_{ty}	Tensile yield strength
HDPE	High density polyethylene
i.e.	id est (that is)
kg	Kilogram or kilograms
kg/m^3	Kilograms per cubic meter
km/s	Kilometers per second
kPa	Kilo Pascal
lb	Pound or pounds
lb/in^3	Pounds per cubic inch
m	Meter or meters
m/s	Meters per second
MINER	Mine Improvement and New Emergency Response
mm	Millimeter
m_p	Projectile mass

MPa	Mega Pascal
N	Nose shape parameter
P	Pressure
psf	Pounds per square foot
psi	Pounds per square inch
s	Second or seconds
u	Particle velocity
U	Shock velocity
V_p	Projectile velocity
V_s	Striking (impact) velocity
We	Weber number
z	Penetration depth
Z	Shock impedance
$\dot{\epsilon}$	Strain rate
ε	Strain
ζ	Damping coefficient
μm	Micrometer
ν	Poisson's Ratio
ξ	Modified Weber number
ρ	Density
σ	Stress
φ	Angle of internal friction
ψ	Caliber-radius-head

1. INTRODUCTION

1.1. BACKGROUND

On 2 January 2006, an explosion occurred at the Sago mine. Root cause analysis revealed that lightning was the most likely ignition source of the methane air atmosphere behind a sealed, unmonitored section of the mine. Overpressures generated during the incident were estimated in excess of 93 psi (641.2 kPa). During this disaster, the lives of 12 miners were lost. As a result of mining related disasters, Congress passed the Mine Improvement and New Emergency Response Act (MINER) on 15 June 2006. The MINER act regulates design criteria for monitored and unmonitored seals. Monitored and unmonitored have different risk profiles for accidental explosions due to the differing atmosphere conditions. The amount of methane and oxygen in the atmosphere behind a monitored seal is controlled and kept inert to reduce the explosibility risk. For unmonitored seals, as the atmosphere ranges in methane and oxygen content, the explosibility risk is also variable (Zipf, Sapko, & Brune, Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines, Information Circular 9500, 2007). Monitored seals are required to withstand 50 psi (344.7 kPa) overpressure, and unmonitored seals are required to withstand 120 psi (827.4 kPa), both design pressure curves have an instantaneous rise time to the pressure limit and release after 4 seconds.

Methane-air explosions are the most common source of accidental explosions reported among major mining accidents in the United States. From the earliest recorded mining accidents in the United States between 1810 and 1958, most accidental explosions were determined to involve a methane-air atmosphere (Humphrey, 1960); from 1959-

1981 there were 20 major mining accidents, the preponderance of which were methane and coal dust explosions (Richmond, Price, Sapko, & Kawenski, 1983). More recently, there have been 12 explosions in sealed areas of active U.S. underground coal mines between 1986-2006 (Zipf, Sapko, & Brune, Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines, Information Circular 9500, 2007). Explosive mixtures in coal mines are commonly formed from air, natural gas (which is composed primarily of methane), and coal dust. Most accidental explosions in coal mines are deflagrations, but detonations can occur and can be extremely destructive, generating explosion pressures up to 1,450 psi (10 MPa) on reflections (Zipf, et al., Preliminary Large-scale DDT Experiments at NIOSH Lake Lynn Laboratory, 2011).

Even with the improvement of mine seal design criteria as a result of the "MINER Act" in 2007, projectile impact damage criteria for mine seals is not specified. A review of the recommended design criteria suggests that blast overpressures attained in the sealed atmosphere is correlated to the explosive run-up length[1] (Zipf, Sapko, & Brune, Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines, Information Circular 9500, 2007). The blast overpressure is critical for impact modeling because it is directly correlated to projectile velocity. Understanding the impact effects of mine seals would provide guidance or even specific criteria to mine operators and seal designers with which to assess seal integrity after an accidental explosion.

---

[1] For unmonitored seals with over 165 ft (50 m) explosive run-up length, the pressure peaks at 640 psi (4.4 MPa) and then drops to 120 psi (0.8 MPa); whereas for unmonitored seals with explosive run-up lengths under 165 ft (50 m), the pressure peaks at 120 psi (0.8 MPa).

A consideration of projectile loading of a structure is significant because as research has suggested there is a synergistic effect for blast and projectile loading. This results in an increase in structural response when both modes of loading are combined, in some cases internal stresses resulting from coupled blast and projectile loads were up to four times those for blast loads alone (Marchand, Vargas, & Nixon, 1992). While a mine seal may be designed to sustain a pressure wave in isolation, the coupled effects of blast and projectile loading may induce failure as a result of these synergistic effects.

In summary, the driving motive behind this research is to address the risk of mines seal failure as the result of projectile loading resulting from accidental methane explosions. Risk has been mitigated with the introduction of mine seals and increased overpressure design limits; however, the seals were designed specifically to resist the overpressure experienced during an accidental explosion. This research seeks to identify the risk to mine seals suffering damage from projectiles created during an explosion and warrant specific impact damage criteria.

1.2. LITERATURE REVIEW

A mine seal involved in an accidental explosion will sustain overpressure loading from blasting and, potentially, impact loading from projectiles. The loading from projectiles, and ultimately the dynamic response of the seal, will be dependent on the projectile velocity, which in turn, is dependent on the blast characteristics. The parameters of interest are blast overpressures, blast wave velocity, and duration of the blast.

The detonation limits of methane-air atmosphere have important implications for underground coal mines and hence the design of mine seals. Coal mines are prone to accidental explosions because natural gas (primarily methane) is released by coal layers (Gamezo, Zipf, Kessler, & Oran, 2013). To be combustible and explosive, an atmosphere containing methane (CH_4) requires oxygen[2]. A methane and oxygen atmosphere is flammable in the concentration limits of 5%-15% methane (Zabetakis, 1965), these limits are comparable to those seen in the Coward's triangle which shows explosibility limits for methane, as in Figure 1.1.

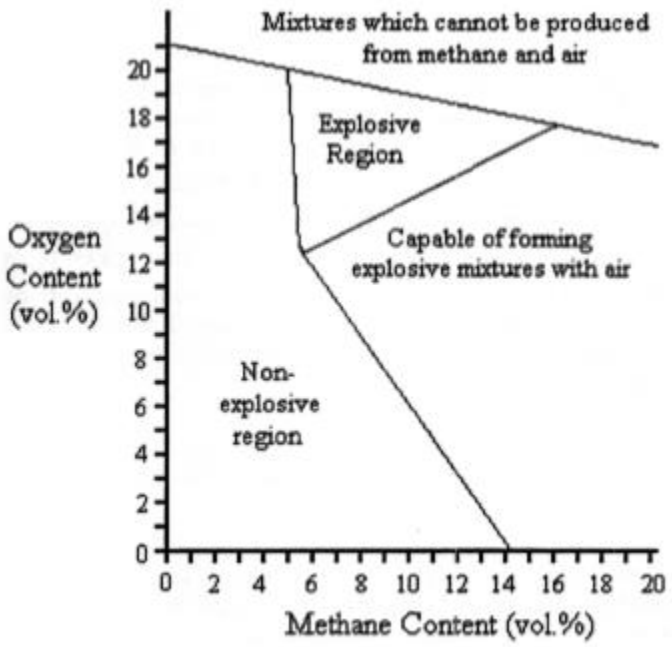


Figure 1.1. Coward's triangle for methane (Gillies & Jackson, 1998)

[2] The oxidation reaction of methane is: $CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$. The heat of reaction per unit weight of methane is 1,206.1 kcal/lb (11.125 kJ/g) (Cooper, 1996).

Research indicates that “most sealed areas will eventually enter this explosive range at some point after sealing” (Zipf, Sapko, & Brune, Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines, Information Circular 9500, 2007). Tests were conducted at the NIOSH Lake Lynn Laboratory of methane-air detonations which successfully demonstrated detonations for mixtures between 5.3% and 15.5% methane. As part of this test, detonation velocities were measured between 4,961 – 6,112 ft/s (1,512 – 1,863 m/s), and overpressures of 174 – 247 psi (1.2 – 1.7 MPa) (Zipf, et al., Methane-Air Detonation Experiments at NIOSH Lake Lynn Laboratory, 2010). Research into the burning velocity of stoichiometric or near-stoichiometric mixtures of methane and air suggests optimized burning velocities near stoichiometric mixtures (Zabetakis, 1965), but it is possible to develop high flame speeds and high shock pressures for mixture compositions that are far from stoichiometric (Zipf, et al., Preliminary Large-scale DDT Experiments at NIOSH Lake Lynn Laboratory, 2011). Although the overpressures for methane-air explosions and confined explosions are not comparatively high when measured against detonation pressures, the durations are relatively long and can be on the order of seconds as opposed to milliseconds for external blast waves (Kinney & Graham, 1985). This would result in an increased duration on which to transfer energy to potential projectiles.

A review of the literature yielded models for an internal, non-vented methane gas explosion which could be used to determine pressure-time relations. During an internal blast, the general form of the pressure-time trace suggests three distinct regions: a) pressure rise, b) high pressure region, and c) decay, as depicted in Figure 1.2.

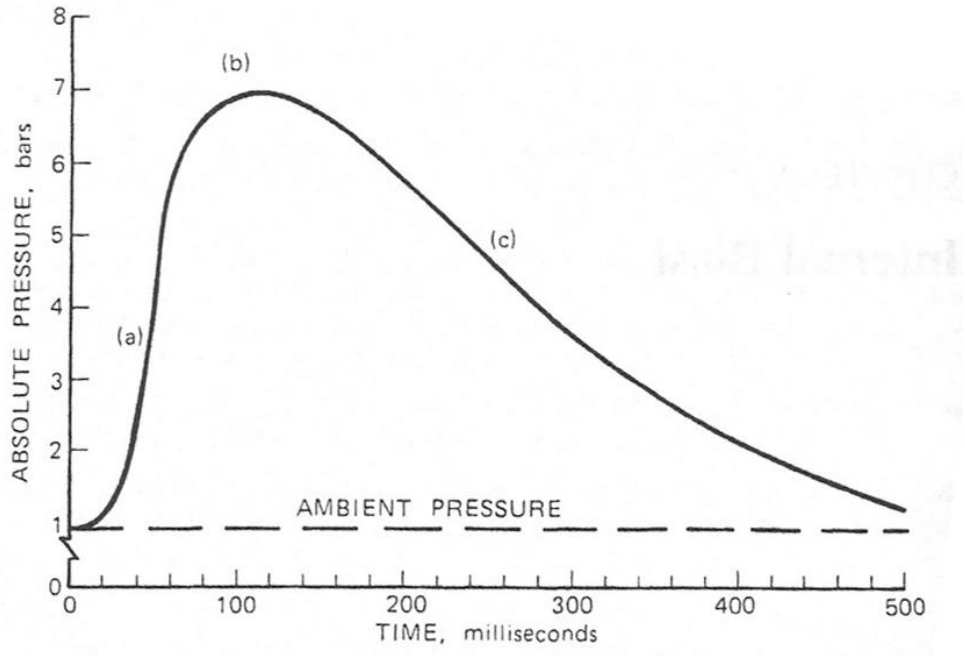


Figure 1.2. Pressure-time trace for a typical internal explosion, from (Kinney & Graham, 1985)

Atmosphere conditions (i.e. methane and oxygen content) and quantity of the combustible material (i.e. gas volume) will drive the blast overpressure levels and duration. The typical coal mine tunnel dimensions are 7 ft (2 m) high, 20 ft (6 m) wide, and 3,281 ft (1,000 m) long tunnel (Gamezo, Zipf, Kessler, & Oran, 2013). The length of the tunnel is sufficient to expect a DDT, based on explosive run-up lengths, provided the proper conditions. Research indicates that since typical coal mine tunnels greatly exceed the estimated run-up distance to DDT[3], it is "very likely that detonation can develop during accidental natural gas-air explosions in coal mines" (Gamezo, Zipf, Kessler, & Oran, 2013). At a point far from the initiation point, the initiation mechanism is

[3] Approximately 59 – 89 ft (18 – 27 m) for typical tunnels

negligible and a mathematical simulation can represent the blast wave, since "all blast waves at reasonable distances from the center of an explosion show similar wave forms" (Kinney & Graham, 1985). The pressure rise region is modeled using the rate of pressure increase equation for an internal blast (Kinney & Graham, 1985):

$$\dot{P}_c = \frac{dP}{dt} = C_r \left(\frac{S}{V}\right)(P - P_{ind})(P_{ad} - P) \tag{1}$$

where $\dot{P}_c$ is the rate of pressure increase, C_r is the pressure rise rate coefficient, S is the internal surface area of the confining volume V, P is the absolute pressure at time t, P_{ad} is the theoretical limiting adiabatic pressure, and P_{ind} is a term introduced to provide for possible initial pressure rise delay (Kinney & Graham, 1985).

The duration of the blast wave can be simulated using the following relationship:

$$\frac{t_d}{W^{\frac{1}{3}}} = \frac{980\left(1 + \left(\frac{Z}{0.54}\right)^{10}\right)}{\left(1 + \left(\frac{Z}{0.02}\right)^{3}\right)\left(1 + \left(\frac{Z}{0.74}\right)^{6}\right)\sqrt{1 + \left(\frac{Z}{6.9}\right)^{2}}} \tag{2}$$

where $t_d/W^{1/3}$ is the duration in milliseconds for a one kilogram TNT explosion at scaled distance Z in meters (Kinney & Graham, 1985). The above model is for an external blast, the duration of which is comparatively short relative to an internal blast.

Evaluating the pressure design criteria used for the mine seals is neither the intent nor within the scope of this research as this thesis focuses specifically on the projectile loading of mine seals. However, a comparison of the design criteria used for the construction of the mine seals supports the relevance and applicability of the internal blast wave simulations. The NIOSH design pressure-time curves for 120 psi (0.8 MPa) and 50 psi (0.3 MPa) provide the limits used for validation of the model. The NIOSH 120 psi (0.8 MPa) pressure-time curve rises to 120 psi (0.8 MPa) in 0.25 seconds and

maintains the pressure for 1 second, the 50 psi (0.3 MPa) pressure-time curve rises to 50 psi (0.3 MPa) in 0.10 seconds and maintains the pressure for 1 second (Zipf, Sapko, & Brune, Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines, Information Circular 9500, 2007). The Code of Federal Regulations specifies the blast duration at a more conservative four second duration with the load instantaneously applied and released (30 C.F.R. §75.335, 2019).

During a blast, a secondary effect is the generation of projectiles from debris or other material in the path of the blast wave. These projectiles can be propagated at high velocities and cause damage to structures or personnel distal to the locus of blast initiation.

The Unified Facilities Criteria provides a model for estimating the velocity of unconstrained objects. The interaction of the blast wave with the object is represented in three phases: incident blast wave with object, diffraction of blast wave around the object, and the reformed blast wave/drag phase (Department of Defense, 2008). Fragments or projectiles formed as the result of an accidental explosion will be of various shapes, but in general tend to exhibit drag coefficients of much greater magnitude than lift coefficients (Department of Defense, 2008). Estimates for the drag force, and drag-only acceleration of the fragments is given by UFC 3-340-02:

$$F_D = \frac{1}{2} C_D A_D \rho v^2 \tag{3}$$

where F_D is the drag force, C_D is the drag coefficient, A_D is the drag area, ρ is the density of the medium, and v is the relative velocity.

In combination with the drag force, the velocity of a fragment can be determined through numerical simulation, utilizing the classical equation of motion and small-time steps:

$$F = m * \frac{d^2x}{dt^2} \quad (4)$$

where *F* is the applied force, m is the mass of the fragment, *x* is distance, and *t* is time.

Assessing impact phenomena is a requires an integration of multiple fields of study, from solid mechanics, dynamics, materials science, fluid dynamics, wave physics and others. The material properties of mine seals adds complexity to impact events because the mechanical properties of concrete are strain rate dependent, and higher strain rates yield higher strength behavior (Cadoni, Albertini, Labibes, & Solomos, 2001). Low velocity impacts, e.g. below 6.6 ft/s (2.0 m/s), can generally be modeled using structural dynamics. However, as the impact velocity increases, e.g. 1,640 – 4,921 ft/s (0.5-1.5 km/s), the response of the target tends to become localized (2-3 projectile diameters) near the impact zone (Zukas, Impact Dynamics: Theory and Experiment, 1980). An analytical approach of the impact response is not only a difficult problem but it would require material characterization which is not available (Zukas, Impact Dynamics: Theory and Experiment, 1980). Even with analytical models there remains the need to verify the model predictions. The cost to verify impact response for a full-scale test with velocities in the anticipated range was not prohibitive as the required materials and propulsion method were readily available. As a result, an empirical approach was determined to be a viable approach, which would incorporate the support of numerical models. One of the methods of characterizing the impact phenomena is through the use of impact velocity. Zukas provides a range of strain rates correlated to the impact velocity, material effect,

and method of loading. For impact velocities in the range of 164 – 3,281 ft/s (50 - 1,000 m/s) the research yields a notional range for estimated strain rates of 10^2 to 10^5 s^{-1} with powder guns being a suitable method of loading (Zukas, Impact Dynamics: Theory and Experiment, 1980). Note also, the correlation from strain rate to dynamic considerations listed in Table 1.1.

Table 1.1. Dynamic behavior, adapted from (Zukas, Introduction to Hydrocode, 2004)

Duration (s)	10^8	10^6	10^4	10^2	10^0	10^{-2}	10^{-4}	10^{-6}	10^{-8}
Strain rate $\dot{\epsilon}$ (s^{-1})	0	10^{-8}	10^{-6}	10^{-4}	10^{-2}	10^0	10^2	10^4	10^6
	Creep		*Quasi-static*			*Intermediate strain-rate*	*Impact*	*High velocity impact*	
Method of loading	Constant load stress machine		Hydraulic screw machine			Pneumatic mechanical machines	Mechanical explosives impact	Gas gun Explosively driven plate impact	

Table 1.1. Dynamic behavior, adapted from (Zukas, Introduction to Hydrocode, 2004) (cont.)

Dynamic consideration	Strain vs. Time Creep rate record	Constant strain rate test	Mechanical resonance on specimen and machine	Elastic-plastic wave propagation	Shockwave propagation
	Inertia force neglected		*Inertia force important*		
	Isothermal		*Adiabatic*		

For the impact velocities expected, a powder gun was determined a feasible option as it is expected to have the capability to generate the required projectile velocities. In the estimated strain-rate range, mechanical resonance of the target is of interest, the inertia forces are important, and the system can be modeled as adiabatic, due to the short impact duration (~10^{-2} s) (Zukas, Introduction to Hydrocode, 2004).

To discuss impact on the macroscopic level, it is helpful to understand what is happening on a microscopic level. In static and quasi-static deformation, the applied and internal forces are in equilibrium and the net forces on each element in the body is zero or

near-zero (Meyers, 1994). In dynamic deformation, when the external load is initially applied, the stress has not yet been transferred to all the elements in the body as it is transferred in the body via mechanical waves at specified velocities (Meyers, 1994). The velocity at which this stress is transferred depends upon the type of wave. The primary types of waves encountered are elastic and plastic. Elastic waves, which are composed of longitudinal (p-wave), distortional (shear), surface (Rayleigh and Love), interfacial, and flexural (Meyers, 1994). The magnitude of the stress wave is dependent on the external force. If the magnitude of this stress wave exceeds the dynamic elastic strength of the material, a plastic wave will form, and in a bounded medium, it will become a shock wave (Meyers, 1994).

The simplest impact scenario that can be studied is a planar, normal, parallel impact (Meyers, 1994). This can be used to model a perpendicular impact between two flat and parallel surfaces. In this case, the contact of the colliding surfaces occurs simultaneously.

If during the impact event the stress wave amplitude is much greater than the dynamic flow strength of a material, then the shear stresses can be neglected (Meyers, 1994). A shockwave is defined as a "discontinuity in pressure, temperature (or internal energy), and density" (Meyers, 1994). In order to establish a shock wave, the velocity of the pulse increases with increasing pressure (Meyers, 1994). For an established shock wave, the calculation of the wave parameters is based on the Rankine-Hugoniot conservation equations, derived from the conservation of mass, momentum, and energy. The conservation equations for mass, momentum, and energy are:

$$\rho_0 U = \rho(U - u) \tag{5}$$

$$(P - P_0) = \rho_0 U u \tag{6}$$

$$E - E_0 = \frac{1}{2}(P + P_0)(V_0 - V) \; or \; E - E_0 = \frac{Pu}{\rho_0 U} - \frac{1}{2}\rho_0 \frac{Uu^2}{\rho_0 U} \tag{7}$$

where ρ is density, U is the shock velocity, u is particle velocity, P is pressure, E is energy, and V is the specific volume.

The shock equation of state, U-u Hugoniot, of a material is given by:

$$U = C_0 + S_1 u + S_2 u^2 + \cdots \tag{8}$$

where S_1, S_2 are empirical parameters, and C_D is the sound velocity of the material. For most metals, the shock equation of state is a linear relationship. Equation of state parameters for concrete are taken from a Sandia National Laboratories research report in 1996. A number of impact tests were identified in the report but the shock parameters close to the estimated impact velocities were taken. From the report, the test parameters used were for concrete with density at 0.082 – 0.083 lb/in^3 (2,260-2,290 kg/m^3), impact velocity of 1,716 – 1,719 ft/s (0.523-0.524 km/s). The equation of state parameters are C_0 of 10,827 ft/s (3.3 km/s), and S of 2.0 (Grady, 1996). Therefore, the Equation of State for concrete can be estimated by:

$$U_s = 3.3 + 2.0U_p \tag{9}$$

In an impact event, one of the key parameters that assists in describing the response is the shock impedance. Shock impedance (Z) is given by the quantity:

$$Z = \rho_0 U_s \tag{10}$$

where ρ_0 is the density, and U_s is the shock velocity. Figure 1.3 illustrates the impact response cases based on the projectile and target impedance relationships.

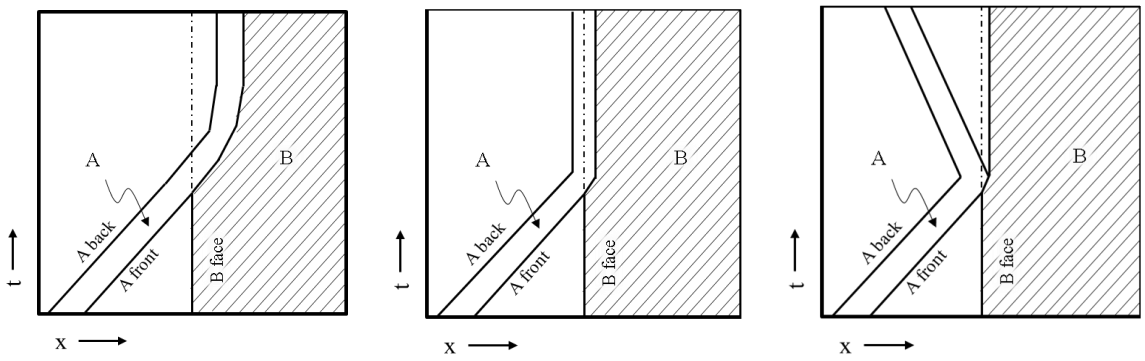


Figure 1.3. Projectile and target impact response x-t diagrams correlated to impedance relationships, adapted from Cooper (Cooper, 1996): a. $Z_{projectile} > Z_{target}$ b. $Z_{projectile} = Z_{target}$ c. $Z_{projectile} < Z_{target}$

The target impacted by the projectile will exhibit wave behavior, and the waves of primary interest are elastic, plastic, and shock waves. Of the elastic waves, Rayleigh, shear, and compression waves dominate, with Rayleigh waves accounting for approximately 67% of the total energy, shear waves for 26%, and compression waves for about 7% (Meyers, 1994). A visual depiction of these waves can be seen in Figure 1.4. When a stress pulse applied to a target has a greater amplitude than the material elastic limit, a plastic wave is generated. The wave front of a plastic wave can develop into a shock wave when the stress pulse "greatly exceeds the dynamic flow strength" of the target material (Meyers, 1994). The shock wave velocity is directly proportional to the pressure wave amplitude that exceeds the material flow strength, whereas the elastic wave velocities are correlated to initial material parameters.

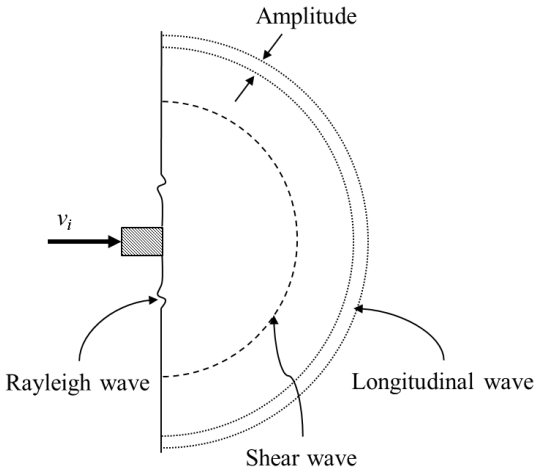


Figure 1.4. Elastic waves resulting from impact on a semi-infinite target, adapted from (Meyers, 1994)

For an unbounded, isotropic, elastic medium, the wave velocities for longitudinal, shear, and Rayleigh waves can be determined by the following equations (Meyers, 1994):

$$C_L = \left(\frac{\lambda + 2\mu}{\rho}\right)^{\frac{1}{2}} \tag{11}$$

$$C_S = \left(\frac{\mu}{\rho}\right)^{\frac{1}{2}} \tag{12}$$

$$C_R = \left(\frac{0.862 + 1.14\nu}{1 + \nu}\right) C_s \tag{13}$$

where C_L is the longitudinal wave velocity, C_S is the shear wave velocity, and C_R is the Rayleigh wave velocity, ρ is the medium density, ν is Poisson's ratio, and λ and μ are Lame's parameters.

As the stress waves are propagating through a material in a non-infinite medium, they will encounter boundaries. The type of boundary is relevant for this research as the seal boundaries will interact with stress waves during impact. For fixed boundaries, an

incident stress wave will maintain its sense (i.e. a compressive wave will reflect as a compressive wave). Whereas, for free boundaries, an incident stress wave will change sense (i.e. a compressive wave will reflect as a tensile wave). Or, more particularly, for an interface between two materials, A and B, a reflected pulse is of the same sign as the incident pulse when the sonic impedance of B is greater than the impedance of A, and it will be of the opposite sign when the impedance of B is less than the impedance of A (Meyers, 1994). This is noteworthy as seals in an operational configuration are integrated into the surrounding medium on all edges, resulting in a fixed-fixed-fixed-fixed plate configuration. The free surface that is of particular interest is the rear face, as it is this face that will likely experience spalling as a result of reflected stress waves (i.e. dynamic tensile failure). Impacts located far from free edges will minimize spurious free surface effects as the stress wave decays during propagation through the medium.

One of the challenges with applying many of the analytical models is that they assume a homogenous, isotropic, and elastic medium. A material can be said to be isotropic, "if the constituents of the material of a solid member are distributed sufficiently randomly, any part of the member will display essentially the same material properties in all directions" (Boresi & Schmidt, 2003). Concrete is by design, a heterogeneous mixture and in standard temperature and pressure conditions exhibits brittle behavior. This research will in some cases compare analytical and empirical results. Effort has been made to identify limiting assumptions that may point to the disparity or applicability of results.

From a general theoretical approach, the dynamic response of a single degree of freedom cantilever system subjected to an impulse load, can be estimated using (Rao, 2011):

$$x(t) = F\Delta t * \frac{e^{-\zeta\omega_n t}}{m\omega_d}\sin(\omega_d t) \tag{14}$$

where *x(t)* is the displacement relative to time, *F* is the impact force and Δt is duration of the applied force, ζ is the damping coefficient, *m* is the total mass of the system (projectile and target), and ω_n and ω_d are the undamped and damped natural frequencies, respectively. A damping coefficient of 2% of critical was initially selected as a rough estimate for structures from (Dowding, 1985) and this damping coefficient was adjusted to find agreement with the data. The force of impact was taken to be the derived interface pressure and the duration was estimated from the expected pulse duration based on the time required for the interface pressure to decay to a near-zero equilibrium after impact.

Classical impact theory is based on the impulse-momentum law for rigid bodies and incapable of describing the stress transients or deformations. It is instead limited to estimating initial and final velocity states. Goldsmith also states that in classical impact theory, “a negligible fraction of the kinetic energy of the system is converted to vibrations in the colliding bodies”. This assumption was found to be valid for two spheres but not for collisions involving rods, beams, or thin plates. The error of classical impact theory will be low when the duration of the impact event is large compared to the period of the lowest natural frequency of either body (Goldsmith, 1960). Goldsmith classifies impact phenomena based on the dominant type of wave produced by the event:

longitudinal, transverse, or torsional. As noted previously, for the anticipated range for strain-rate, the duration of the expected impact events will be low and body vibrations are not negligible. In addition, the geometries of the projectiles and target involved in our test cases have multiple planar surfaces. Thus, the random impact events realized in the consequence of an accidental explosion in an underground mine is not expected to be characteristic of a collision of two spheres. As a result, classical impact theory is not sufficient to explain projectile impact events with mine seals.

In analyzing impact events, particular interest is given to penetration which is one method to provide an estimate for damage realized to the target. Numerous models have been developed which vary regarding applicability depending on the type of projectile, projectile material, and even target material. Penetration models vary from models tailored to conventional small arms tangent-ogive projectiles to high velocity, over 3,280 ft/s (1,000 m/s), explosively formed projectiles. A search for penetration models tends to produce results related to these two general categories. Two models were identified for use with varying projectiles and a concrete target. Some adjustment to the parameters will be made to accommodate non-tangent-ogive projectiles used in testing. The two models identified were the Forrestal and National Defense Research Committee (NDRC) penetration models. The Forrestal model, assumes a rigid body projectile (non-deforming projectile) and minimal projectile yaw (under 5 degrees). The model can be used to determine depth of penetration (z) into a concrete target:

$$z = \begin{cases} V_p\sqrt{\dfrac{m_p}{c}} & for\ V_p < V_{s2} \\ \dfrac{2m_p}{\pi d_p^2 N \rho_t}\ln\left(1 + \left(\dfrac{N\rho_t V_1^2}{S_f * f_c{}'}\right)\right) - 2d_p & for\ V_p > V_{s2} \end{cases} \tag{15}$$

$$N = \frac{8\psi - 1}{24\psi^2} \tag{16}$$

$$\psi = \frac{s}{d} \tag{17}$$

$$S_f = 93.48\left(\frac{f_c'}{1e6}\right)^{-0.5603} \tag{18}$$

$$c = \frac{m_p}{4d_p^2}\left(V_p^2 - V_1^2\right) \tag{19}$$

$$V_1 = \sqrt{\frac{2m_p V_p^2 - \pi d_p^3 S_f f_c'}{2m_p + \pi d_p^3 N \rho_t}} \tag{20}$$

$$V_{s2} = \sqrt{\frac{\pi d_p^3 S_f f_c'}{2m_p}} \tag{21}$$

where V_p is the projectile velocity (m/s), V_1 is the velocity of a projectile located at the end of the crater region (m), V_{s2} is the projectile velocity under which the projectile will not form a tunnel (m/s), m_p is the mass of the projectile (kg), d_p is the diameter of the projectile (m), ρ_t is the density of the target material $\left(\frac{\text{kg}}{\text{m}^3}\right)$, N is a dimensionless nose shape parameter, ψ is the caliber-radius-head of the projectile, s is the ogive radius of the projectile (m), S_f is a dimensionless strength parameter, and f_c' is the unconfined compressive strength of the target concrete (Pa).

The NDRC model, derived from (NDRC, 1946) is shown below:

$$G\left(\frac{x}{d}\right) = KNd_p^{0.2}\left(\frac{m_p}{d_p^3}\right)\left(\frac{V_p}{1000}\right)^{1.80} \tag{22}$$

$$K = \frac{180}{\sqrt{f_c'}} \tag{23}$$

$$N = 0.72 + 0.25 * \sqrt{\psi - 0.25} \tag{24}$$

$$z = \begin{cases} 2d_p\sqrt{G\left(\frac{x}{d}\right)} & for\ 2\sqrt{G} < 2 \\ d_p\left(1 + G\left(\frac{x}{d}\right)\right) & for\ 2\sqrt{G} > 2 \end{cases} \quad (25)$$

where K is the concrete permeability factor taken from (Kennedy, 1976), N is a dimensionless nose shape parameter, d_p is the diameter of the projectile (in), m_p is the mass of the projectile (lb), V_p is the projectile velocity (ft/s), and f_c' is the unconfined compressive strength of the target concrete (psi).

A review of the pertinent literature ranges from an understanding of methane explosions and DDT considerations, to projectile generation, impact and penetration, to the localized and macro-level structural response of the target. With these considerations, there is space to explore the dynamic response of mine seals when subjected to projectile impact loading and a sufficient base of existing knowledge to analyze test data. This is vital to addressing the current lack of projectile damage criteria for mine seals.

1.3. PRELIMINARY MODELLING

Projectile modelling began with a definition of a projectile, for simplicity, this was modeled as a cylinder 3 inches (76.2 mm) in diameter and 4.5 inches (114.3 mm) in length, oriented with the long axis parallel to the flow. The drag coefficient used was 0.82, taken for a cylinder oriented parallel to flow from UFC 3-340-02. The acceleration simulation was run using a one dimensional flow field velocity of 6,112 ft/s (1,863 m/s) for a duration of 0.5 s and the remaining 3.5 seconds the flow field drops to 14.8 ft/s (4.5 m/s). The post-detonation front burn velocity was estimated from the peak burn rate for methane from (Zabetakis, 1965). The gases behind the detonation front will be moving at

a greater velocity than the burn rate, but it is used as a conservative estimate. In this simulation, it is assumed that the unconstrained objects are far from the initiation point, assuming a parallel flow field and plane wave interaction on the surface of the objects. The resulting projectile velocities for the design projectile and material types are given in Table 1.2:

Table 1.2. Design projectile impact velocities

	HDPE	Steel	Wood	Concrete	Limestone
Fragment mass (lb) [kg]	1.1 [0.5]	9.0 [4.1]	0.4 [0.2]	2.6 [1.2]	3.5 [1.6]
Velocity (ft/s) [m/s]	731.6* [223.0*]	955.7 [291.3]	1434.7 [437.3*]	448.2 [136.6]	500.3 [152.5]

* - The time parameters for projectiles below 2.2 lb (1 kg) were adjusted to prevent unstable velocity growth.

The design pressure curves used could propel projectiles to velocities in excess of 1,000 ft/s (305 m/s) based on projectile models implemented by the author.

As the types of projectiles that could potentially impact a mine seal during an accidental explosion vary in material composition, a cursory review of the types of expected responses is instructive. An example of the types of materials that could be expected to impact a mine seal would be steel, wood, concrete, HDPE, and various rock types (i.e. limestone). Using the Hugoniot relations, estimates for the interface pressure

and shock velocities can be estimated for impacts with these materials. The equation of state parameters for the materials were derived from the Los Alamos Series on Dynamic Material Properties (Marsh, 1999). The parameters are given in Table 1.3:

Table 1.3. Dynamic material properties for test case materials

Parameter	HDPE	Steel (304)	Wood (Sugar Pine)	Concrete	Limestone (approximate Calcium Oxide)
ρ (g/cm^3)	0.95	7.89	0.45	2.30	2.98
C_0 (km/s)	2.81	4.58	0.45	3.38	3.59
S	1.58	1.49	1.33	1.33	1.82
Interface values for estimated impact velocities, u_0					
u_0 (km/s)	0.223	0.291	0.437	0.137	0.153
u_1 (km/s)	0.061	0.240	0.024	0.068	0.088
$P_{interface}$ (GPa)	0.50	2.1	0.19	0.54	0.71
U_{target} (km/s)	3.42	3.78	3.35	3.44	3.48
$Z_{projectile}/Z_{target}$	0.4	4.9	0	1.0	1.4

From Table 1.3 and referencing the shock impedance cases from Figure 1.3, it can be seen that the HDPE, wood, and concrete projectiles are not expected to penetrate the seal; however, the steel and rock projectiles are expected to penetrate.

From the estimated impact velocities, the P-u Hugoniots for the projectile and target were constructed and the interface pressure was determined from the intersection of the Hugoniots as seen in Figure 1.5.

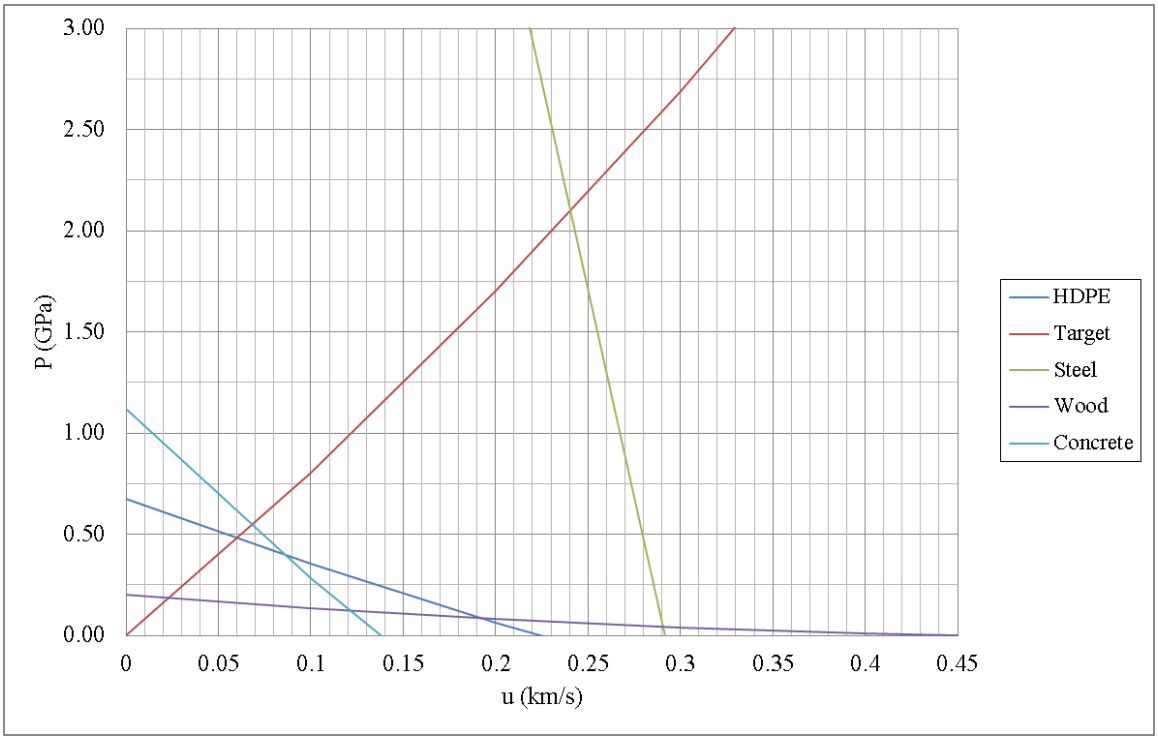


Figure 1.5. Hugoniots for target and test projectile materials

The P-u Hugoniots for the projectiles are denoted by the material type in the legend, and the target Hugoniot is constant for all impacts as the initial target state is at rest. The relative magnitudes of the interface pressures for each impact can be estimated qualitatively by inspection. By observation, it is also apparent that for an equal impact velocity, each material will produce a different interface pressure as the result of the differing slopes of the P-u Hugoniots. These Hugoniots do not only provide the interface pressure on impact, but also estimate the path of the relief waves resulting from impact.

The pressure pulse duration can be estimated for an impact by stepping the impact pressure down to zero through a series of relief waves. If the shock impedance of the projectile is greater than that of the seal, the interface pressure is incrementally reduced by rarefaction waves from the projectile until it reaches zero. This will produce a stepped pressure pulse in the target, for example see Figure 1.6.

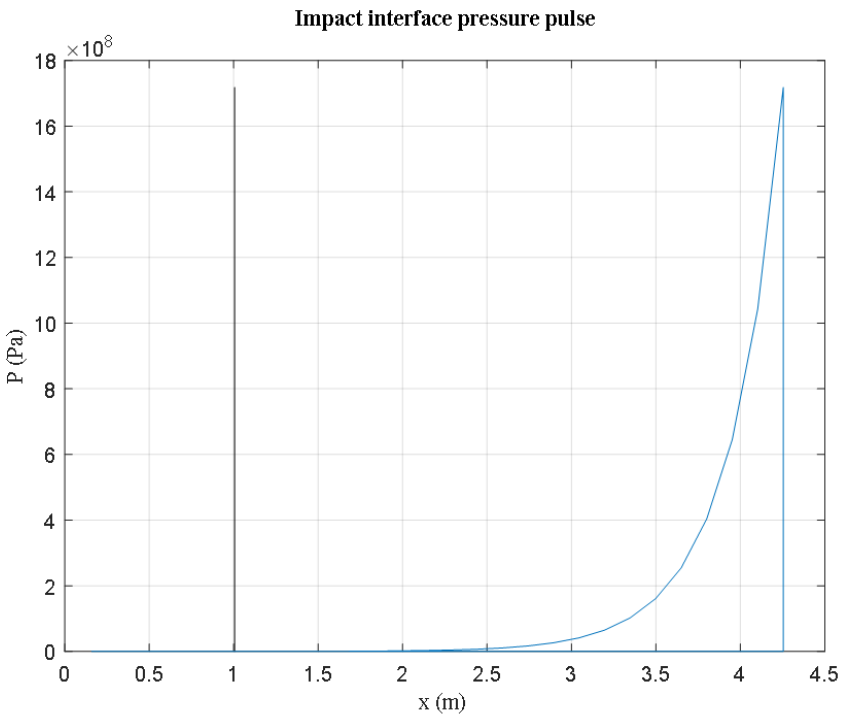


Figure 1.6. Pressure pulse shape for a steel projectile, impact velocity: 732 ft/s (223 m/s)

For a projectile with equal or lesser shock impedance than its target, the pressure pulse will decrease to zero after the initial projectile rarefaction reaches the impact interface, and this will produce a square pressure pulse in the target. The duration of a pressure pulse can then be estimated by taking the cumulative time for the projectile pressure and rarefaction waves to traverse the projectile until the interface pressure is

decremented to zero. The pulse durations are expected to be on the order of milliseconds (e.g. the steel projectile pulse is estimated to be approximately 1.2 ms). However, this is dependent on projectile geometries and relative shock impedances.

The dynamic response was estimated using the single degree of freedom equation, Eq. 13. From the initial damping coefficient (ζ) estimate of 2%, a mid-range value of 10% was also used to estimate a response with greater damping. The modeled dynamic behavior can be seen in Figure 1.7.

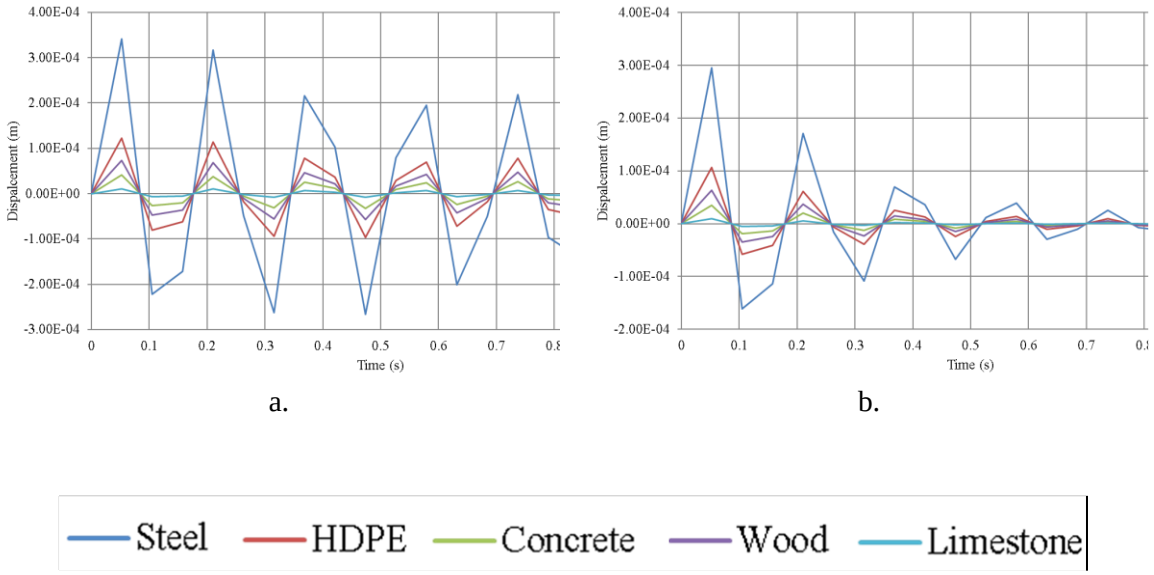


Figure 1.7. Estimated dynamic response for impact with, a. $\zeta = 0.02$, b. $\zeta = 0.10$

From the dynamic, free vibration model, the peak displacements are estimated to range from $3.7\text{x}10^{-4}$ inch – 1.2 inch (9.4 μm – 29.5 mm). As projectile velocities increase, the interface pressure is expected to increase, and thus the displacement magnitude, since impact force and displacement are directly proportional in Eq. 14. The

vibration model, Eq. 14, which does not incorporate shock effects or penetration, predicts projectile mass is expected to have negligible impact on the displacement of the system, as the mass of the seal is orders of magnitude greater than the projectile. After displacement is estimated, the strain would be determined by strain-displacement relations.

To assist in qualifying the impact events observed in testing, a dimensionless parameter was sought that would incorporate projectile and target properties (body and material forces) that would more completely compare impact parameters across projectile types. As a result, a modified Weber number was applied to the test cases. This number is used to represent the relative inertia effects to a material's cohesive ability. In fluid mechanics, it is a ratio of inertial force to surface tension force (Cengel & Cimbala, 2006):

$$We = \frac{\rho V^2 L}{\sigma_s} \tag{26}$$

where *We* is the Weber number, ρ is the density of the fluid, *V* is the velocity of the fluid, *L* is a characteristic length, and σ_s is the surface tension. This fluid mechanics parameter was applied for impact cases by adapting the kinetic energy (of the projectile) and the surface energy (of the target). The Weber number was adapted by correlating the characteristic length to the diameter of the projectile and using the kinetic energy equation as a basis. Fitting these concepts together yielded the modified Weber number:

$$We^* = \xi = \frac{E_{kinetic}}{E_{surface}} = \frac{\frac{1}{2}\rho_p v^2 d_p^2}{R_t} \tag{27}$$

$$R = \frac{1}{2}\frac{F_{HEL}^2}{E_{dyn}} \tag{28}$$

where ξ is the modified Weber number, ρ_p is the density of the projectile, v is the projectile velocity, d_p is the diameter of the projectile, and R_t is the modulus of resilience of the target adapted for dynamic loading, with F_{HEL} being the Hugoniot elastic limit of the target and E_{dyn} as the dynamic modulus of elasticity. Using the Hugoniot elastic limit specifically tailors the parameter to the penetrating resistance of the target. The modulus of resilience "is a measure of the energy per unit volume (energy density) absorbed by a material up to the time it yields under load and is represented by the area under the stress-strain curve to the yield point" (Boresi & Schmidt, 2003). Projectile diameter was selected as the characteristic length as opposed to projectile length because diameter, for a normal impact, is the dimension correlated with impact force[4]. The dimensionless parameter is intended to act as a comparison tool between impact events and the induced stress and penetration.

In addition, an FEA model was developed to simulate impact events, predict the seal stress response, and model the modal response. Due to the number of projectiles and material types anticipated to be used in testing, the complexity (primarily number of elements) of the models was reduced to limit the solution times and facilitate multiple simulation iterations. Simulations were run using primarily hexahedral elements with a constant impact velocity of 732 ft/s (223 m/s) for all tests to compare the seal response with constant input parameters. Which was taken from the estimated impact velocity for an HDPE projectile. Impacts were planar and parallel, with locations varying from

---

[4] Interface pressure can be estimated from Hugoniot relations and translated into force by distribution over the interface area (i.e. projectile diameter).

central to eccentric. The estimated maximum strains for the simulations ranged from 3.47×10^{-4} to 2.56×10^{-3}.

The modal analysis for the 50 psi mine seal, returned the first natural mode at 147.3 Hz[5]. By correlating the impulse duration to the frequency, the natural mode of the seal is a lower frequency than the expected response frequencies[6]. Where the natural frequency may be a consideration would be for long, high impedance projectiles (e.g. steel).

---

[5] The first natural modes are (Hz): 147.3, 217.4, 254.4, 410.2, 494.9, 545.1, 547.4, 575.6, 614.4, 714.5

[6] Preliminary modeling estimates pressure pulse durations at approximately 1 ms, the frequency of this pulse duration would be 1,000 Hz. The expected pulse duration for the generic steel projectile is 1.2 ms, and frequency of 838.1 Hz.

2. METHODOLOGY

It was determined that the most effective approach for evaluating mine seal response to impacts resulting from an accidental explosion was to directly test the product by attempting to recreate representative impact conditions and gather data on the response. The dynamic response was observed by high frequency response strain sensors since the method of loading and estimated impact velocities permitted direct measurement. Surface damage was observed after the seal returned to static equilibrium.

Two full-scale seals were created, one plug seal, and one reinforced seal. The dimensions of the seals are given in Table 2.1.

Table 2.1. 50 psi plug seal dimensions

Dimension	50 psi plug seal	
	(ft)	(m)
Length	10	3.0
Height	6	1.8
Thickness	3.25	1.0

The seals were constructed in a free standing configuration as shown in Figure 2.1. In this configuration, the seals can be represented by a fixed-free-free-free rectangular thick plate.

a. b.

Figure 2.1. Mine seal construction a. Front face b. 50 psi (back) and 120 psi seals (fore)

A powder gun, Figure 2.2, was constructed to generate projectiles. The powder gun, also referred to as the projectile generator or cannon, was fabricated from a 4 inch thick steel tube with an inner diameter of 12 inches (305 mm) and barrel length of 7 ft (2 m).

Figure 2.2. Projectile-generator construction

Concrete projectiles were poured into molds made from layers of 2 inch (50.8 mm) thick foam board, as shown in Figure 2.3. The foam boards act as sabots for the concrete projectiles. The sabots were cut to diameters with a water jet cutter to provide the desired projectile mass for the test projectiles. A wood wad was used to transfer the pressure from the chamber to the projectile and keep the projectile assemblies intact.

a. b.

Figure 2.3. Test concrete projectile fabrication

Powder charges for the cannon, Figure 2.4, were composed of black powder which was centrally positioned in the chamber using foam board sabots. The test cases each used 1 lb (0.45 kg) of black powder, sourced from 1 lb unit cases of Goex® Black Powder, cannon grade.

Figure 2.4. Black powder charge construction

The required charge mass was determined by calculation and derived from the desired muzzle/impact velocity. Muzzle velocities were verified using concrete test projectiles to ensure they were within an acceptable range during qualification testing. The velocity calibration curve, charted against the projectile mass is given in Table 2.2.

Table 2.2. Projectile velocity calibration curve

Projectile mass (lb)	Measured velocity (ft/s)
10	630
20	515
25	507
30	470

To measure the dynamic response of the system, six strain sensors were placed on the rear (non-impact) face of the seal. The locations of the sensors were selected to observe high strain locations, as the intent is to focus the impacts near the center, and also to observe decay behavior of the surface waves. Initial cracks of the seal were noted for comparison after impact test cases. Noted cracks were superficial and a result of the seal fabrication/curing process. The strain sensor installation locations can be seen in Figure 2.5.

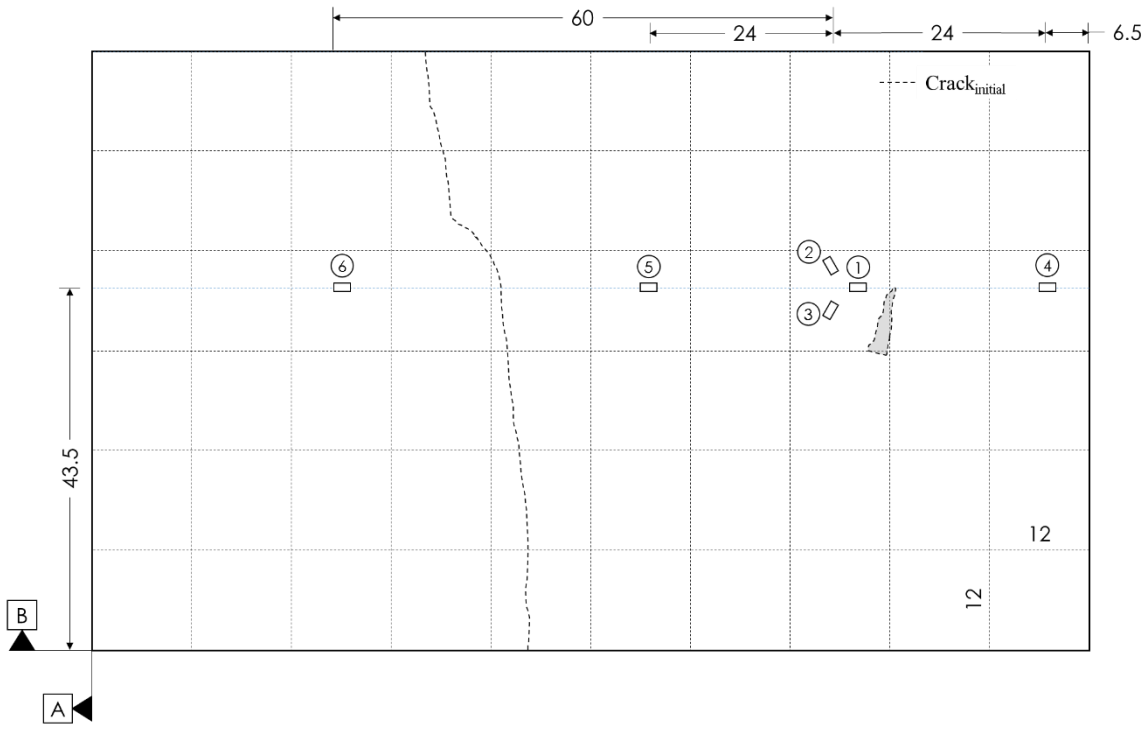


Figure 2.5. 50 psi seal, rear face, initial state and strain sensor installation

A 12 inch by 12 inch (305 mm x 305 mm) grid was marked on the front face of the seal to assist in determining impact location. The initial state of the front face of the mine seal can be seen in Figure 2.6.

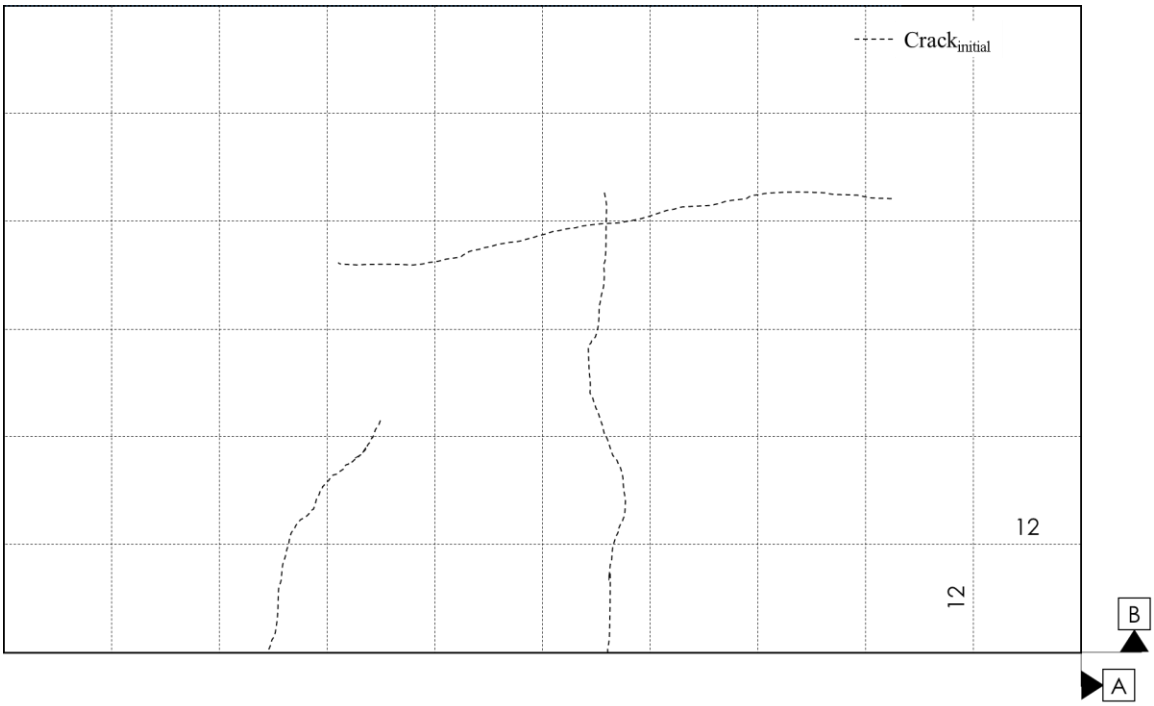


Figure 2.6. 50 psi seal, front face, initial state

The projectiles for the impact test cases were taken from a set of objects that was representative of the types of materials commonly found in an underground coal mine. The projectiles were centrally positioned in the bore of the cannon using foam sabots cut to shape using a water jet cutter and backed by 2 pieces of 3/4 inch (19 mm) plywood wadding positioned between the charge and projectile. The intent of the wadding is to occlude the combustion gases and transfer the energy generated by the pressure rise into kinetic energy in the projectile. A sliding tolerance for the wadding and sabots was used to reduce resistance due to friction and enable loading the cannon from the muzzle end by hand. This was a recognized tradeoff between projectile velocity and the force required to load the projectile, a possible reduction in performance for the former was accepted to enable hand loading.

A laser chronograph was used to measure projectile velocity. Two sensors were spaced parallel with the cannon barrel-line. When a projectile crosses the path of a sensor, the system was triggered and data collection began. The projectile velocity was determined by dividing the displacement between the chronograph sensors by the difference in time between detection of the projectile as measured by the sensor.

The general test configuration can be seen in Figure 2.7.

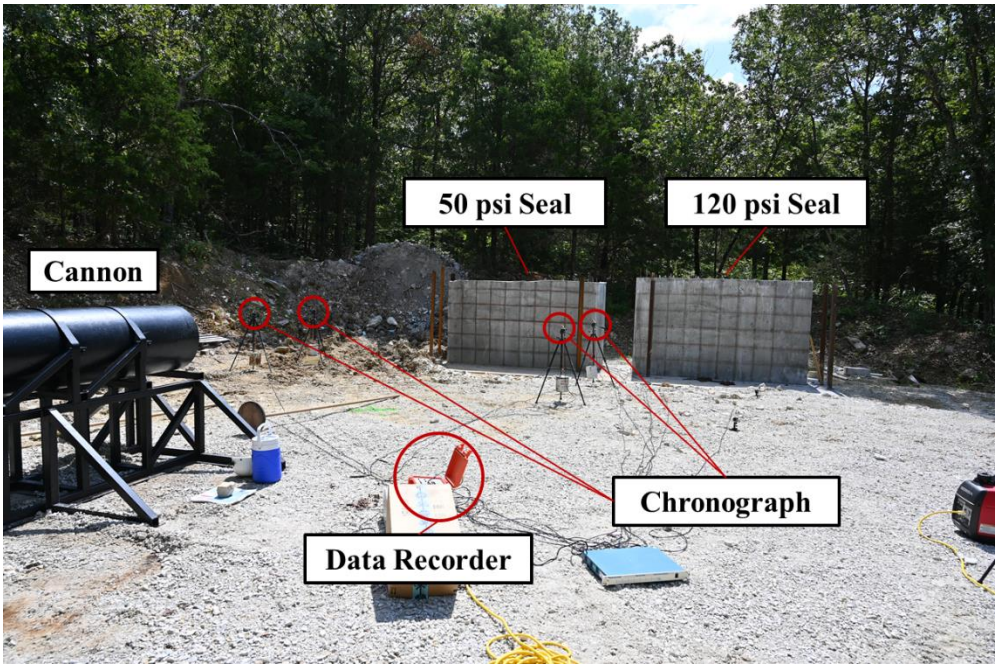


Figure 2.7. General test configuration

Data from the chronographs and the strain sensors were collected using a DataTrap II™ data acquisition system. There were nominally eight channels recorded[7],

[7] Testing on 19 September 2019 did not use the chronographs (channels one and two) as they were frequently obscured during testing by smoke, resulting in velocity measurement error

channels one and two for the chronographs, and the remaining six for the strain sensors, which were sampled at 1 MHz, or 1,000,000 samples per second. From the Hugoniot calculations performed in Section 1.3, the initial estimates of pulse durations are on the order of milliseconds and frequencies approximately 1,000 Hz. It is expected that the test data will provide representative data curves and also avoid aliasing as the sample frequency is greater than twice the highest expected frequency.

The strain sensors are piezoelectric sensors sourced from PCB Piezotronics, model 740B02, with average sensitivity of 50 mV/με. The strain sensors measure the dynamic strain response of the seal directly as they are bonded to the rear face. The sensors provide an output voltage between ±5 volts. Negative voltage outputs indicate a compressive load, while a positive voltage output indicates a tensile load.

3. TEST RESULTS

3.1. VELOCITY MEASUREMENTS

There were a variety of objects shot from the cannon ranging from small and light objects to larger and heavier ones. The as measured projectile velocities exhibited considerable scatter, as shown in Figure 3.1.

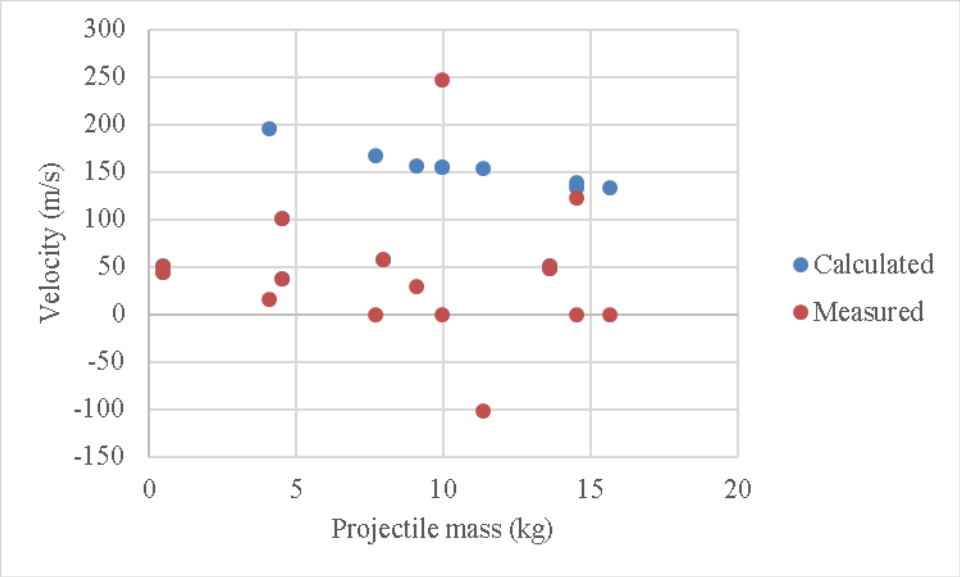


Figure 3.1. Projectile velocities

Some projectiles were not detected and others were sensed at non-representative velocities. Video footage of test cases was reviewed to verify the as-measured values.

In some test cases, the sensors were obscured by combustion products discharged from the barrel that blew by the projectile while it was still in the bore. In these cases, the calibration curve derived from test data during qualification testing of the projectile generator was used to estimate the projectile velocity relative to projectile mass. This led

to a disparity between the measured velocity and the impact velocity for some test cases. In other cases, support material (e.g. sabot) was detected by the chronographs. Lacking verifiable data, the calibration data was used as the default[8].

In two test cases, roof bolts and 25 lb steel rail, had uncharacteristic velocity readings. The method used to determine velocity subtracted the times of first detection between the first and second chronographs and then divided that quantity by the chronograph displacement. In the roof bolts case, it appears that multiple fragmented pieces of the support material were projected into the line of sight of the chronographs while smoke triggered the first chronograph, resulting in an infinitesimal time difference. In the 25 lb steel rail case, support material triggered the second chronograph first and then smoke triggered the first chronograph, resulting in a negative velocity measurement.

3.2. UNREINFORCED 50 PSI SEAL STRAIN RESPONSE

Six strain sensors were placed on the rear face of the unreinforced 50 psi seal. A battery of impact tests was performed on the 50 psi seal, using sets of light[9], medium[10], and heavy[11] projectiles. The volume loss after each test case were measured by a research team using a software algorithm to perform optical analysis. The list of the test cases, with measured and calculated velocity and volume loss data is given in Table 3.1.

---

[8] 10 of the 16 tests relied on velocity estimates from the cannon calibration data.

[9] For projectile masses ranging from 0 to 10 lb (0 – 4.5 kg)

[10] For projectile masses ranging from 10 to 25 lb (4.5 – 11.3 kg)

[11] For projectile masses ranging above 25 lb (11.3 kg)

Table 3.1. Test case identification, impact velocity, and volume loss

ID	Test case	Number of projectiles	Projectile mass (lb) [cumulative]	Velocity (ft/s) [measured]	Velocity (ft/s) [from calibration curve]	Volume loss (in^3)
A	Hard hat	1	1	144.5	733.5	0
B	Hand tools	3	9	53.6	641.5	10.37
C	Roof bolt plates	5	17.5	191.1	559.9	3.60
D	Water jug	1	1	169.0	733.5	0
E	5 in x 5 in (nominal) Lumber	1	30	158.9	470.0	0
F	Roof bolts	4	17	Inf*	549.5	0.67
G	Concrete (10 lb)	1	10	122.1	630.0	0
H	Concrete (20 lb)	1	20	96.6	515.0	0
I	Concrete (30 lb)	1	30	168.2	470.0	0

Table 3.1. Test case identification, impact velocity, and volume loss (cont.)

J	Steel penetrator (10 lb)	1	10	333.4	630.0	40.8
K	Steel rail (25 lb)	1	25	-333.7*	506.0	1.10
L	Steel penetrator (20 lb)	1	22	811.7	511.8	10.19
M	Steel penetrator (30 lb)	1	32	404.5	455.2	24.7
N	Steel rail (35 lb)	1	34.5	N/A**	437.6	3.2
O	Steel penetrator (20 lb)	1	22	N/A**	511.4	10.2
P	Steel penetrator (30 lb)	1	32	N/A**	455.60	4351.1

* - these values are identified as non-representative of the projectile and are a result of a sensor detection error, likely smoke or different fragments captured by the sensor
** - these values are not available as a result of the chronographs not used for this test

The raw data from the test cases can be referenced in APPENDIX A, a selection of data sets will be presented in Section 4.3.

A diagram of the impact locations on the front face of the seal for each test case and the observed damage on the front and rear faces are given in Figure 3.2 and Figure 3.3 respectively.

The legend for the test cases designated in Figure 3.2 is found in Table 3.1. The number following the test case identifier in Figure 3.2 is an indication of the wad impact location. For some test runs, there were multiple wads which translated to multiple impacts identified for a single test case.

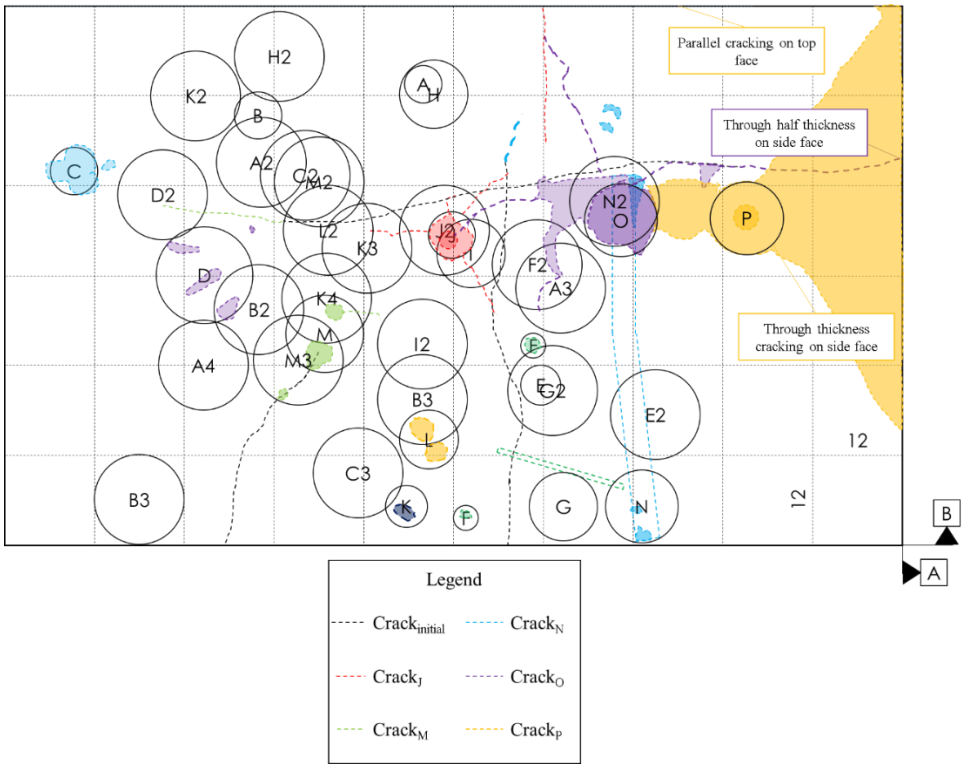


Figure 3.2. Front face projectile impact locations and observed damage

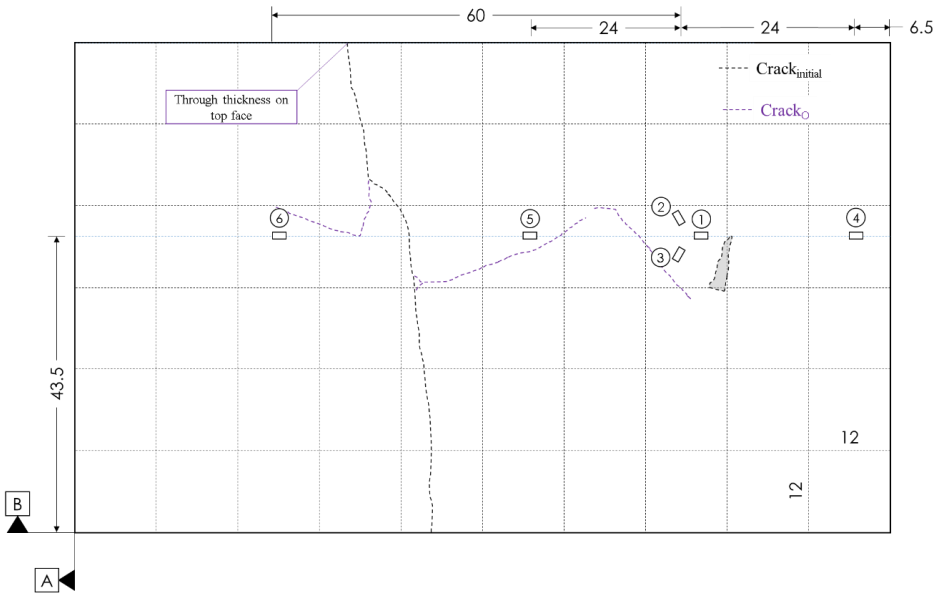


Figure 3.3. Rear face observed damage

For each case, the impact location was determined by observation after the test was run. Where there was cratering or chipping, the impact location was obvious, but in all cases, a trace of the projectile and/or wadding was visible on the impact face. This was generally in the form of a shadow mark (often soot) transferred from the projectile to the impact face.

In a number of cases, the smoke obscured the seal face and/or projectile such that the relative impact location could not be confirmed by a review of the video. In these cases, only the photograph of the seal face was used to determine impact location, as depicted in Figure 3.4. A comparison of the photographs between the current and previous test case enabled accurate identification of impact locations.

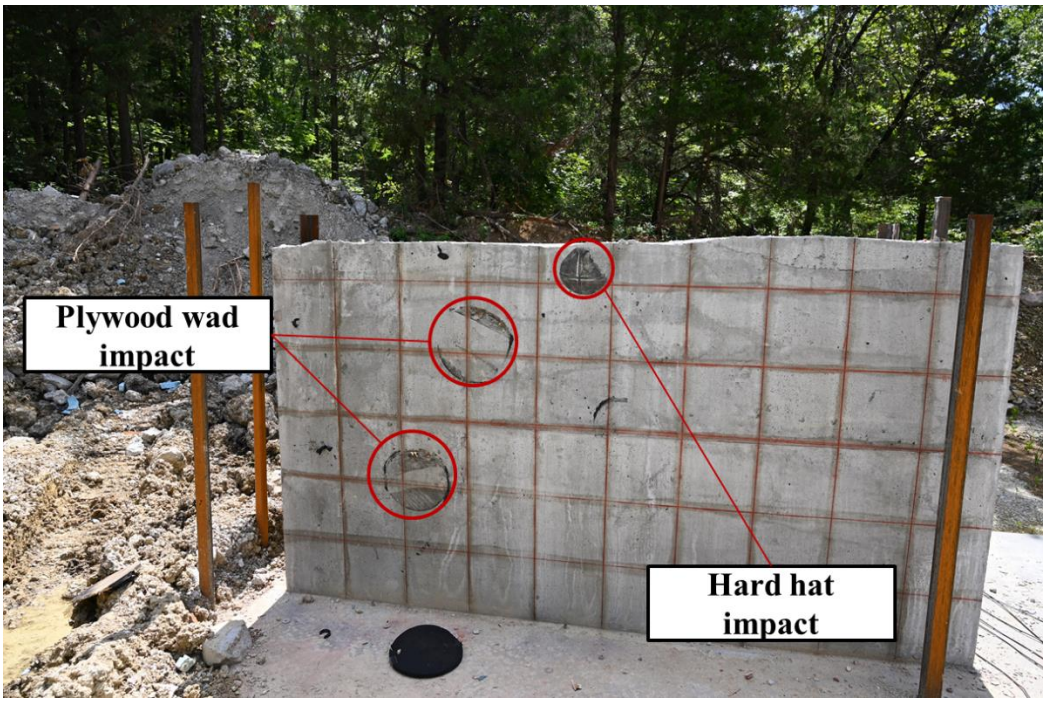


Figure 3.4. Identifying impact locations post test run

The light projectiles generally contributed minimal damage to the seal, with the notable exceptions of the 10 lb steel penetrator and the hand tools test cases. The 10 lb steel penetrator accounted for the third greatest volume loss during testing. The medium projectiles contributed marked damage to the seal. In some cases, penetration was observed on the impact face of the seal and minor cracking too the impact face. The 20 lb projectile (19 Sep 19) was the only test to result in visible cracking on the rear face. The heavy projectiles resulted in a greater total volume loss overall than all other projectile types and generally exhibited penetration, cracking, and potential spalling. For a visual comparison of the volume loss between projectile weight classes, refer to Figure 3.5. The steel projectiles contributed over 99% of all observed volume loss.

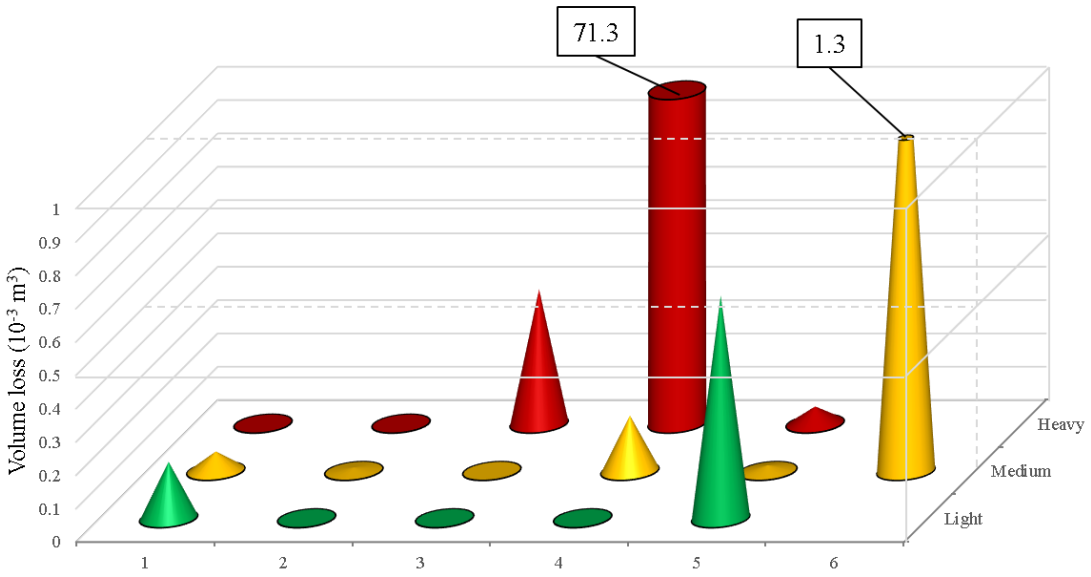


Figure 3.5. Volume loss comparison between projectile weight classes

In Figure 3.5, the test cases are grouped by weight class. The test cases for the light projectiles are: 1) hand tools, 2) hard hat, 3) water jug, 4) 10 lb concrete, and 5) 10 lb steel penetrator. For the medium projectiles, the test cases are: 1) roof bolt plates, 2) roof bolts, 3) 20 lb concrete, 4) 20 lb steel penetrator (6 Sep 19), 5) 25 lb steel rail, 6) 20 lb steel penetrator (19 Sep 19). For the heavy projectiles, the test cases are: 1) 5 x 5 lumber, 2) 30 lb concrete, 3) 30 lb steel penetrator (6 Sep 19), 4) 30 lb steel penetrator (19 Sep 19), 5) 35 lb steel rail.

During testing for medium and heavy projectiles on 19 September 2019, which included the steel rail (35 lb), and steel penetrators (20 lb and 30 lb), the seal suffered catastrophic failure, as shown in Figure 3.6.

Figure 3.6. 50 psi seal catastrophic failure

The final test case, the steel penetrator (30 lb), produced the greatest volume loss for all other test cases combined.

3.3. PROJECTILE AND NON-PROJECTILE IMPACTS

The strain sensors are agnostic to the method and type of loading which effects the deformation in the seal. This can introduce challenges with interpreting the data if one does not have a mechanism to discriminate between authentic and spurious data. As part of the means of delivering the projectile to the seal, support materials (wadding, sabots) were introduced to transfer the combustion energy to projectiles and also to position them in the bore, as can be seen in Figure 3.7.

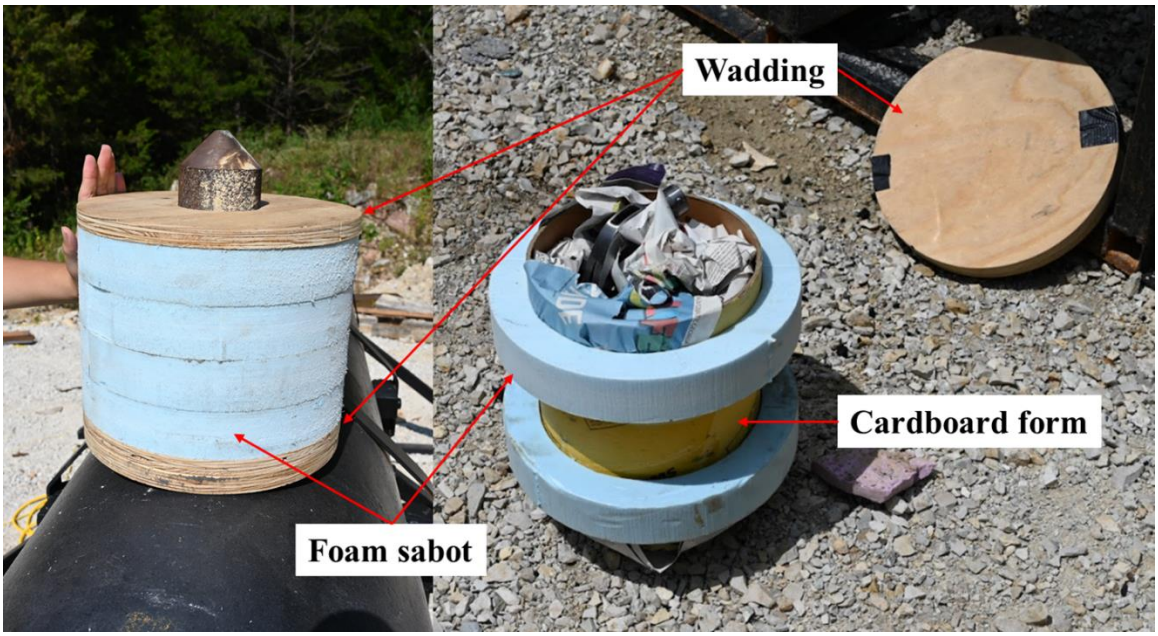


Figure 3.7. Typical projectile support materials

In some instances, the support materials, or fragments of them, would impact the seal prior to the projectile. This can be observed in the sensor readouts from typically small amplitude strain spikes prior to the projectile impact strain spike. In some cases, the extraneous impacts occurred so far in advance of the projectile and imparted sufficient energy to induce observable structural response in the seal such that measurements indicated a bias wave that persisted through projectile impact and data capture. Comparing the sensor readouts and video recording enabled the discrimination between multiple impacts and projectile and non-projectile impacts. One illustration of this is with the 25 lb steel rail test case. As the figures below indicate, the sequence of events shows multiple smaller impacts prior to the projectile impact. In cases where multiple non-projectile impacts occurred, the relative strain amplitudes and durations between the impacts in the data readout is apparent.

a.

b.

c.

d.

e

f

Figure 3.8. Impact sequence for 25 lb steel rail

With video recording, the sequence of events helps to paint a more complete picture and help explain the bias wave in some data sets, an example shown in Figure 3.9, which results from support materials (e.g. wadding) impacting the seal at approximately 8 ms, prior to the rail impact, which occurs at approximately 22 ms.

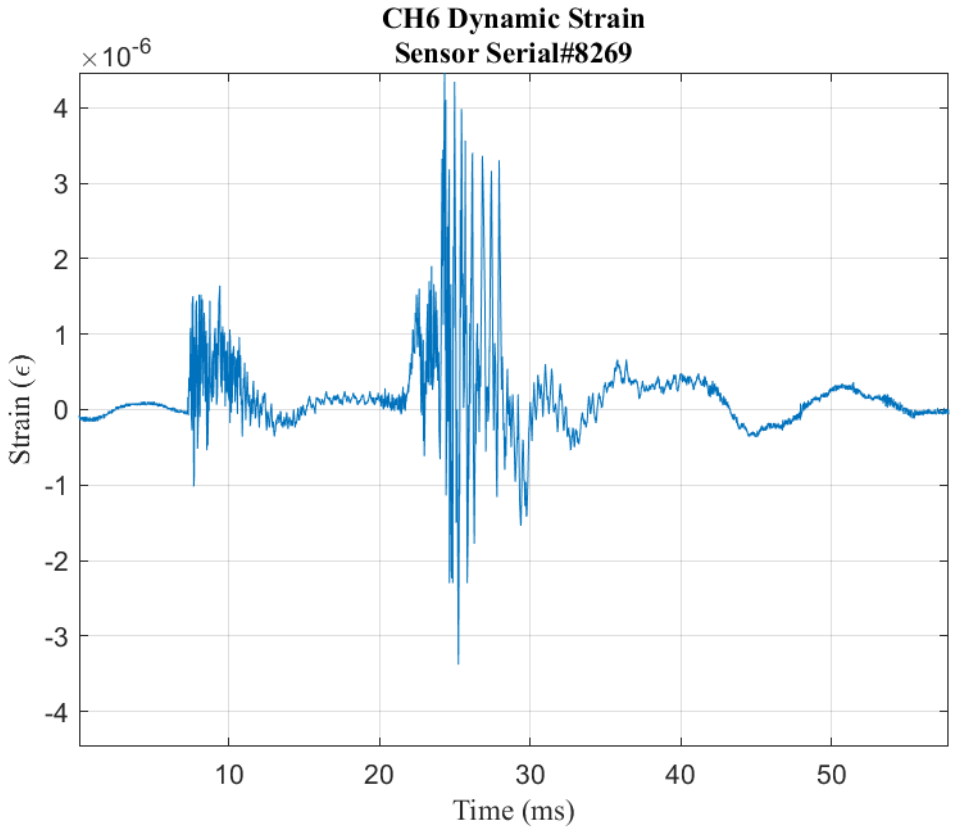


Figure 3.9. Dynamic response bias wave

4. DISCUSSION

4.1. HUGONIOT EQUATIONS

From the velocity data and material properties, the pressure-particle velocity (P-u) Hugoniot can be represented by (Meyers, 1994):

$$P = \rho U u = \rho(C_0 + Su)u = \rho C_0 u + \rho S u^2 \tag{29}$$

As may be noted from above, the P-u Hugoniot utilizes the shock velocity-particle velocity (U-u) Hugoniot. This general equation is applied to both the target and projectile material and the state variables for the interface are determined. The shock pulse can be estimated by taking the interface pressures from the initial impact and successive rarefaction fronts from the projectile. The shock impedance, *Z*, which is determined by the quantity:

$$Z = \rho U \tag{30}$$

The shock impedance provides an indication of the shock pulse shape. For projectiles with a shock impedance higher than the target, the projectile will penetrate the material and the pressure wave in the projectile will incrementally decay with each rarefaction wave from the free surface. As a result, the transmitted pressure pulse into the target will have a steep front and stepped tail. For projectiles with an equal or lower shock impedance than the target, the projectile will reflect from the target after the initial projectile rarefaction wave front interfaces with the target. This results in a square pressure pulse.

The Hugoniot relations were developed to estimate the interface pressure, predict if penetration will occur, and derive a theoretical impulse displacement-time response of

the system. The shock impedance estimates accurately predicted penetration in all cases except in the roof bolt plates, 20 lb concrete, 10 lb steel penetrator, 30 lb steel penetrator (6 Sep 19), and 35 lb steel rail. In all cases but the 20 lb concrete, the shock impedance predicted no penetration.

4.2. IMPACT LOCATION

For a plate clamped on all edges, a point load at the center of the plate would result in the greatest total displacement of the plate. This is intuitive, as this point is the farthest from each support. While the test setup involves a clamped-free plate and the greatest displacement would be realized by a point load on the free edge opposite the constraint. A central impact was determined to be the preferred case to investigate in order to obtain insight on the stress wave behavior on the rear face of the seal since the response immediately after impact would not be significantly different between a clamped-clamped and a clamped-free plate. This is because the stress waves originating from a central location would not reach the free or clamped edge before reaching the majority of the strain sensors on the initial pulse. After the stress waves arrive at the edges and reflect, the edge effects, whether clamped or free becomes non-trivial. A true center impact is difficult to achieve in practice due to the limiting conditions of the test setup. As a result, all the impacts will exhibit some level of eccentricity, loading the seal asymmetrically. The projectile generator bore was aligned with the desired impact area, charged with propellant and projectile, and then fired. The varied types of projectiles contributed to diverse impact locations and number of impacts. While the variability in test cases lends to challenges in making strict comparison between impact test cases, it

does provide a more representative data set for impacts to mine seals in use, since it is assumed that impact locations in an accidental mine explosion would be distributed randomly on the seal face. As a result, impact response based on varied locations and number of impacts was measured.

The locations of the projectile craters and wad impacts are measured from the datums indicated in Figure 2.6 displayed in Table 4.1:

Table 4.1. Projectile impact locations

Test case	Impact location [x] (in) [from Datum A]	Impact location [y] (in) [from Datum B]
Hard hat	63.0	61.5
	82.8	49.5
	45.6	33.6
	93.0	22.8
Hand tools	111.6	49.5
	109.5	48.0
	109.8	51.0
	78.0	49.5
	73.5	9.0
Roof bolt plates	96.0	41.4
	93.0	35.4
	90.0	30.0
	97.5	46.8

Table 4.1. Projectile impact locations (cont.)

Water jug	85.5	57.6
	7.1	31.2
	102.0	6.0
5 in x 5 in nom. Lumber	48.8	21.0
	33.0	17.4
Roof bolts	57.6	3.6
	49.5	27.0
	46.2	10.8
	48.8	37.5
Concrete (10 lb)	45.0	6.0
	46.8	21.0
Concrete (20 lb)	62.4	60.0
	82.8	65.4
Concrete (30 lb)	57.6	40.0
	64.8	27.0
Steel penetrator (10 lb)	61.5	41.4
	62.4	42.0
Steel penetrator (20 lb)	64.8	15.6
	63.6	12.8
	76.8	34.2

Table 4.1. Projectile impact locations (cont.)

Steel penetrator (30 lb)	76.8	31.8
	78.0	26.4
	82.8	21.0
	76.8	33.6
Steel rail (25 lb)	67.2	4.8
	94.8	60.0
	71.4	39.6
	76.8	33.6
Steel rail (35 lb)	33.6	1.5
	37.5	47.4
Steel penetrator (20 lb)	64.8	15.6
	63.6	12.75
	76.8	34.2
Steel penetrator (30 lb)	21	43.2

4.3. STRAIN AND STRAIN RATE

A MATLAB® script was developed to take the input strain sensor data, convert to strain, and determine strain rate in addition to all other general data processing. The dynamic strain of the system was derived by taking the sensor time-voltage response and converting the voltage to strain by applying the sensitivity of each sensor to the measured voltage values. The sensitivity of each strain sensor was adjusted to concrete since the

material used in calibrating the sensors was steel and concrete has a sensitivity less than steel (PCB Piezotronics). Each sensor sensitivity was adjusted based on serial number and reported sensitivity defined in the calibration certificate. The calibration curve used for adjusting the sensor sensitivity relative to the modulus of elasticity is given in Table 4.2, adapted from (PCB Piezotronics).

Table 4.2. Sensor sensitivity and sensed material elasticity modulus

Material	Modulus of elasticity, *E* (psi)	Sensitivity (mV/με)
Steel	30e6	50
Aluminum	10e6	41
Acrylic	0.4e6	5

The normal strains ($\varepsilon_{xx,}$ ε_{yy}) for each sensor location were derived from the sensor measurements with the applied sensitivity correction. The delta rosette, with configuration shown in Figure 4.1, was used to determine the average state of strain at location A (ε_{xx}, ε_{yy}, ε_{xy}). The angle between sensors was 120 degrees. The strain relations, Eqs. 30 – 32, used in the analysis, are given below, for a delta rosette (Boresi & Schmidt, 2003). These relations convert the strain sensor measurements to strain in the primary coordinate system (i.e. x, y). These strains can then be converted into strain in the principal axes.

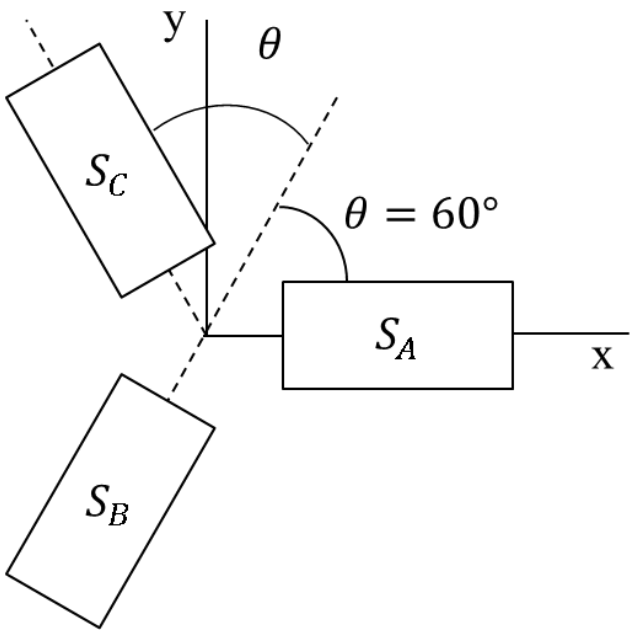


Figure 4.1. Strain sensor delta rosette configuration

$$\epsilon_1 = \epsilon_a \tag{31}$$

$$\epsilon_{yy} = \frac{2(\epsilon_b + \epsilon_c) - \epsilon_a}{3} \tag{32}$$

$$\epsilon_{xy} = \frac{\epsilon_b - \epsilon_c}{\sqrt{3}} \tag{33}$$

For each sensor, the strain-rate was derived from the strain-time response. The strain-rate is obtained by taking the difference between the measured strain for each time step and the previous for each time step, as in Eq. 34.

$$\dot{\epsilon} = \frac{\epsilon_i - \epsilon_{i-1}}{t_i - t_{i-1}} \tag{34}$$

The strain-rates ($\dot{\epsilon}$) for the test cases are given in Table 4.3, including the impact velocity used for the test case.

Table 4.3. Test cases and derived peak strain rate

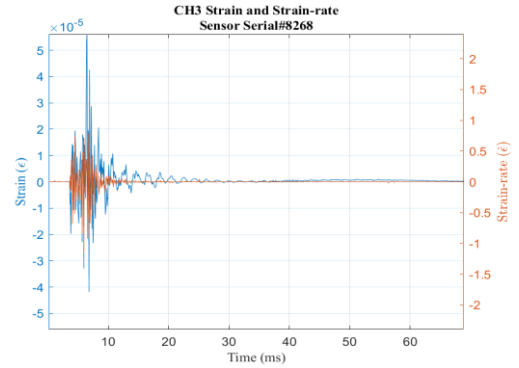
Test case	Velocity (ft/s)	$\dot{\epsilon}$ (s^{-1})
Hard hat	144.5	0.4
Hand tools	641.5	1.2
Roof bolt plates	191.1	0.5
Water jug	169.0	0.7
5 in x 5 in nom. Lumber	158.9	0.1
Roof bolts	549.5	1.4
Concrete (10 lb)	122.1	0.3
Concrete (20 lb)	515.0	0.7
Concrete (30 lb)	168.2	0.9
Steel penetrator (10 lb)	333.5	0.2
Steel penetrator (20 lb)	511.40	1.1
Steel penetrator (30 lb)	404.53	1.6
Steel rail (25 lb)	506.0	0.5
Steel rail (35 lb)	437.6	1.5
Steel penetrator (20 lb)	511.4	2.9
Steel penetrator (30 lb)	455.6	21.5

The velocity and strain-rate ranges observed appear in the range, albeit on the low side, with those identified in the research. Zukas observed velocities ranging from 164 – 1640 ft/s (50 – 500 m/s) the strain-rate ranges from 1 to 100 s^{-1} (Zukas, Impact

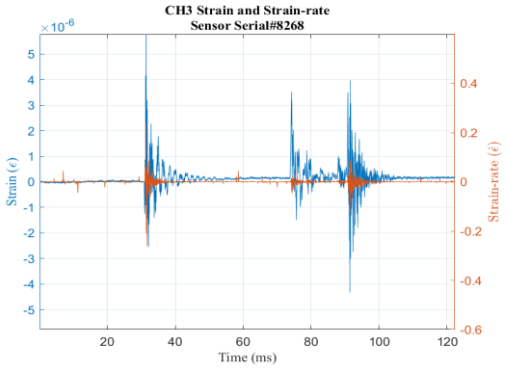
Dynamics: Theory and Experiment, 1980). The final 30 lb steel penetrator test produced a strain-rate of 21.5 s^{-1} which seems to fit better with expected range of values from the research. It should be noted that the values referenced from the research are intended to be for reference and not used as a rule for correlating velocity and strain-rate since there are myriad other parameters that have a direct impact on the strain-rate. Since the strain was measured on the non-impact face, the thickness or relative projectile to target mass will have a marked impact on the structural deflection and hence strain-rate as the stress waves decay through the target. The strain and strain-rate on the impact face was not determined and it was on this face that the majority of cracking was observed.

Applying knowledge of fracture mechanics to the failure modes exhibited by the mine seal helps to explain the observed cracking behavior. In addition to micro-cracks in the structure, the mine seal contained initial surface macro-cracks. This facilitated crack growth and enabled the structure to proceed beyond the early stages of micro-crack growth and into macro-crack propagation because of existing macro-cracks. The stress concentrations near the crack boundaries and especially near the termination points encouraged macro-crack propagation when subjected to tensile loading. In addition, as found in the research, when subjected to impact loading and sufficient strain-rates, the seal exhibited multiple macro-cracks activated simultaneously (Cadoni, Albertini, Labibes, & Solomos, 2001).

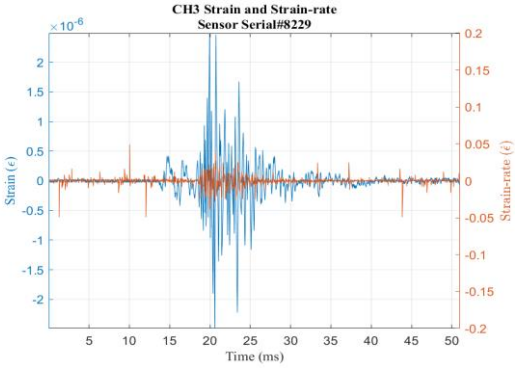
The strain and strain-rate were plotted against time for each test case, the data recorder reported the data by channel, which was then matched to sensor serial and verified for each test set. The data set for each test case, with larger figures, can be referenced in APPENDIX A, a selection of test cases can be seen in Figure 4.2.



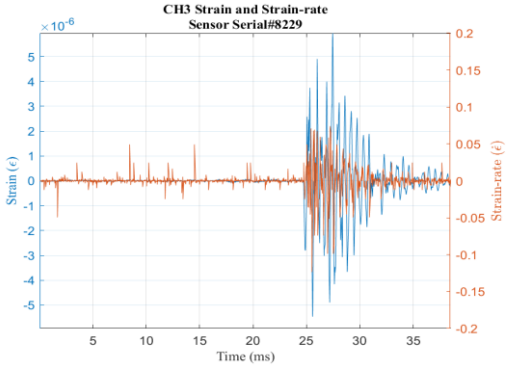
a.



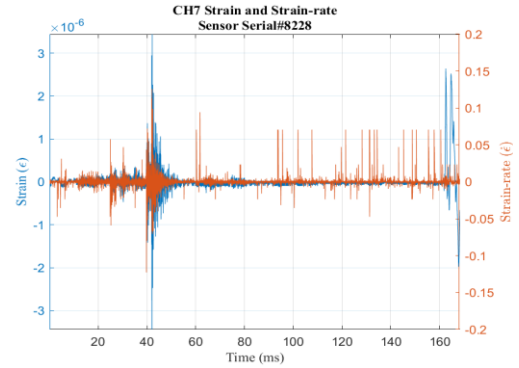
b.



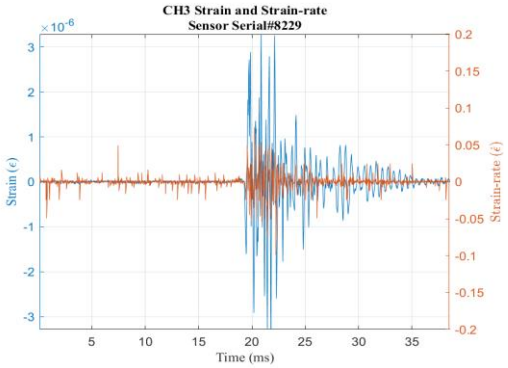
c.



d.



e.



f.

Figure 4.2. Strain and strain-rate for selected test cases: a. Hand tools b. Hard hat c. Concrete (10lb) d. Concrete (30lb) e. 5 in x 5 in nom. Lumber f. Steel penetrator (10lb)

4.4. STRESS STATE

The MATLAB® script took input material properties and calculated the stress state for each test case. The stress state of the rear face of the seal was determined by adapting the strain data to the stress-strain relations for a linearly elastic isotropic material in plane strain. The coordinate system used can be seen in Figure 4.3.

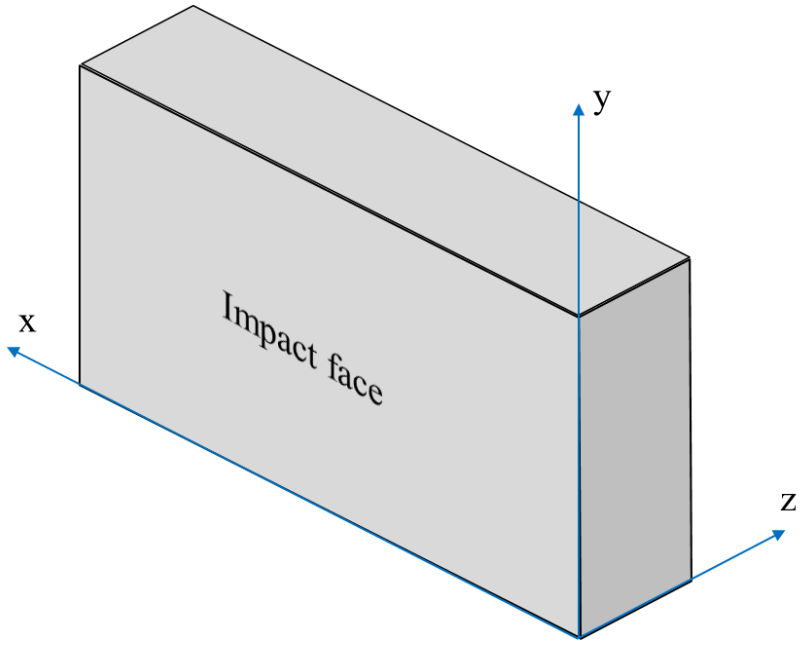


Figure 4.3. Mine seal coordinate system

The plane strain stress-strain relations can be referenced in Eqs. 35 – 38.

$$\sigma_{xx} = \frac{E}{(1+\nu)(1-2\nu)}\left((1-\nu)\epsilon_{xx} + \nu\epsilon_{yy}\right) \tag{35}$$

$$\sigma_{yy} = \frac{E}{(1+\nu)(1-2\nu)}\left(\nu\epsilon_{xx} + (1-\nu)\epsilon_{yy}\right) \tag{36}$$

$$\sigma_{zz} = \frac{\nu E}{(1+\nu)(1-2\nu)}\left(\epsilon_{xx} + \epsilon_{yy}\right) \tag{37}$$

$$\sigma_{xy} = \frac{E}{1+\nu}\epsilon_{xy} \tag{38}$$

where σ is the stress component for the subscripted axes, E is the modulus of elasticity, ν is Poisson's ratio, and ϵ is the respective strain component.

The principal stress was determined by finding the roots to the system of equations for the stresses on the principal strains, using the stress invariants I:

$$\sigma^3 - I_1\sigma^2 + I_2\sigma - I_3 = 0 \tag{39}$$

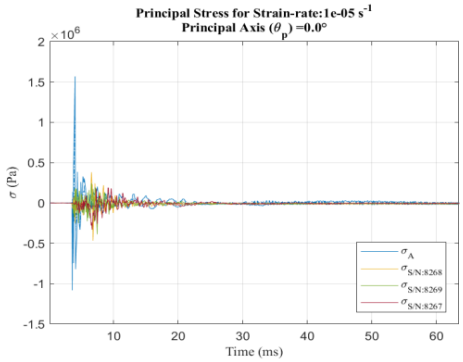
$$I_1 = \sigma_{xx} + \sigma_{yy} + \sigma_{zz} \tag{40}$$

$$I_2 = \sigma_{xx}\sigma_{yy} + \sigma_{xx}\sigma_{zz} + \sigma_{yy}\sigma_{zz} - \sigma_{xy}^2 - \sigma_{xz}^2 - \sigma_{yz}^2 \tag{41}$$

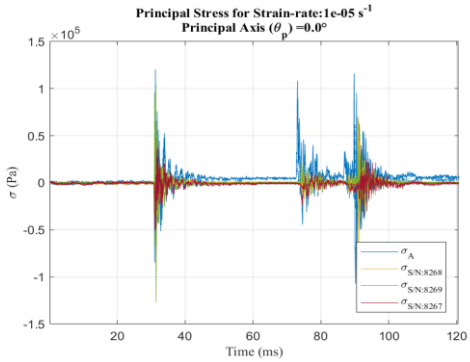
$$I_3 = \begin{vmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{vmatrix} \tag{42}$$

From the solutions, it was observed that σ_{zz} was non-zero but was negligible compared to the magnitudes of either σ_{xx} or σ_{yy}. The derived principal stresses, which were determined for each test case, were applied in the failure prediction models. Principal stresses for a selection of test cases is shown in Figure 4.4 and also in Figure 4.5 for principal stresses at specified strain-rates, which was used in determining failure criteria for strain-sensitive behavior. The strain-rate bins used to separate the data were derived from the estimated strain sensitive strength categories defined later in Table 4.4, these were determined by applying the limits shown in Eq. 43.

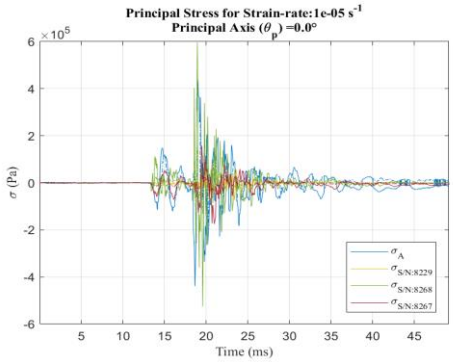
$$\dot{\epsilon}_{bin} = \begin{cases} 10^{-6} & for\ \dot{\epsilon} \le 5^{-2} \\ 10^{-2} & for\ 5^{-2} < \dot{\epsilon} \le 0.5 \\ 1 & for\ 0.5 < \dot{\epsilon} \le 5 \\ 10 & for\ 5 < \dot{\epsilon} \end{cases} \ (s^{-1}) \tag{43}$$



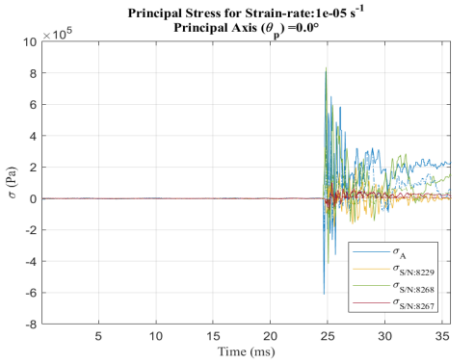
a.



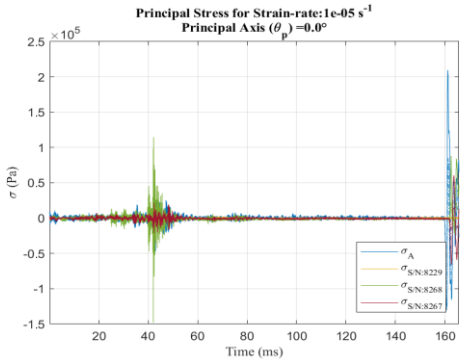
b.



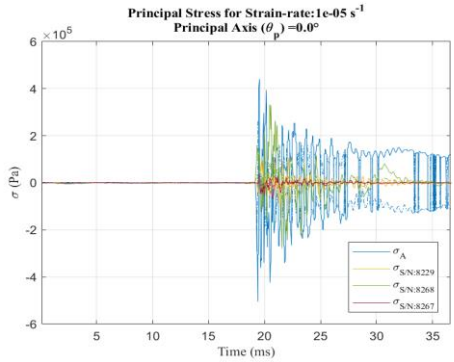
c.



d.

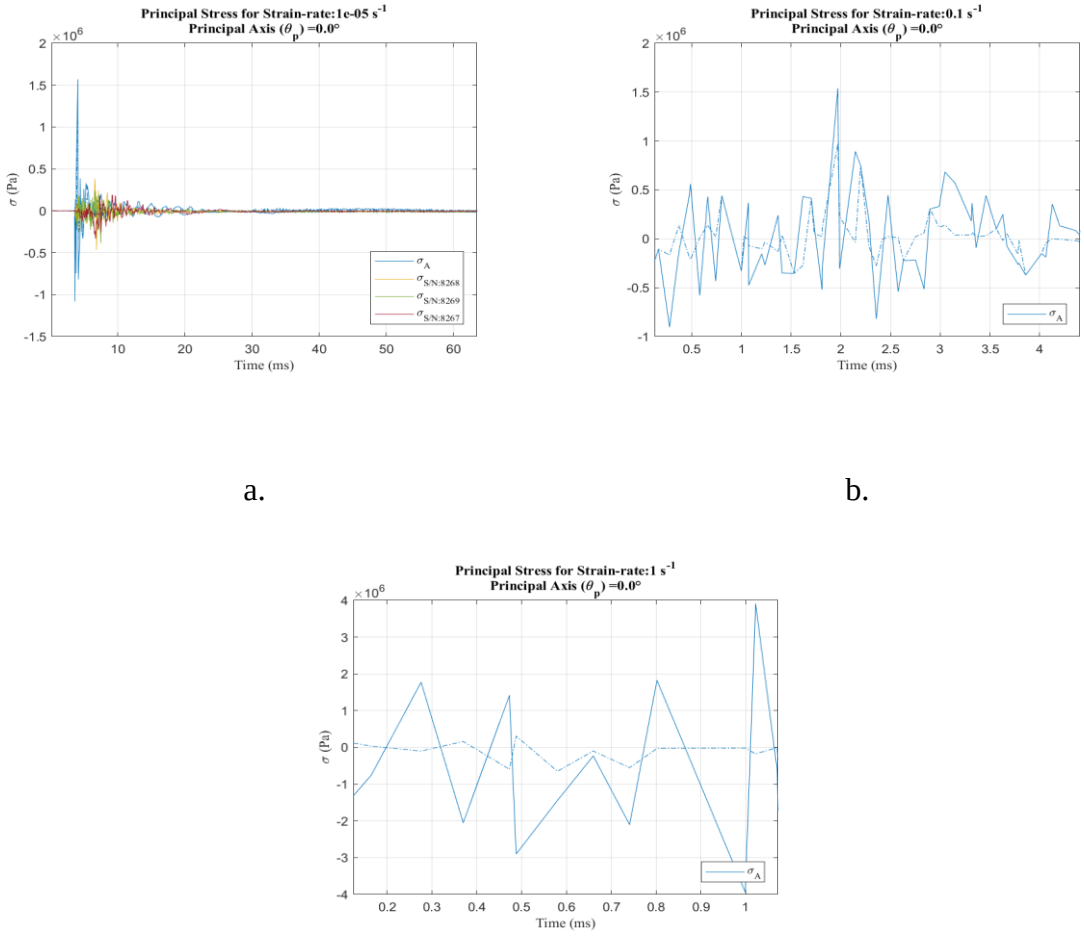


e.



f.

Figure 4.4. Principal stresses for selected test cases: a. Hand tools b. Hard hat c. Concrete (10lb) d. Concrete (30lb) e. 5 in x 5 in nom. Lumber f. Steel penetrator (10lb)



a. b.

c.

Figure 4.5. Principal stresses at defined strain-rates, Hand tools test case

4.5. FAILURE MODEL

The failure of the rear face was estimated by using the maximum normal stress criteria (Coulomb) and the Mohr-Coulomb failure criteria. The maximum normal stress criterion, which is well suited for brittle materials (Boresi & Schmidt, 2003) is represented by:

$$1 \geq \frac{\sigma_1}{F_{ty}} \tag{44}$$

$$1 \geq \frac{\sigma_3}{f_c'} \tag{45}$$

where F_{ty} and f_c' are the tensile yield and compressive strengths of the material, and σ_1 and σ_3 are the maximum and minimum principal stresses.

The Mohr-Coulomb failure criteria is specifically suited for cohesive materials, which includes rock and concrete (Boresi & Schmidt, 2003). For this criterion, the yield criteria for tension and compression can be determined by:

$$Y_T = \frac{2c\cos\phi}{1+\sin\phi} \tag{46}$$

$$Y_C = \frac{2c\cos\phi}{1-\sin\phi} \tag{47}$$

where Y_T and Y_C are the yield limits for tension and compression respectively, c is the cohesion property, and φ is the angle of internal friction. For materials where these values are readily available, the yield stress calculations are straightforward, otherwise the parameters can be determined by:

$$c = \frac{1}{2}\sqrt{Y_T Y_C} \tag{48}$$

$$\phi = \sin^{-1}\left(\frac{Y_C - Y_T}{Y_C + Y_T}\right) \tag{49}$$

These failure criteria were used to develop a failure envelope for each test case, against which the state of stress was plotted. As concrete is strain-rate sensitive, the strain-rate dependent strengths (i.e. tensile strength) were determined for each strain-rate, categorized into strain-rate bins (i.e. 10^{-6} s^{-1}, 10^{-2} s^{-1}, 1 s^{-1}, and 10 s^{-1}), and that strength

was used to determine the envelope. The strengths for each bin were referenced from (Cadoni, Albertini, Labibes, & Solomos, 2001), shown in Table 4.4.

Table 4.4. Strain-rate effect on tensile strength, adapted from (Cadoni, Albertini, Labibes, & Solomos, 2001)

Strain-rate (s^{-1})	Tensile strength (psi)
10^{-6}	522.1
10^{-2}	638.2
1	1073.3
10	1522.9

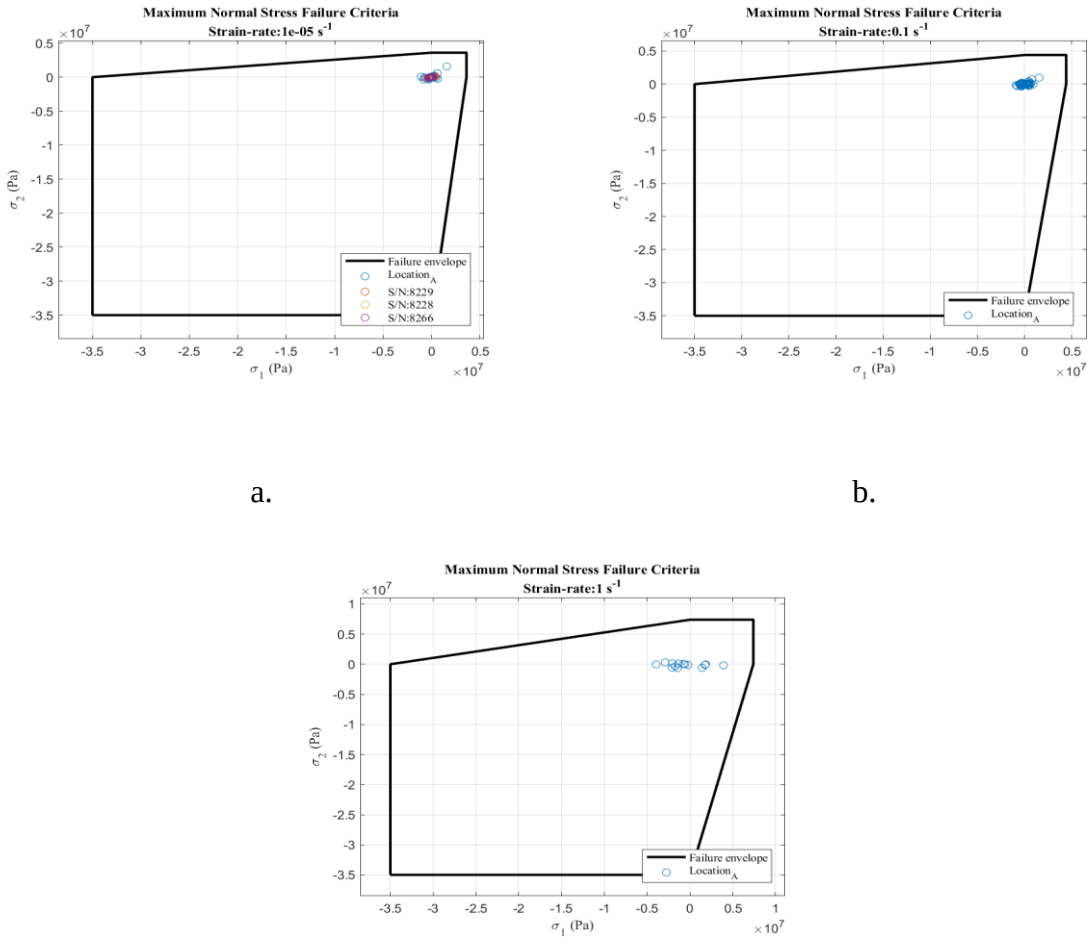
From Table 4.4, it can be seen that there is a positive correlation between strain-rate and the tensile strength of concrete. At the highest strain-rate, the dynamic tensile strength is nearly three times that of the quasi-static strength.

For each test case, the state of stress was derived from the strain data using the stress-strain relations. From the state of stress, the principal stresses were determined and separated into the strain-rate bins using the same limits defined in Section 4.4. Then the failure criteria were applied to the principal stress state for each test case. For both the Coulomb and Mohr-Coulomb criteria, if a principal stress set falls outside the envelope, failure is indicated. The Coulomb and Mohr-Coulomb failure predictions and qualitative observations are shown in Table 4.5.

Table 4.5. Failure prediction and observed failure behavior

Test case	Coulomb criteria	Mohr-Coulomb criteria	Observed failure
Hard hat	No	No	None
Water jug	No	No	None
Hand tools	No	Yes	Cratering
Roof bolt plates	No	No	Cratering
5 x 5 lumber	No	No	None
Roof bolts	No	No	Cratering
Concrete (10 lb)	No	No	None
Concrete (20 lb)	No	No	None
Concrete (30 lb)	No	Yes	None
Steel penetrator (10 lb)	No	Yes	Cratering, surface cracking
Steel rail (25 lb)	No	Yes	Cratering
Steel penetrator (20 lb)	No	Yes	Cratering
Steel penetrator (30 lb)	No	Yes	Cratering, surface cracking
Steel rail (35 lb)	No	No	Cratering, surface cracking
Steel penetrator (20 lb)	No	No	Cratering, spallation, cracking (front/side/top/rear)
Steel penetrator (30 lb)	No	Yes	Tunneling, cracking (front/side/top), fragmentation

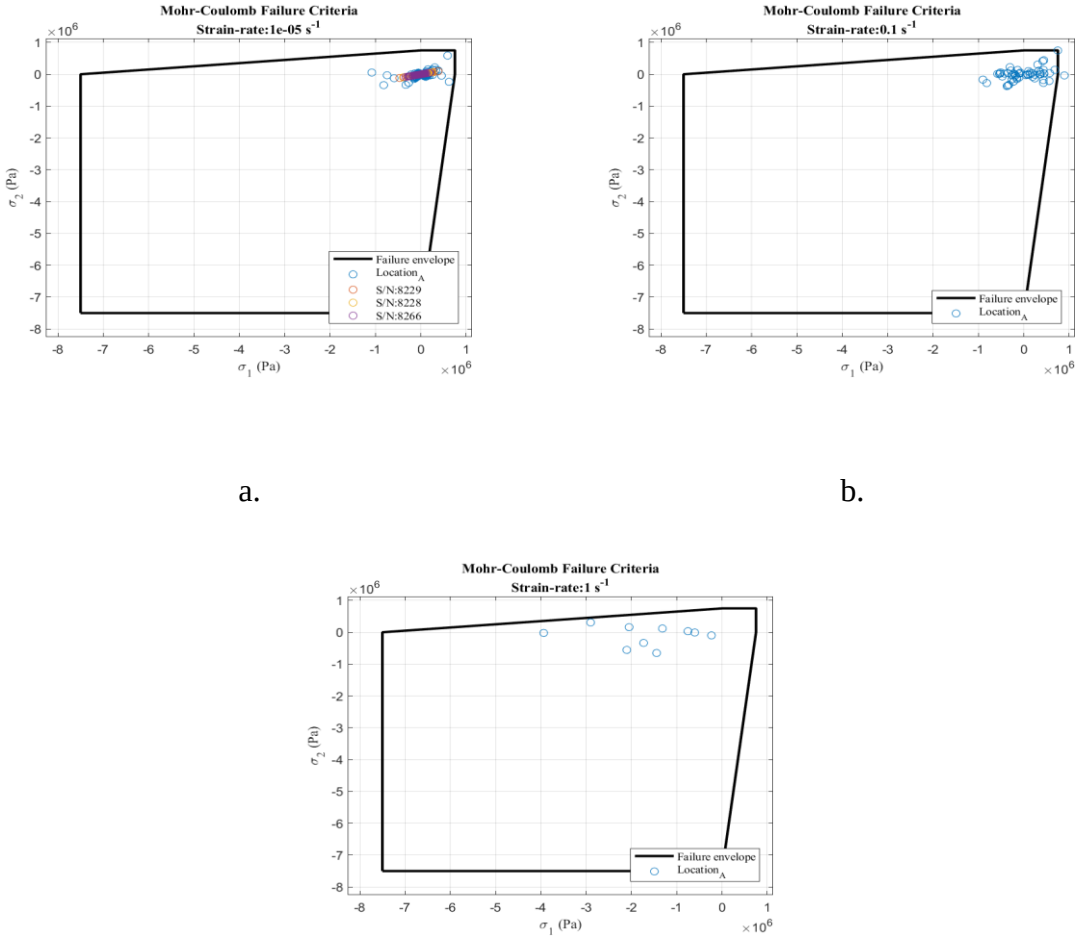
The predicted failure criteria, maximum normal stress and Mohr-Coulomb, were plotted for each case, and are available in APPENDIX A, the plots for the Hand tools test case are shown in Figure 4.6 and Figure 4.7.



a. b.

c.

Figure 4.6. Maximum normal stress failure criteria for the Hand tools test case



a. b.

c.

Figure 4.7. Mohr-Coulomb failure criteria for Hand tools test case

The failure predictions pertain to the strain sensor locations, and with this consideration, only the maximum normal stress failure criteria accurately predicted failure for these locations, since there was no cracking observed at the strain sensor locations. However, at a system level view, the localized failure predictions did not

accurately predict failure in the seal as there was observed failure for a number of test cases and even rear face cracking for the 20 lb steel penetrator.

From the estimated failure criteria, and by visual inspection of the seal after testing, there is notable disagreement between the Mohr-Coulomb failure criteria and the maximum normal stress failure criteria predictions. This could be due, in part, to the cohesion and friction angle parameters used for the analysis. These parameters did not adjust for strain rate sensitive behavior, which is a known behavior for concrete. However, for this and other test cases that failure was predicted could have exhibited internal fracturing which would not have been apparent via visual inspection. This is especially apparent when reviewing the data from the 30 lb steel penetrator test case from 19 Sep 19. The strain response plot shows a discontinuity, as indicated in Figure 4.8.

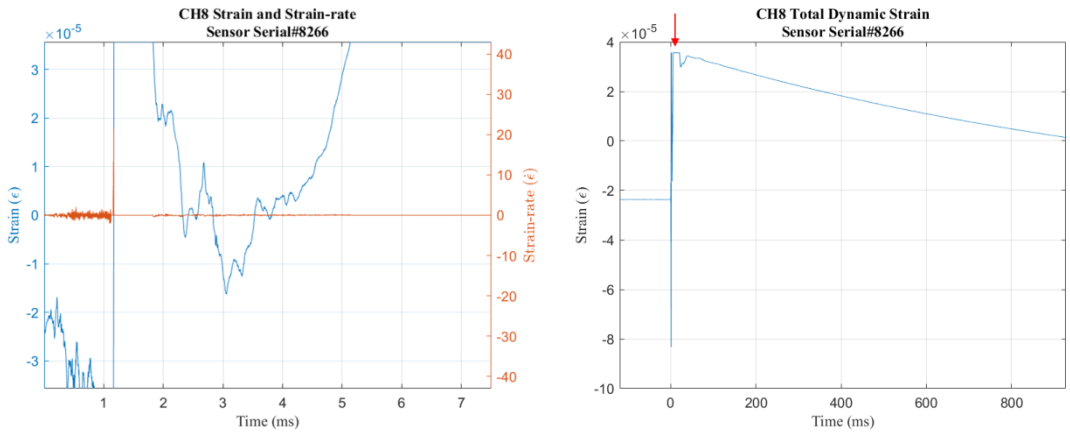


Figure 4.8. Strain response discontinuity for steel penetrator, 30 lb, test date: 19 Sep 19

At the transition point from compressive to tensile loading, the strain rate reaches a peak level of 21.5 s^{-1}. At this strain-rate, the tensile strength is estimated to be 1,523 psi (10.5 MPa). However, the estimated peak principal stresses at this location were only

172.3 psi (1.2 MPa) and 101.5 psi (0.7 MPa) which do not exceed the maximum normal stress failure criteria, even the differential between the compressive and tensile stress at the transition point is only approximately 580 psi (4 MPa), yet catastrophic failure of the seal was observed for this test. The discontinuity provides additional insight into the behavior of the rear face at this location. This sensor was positioned almost directly opposite the impact location. The strain plateau, from approximately 1.25 to 1.75 ms as indicated in Figure 4.8, suggests a potential plastic deformation or displacement at the sensed location since the sensor measured a constant strain level and temporary relief, from approximately 1.75 to 5.25 ms, which would be indicated if a subsurface crack or spall were developed. Following the material failure, additional relief waves would be developed from the new interface. This appears to be in agreement with the observed parallel cracking (flexural failure) that was observed on the top face of the seal and the fragmentation during this test as seen in Figure 4.9 and Figure 4.10.

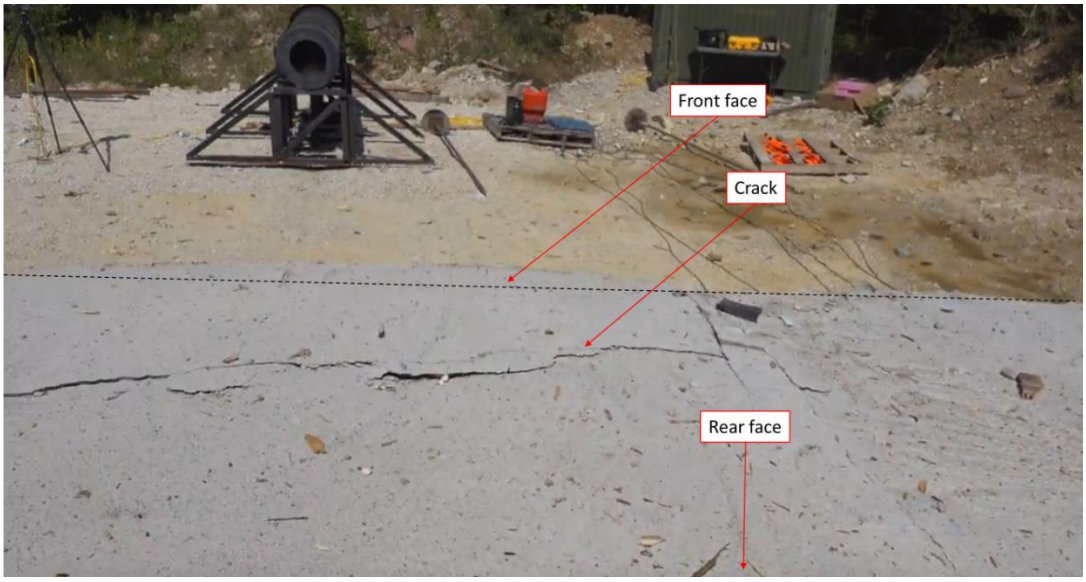


Figure 4.9. Flexural cracking in top face

Figure 4.10. 50 psi seal fragmentation planes

The signal behavior observed during this test was unique from the other test cases, and cracking observed during other tests was radial or perpendicular to the sensing plane which would not be as readily detected by the sensors. This test case further illustrates the incomplete information an observer would have were they to assess the rear face of the seal for damage after an impact event.

In addition to strain, the stress for each test case was derived and plotted. The peak stress on the rear face is plotted against the modified Weber number, ξ, in Figure 4.11.

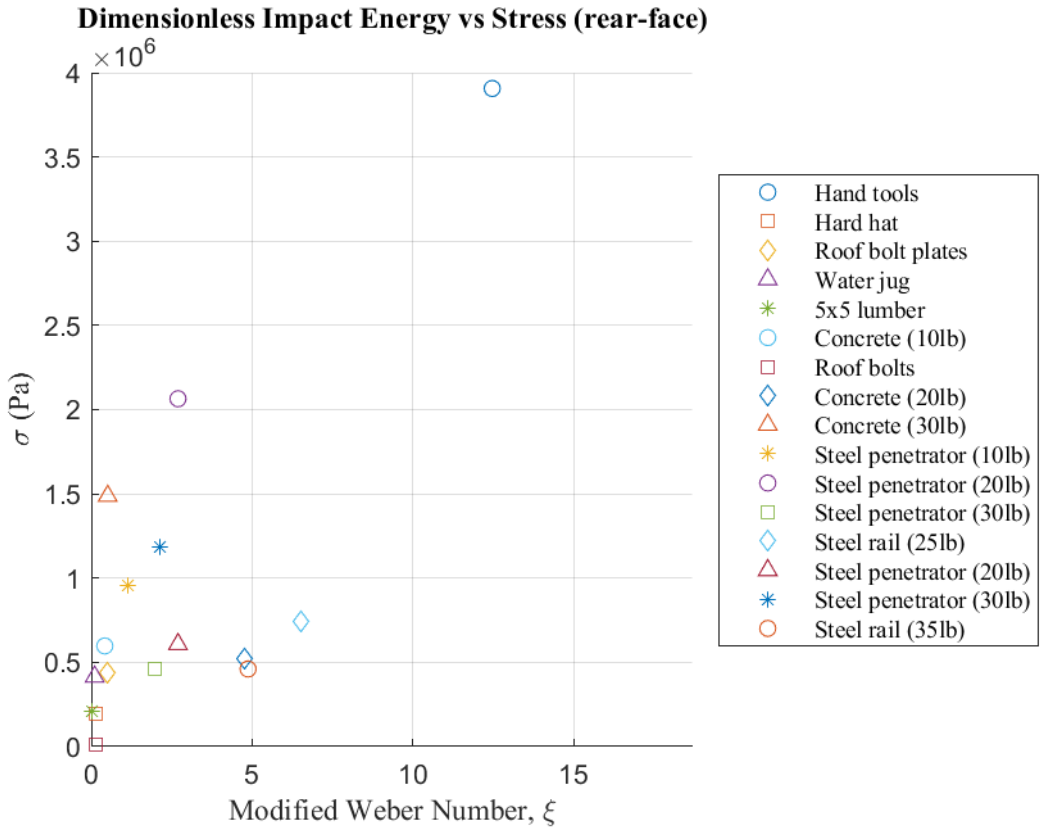


Figure 4.11. Dimensionless impact energy and peak rear face stress

Failure of the seal, in the form of cracking occurred during the Steel penetrator (10 lb), Steel penetrator (30 lb)[12], Steel rail (35 lb), Steel penetrator (20 lb), and Steel penetrator (30 lb)[13] tests.

The test cases with the highest derived rear face stresses occurred during tests that did not exhibit rear face cracking. However, for test cases with a peak rear face stress above 145 psi (1 MPa), the seal did experience failure to the impact face (cratering or

[12] Testing on 6 September 2019

[13] Testing on 19 September 2019

cracking) with the exception of the 30 lb concrete test case. In addition, for the test cases where failure occurred, energy was absorbed during local failure of the impact face and would not be observed in the rear face. This could account for the decreased stress levels in such cases as the 20 and 30 lb steel penetrators tested on 19 Sep 19.

Catastrophic failure of the seal occurred during the steel penetrator (30 lb) test case, 19 Sep 19. This case is particularly interesting because the angles at which the fragmentation plane lines appeared are symmetrical about the impact location, as seen in Figure 4.12.

Figure 4.12. 50 psi seal failure plane symmetry

One challenge with assessing the damage and failure of structures stems from the fact that a previously damaged structure will be in a weaker state and have reduced load

bearing capacity as a result of the damage. Thus, the final test, which resulted in catastrophic failure, was testing a weakened structure and may not have resulted in the same response if testing an original structure. The effects of cumulative damage effects were not considered in this research, but may contribute to non-representative failures. There are models which incorporate elastic stiffness reduction as a result of damage (Grassl, Xenos, Nyström, Rempling, & Gylltoft, CDPM2: A damage-plasticity approach to modelling the failure of concrete, 2013), but no strength reduction models were incorporated in the analysis for this research.

It is known that on a free surface, a stress wave will change sense (i.e. a compressive wave will become tensile), and that the pressure wave interactions near the free surface are the net reactions of the stress waves acting at that location. It appears that this failure is due to the free edge effects which would not have been present in an underground mine seal since for a fixed boundary, the stress wave maintains its sense at that interaction. In an underground seal, the side and top edges would be fixed in the surrounding medium. However, an even more interesting case is that these fixed surfaces do not change the free surface conditions at the rear face of the seal. Thus, the energy transferred to cause catastrophic failure on the side and impact face would likely be transferred instead to other areas of the seal, possibly the rear face and cause catastrophic failure elsewhere in the seal. Large parallel cracks were observed on the top face of the seal after this shot indicating a tensile flexural failure. In short, the overall failure the seal experienced as a result of edge effects during this test case would not necessarily have been avoided with constrained edges but could have contributed to the magnitude of the failure during this test.

4.6. PENETRATION

Penetration was determined by taking the volume loss data for each test case and deriving an estimate for the measured value. Concrete fails in penetration cases by the formation of an initial crater region and then tunnel region, provided the projectile velocity is sufficient to exceed the crater region (Carlucci & Jacobson, 2008). The crater region is in the general form of a cone. Thus, this shape was used to model the volume of the crater region. The diameter of the cone base was determined and the penetration depth derived by using the volume of a cone.

$$V = \frac{\pi r^2 h}{3} \tag{50}$$

where V is the volume of the cone, r is the radius of the base, and h is the height of the cone. The height (h) was correlated to the penetration depth (z), thus:

$$z = \frac{3V}{\pi r^2} \tag{51}$$

where z is the penetration depth, V is the volume loss data, and r is the radius of the crater at the surface. This equation was the penetration depth equation used to derive a 'measured' penetration value from the measured volume loss data for each test case. The Forrestal and NDRC penetration models were used for estimating the penetration depth. The results of each model were compared to determine if the difference between the two estimates resulted in a spread of greater than a defined limit from the mean, if so, the minimum estimate was taken, if not, the mean of the two estimates was taken as the estimate. In addition to measuring volume loss, the optical measurements estimated impact crater depth. The estimated and derived penetrations as well as the optically measured values for maximum penetration are given in Table 4.6.

Table 4.6. Penetration values for test cases

Test case	Estimated penetration (in)	Derived penetration (in)	Measured penetration (in)
Hard hat	0.2	0	0
Hand tools	1.5	0.5	0.4*
Roof bolt plates	0.7	0.1	0.4*
Water jug	0.3	0	0
5 in x 5 in nom. Lumber	1.6	0	0
Roof bolts	5.2	0.4	0.1
Concrete (10 lb)	0.5	0	0
Concrete (20 lb)	2.9	0	0
Concrete (30 lb)	1.3	0	0
Steel penetrator (10 lb)	2.3	1.3	1.2
Steel penetrator (20 lb)	5.1	1.6	0.9
Steel penetrator (30 lb)	5.0	2.4	0.7
Steel rail (25 lb)	4.3	1.0	0.3*
Steel rail (35 lb)	4.4	0.2	0.9
Steel penetrator (20 lb)	5.1	1.9	3.1
Steel penetrator (30 lb)	5.5	26.0	4.8

* - Indicates an average value

The optical measurement figures can be seen below in Figure 4.13 through Figure 4.16. The maximum depth color for each location was matched to the scale to determine the depth of penetration for each test case.

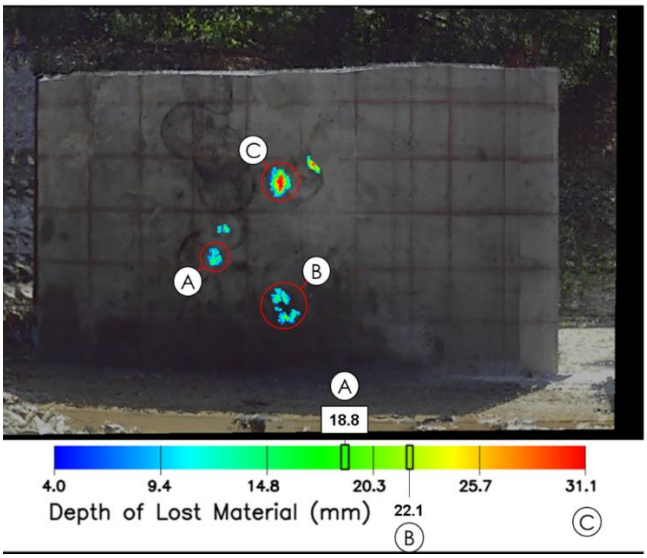


Figure 4.13. Crater dimensions for A. Steel penetrator, 30 lb, test date: 6 Sep 19, B. Steel penetrator, 20 lb, test date: 6 Sep 19, C. Steel penetrator, 10 lb

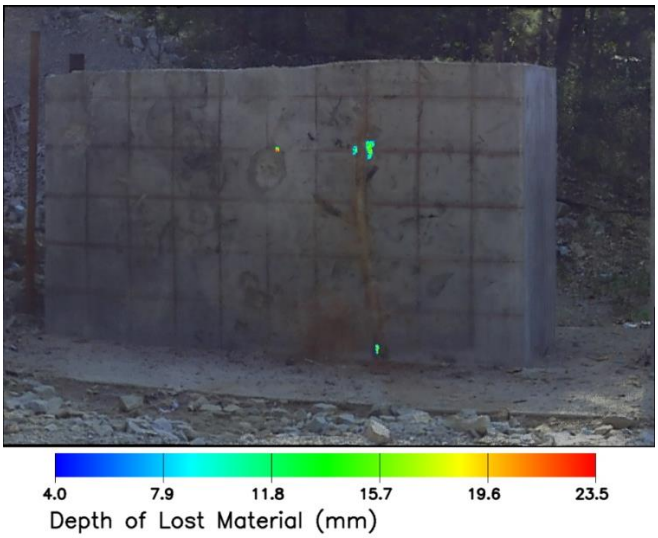


Figure 4.14. Steel rail crater dimensions, 35 lb, test date: 19 Sep 19

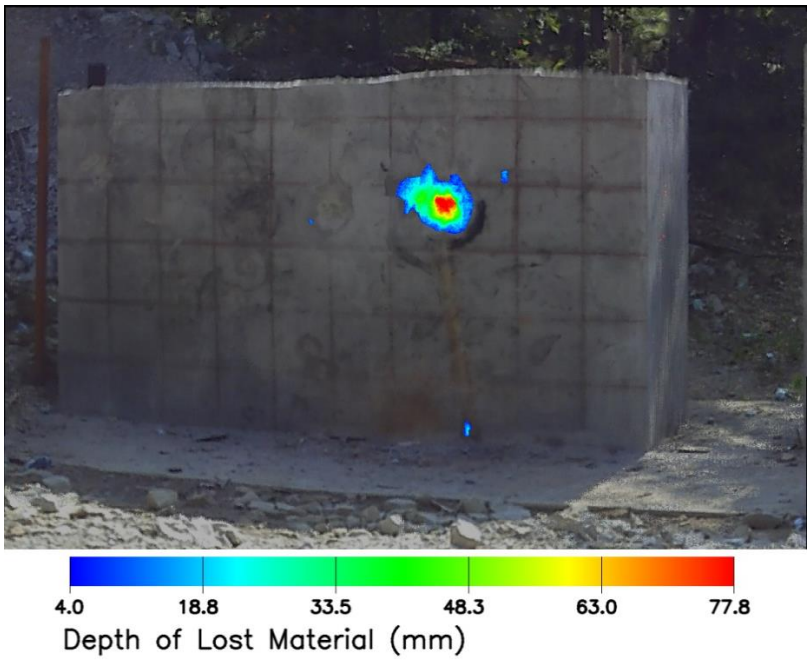


Figure 4.15. Steel penetrator crater dimensions, 20 lb, test date: 19 Sep 19

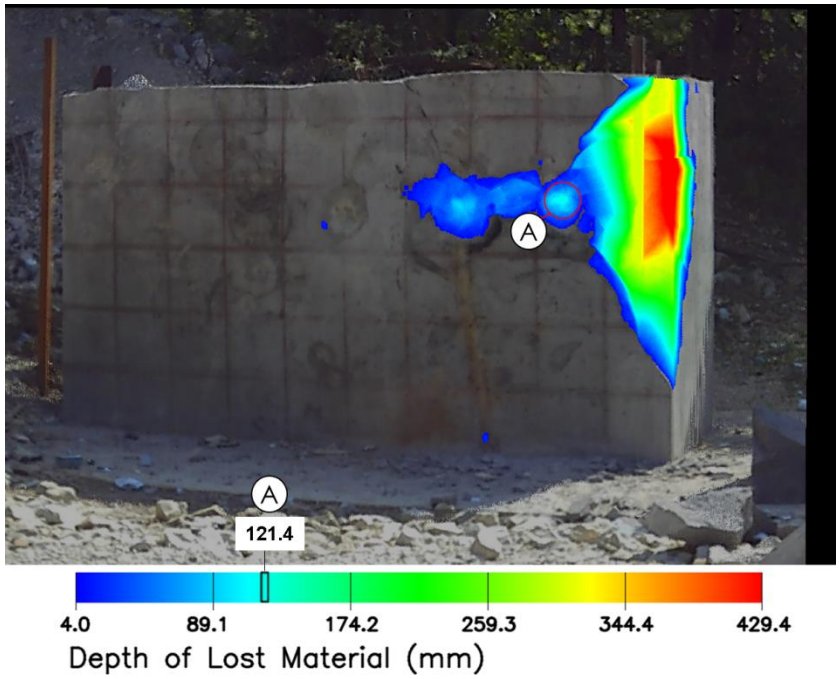


Figure 4.16. Steel penetrator crater dimensions, 30 lb, test date: 19 Sep 19

The measured penetration depths were plotted for each test case with the corresponding modified Weber number, ξ. The relationship can be seen in Figure 4.17.

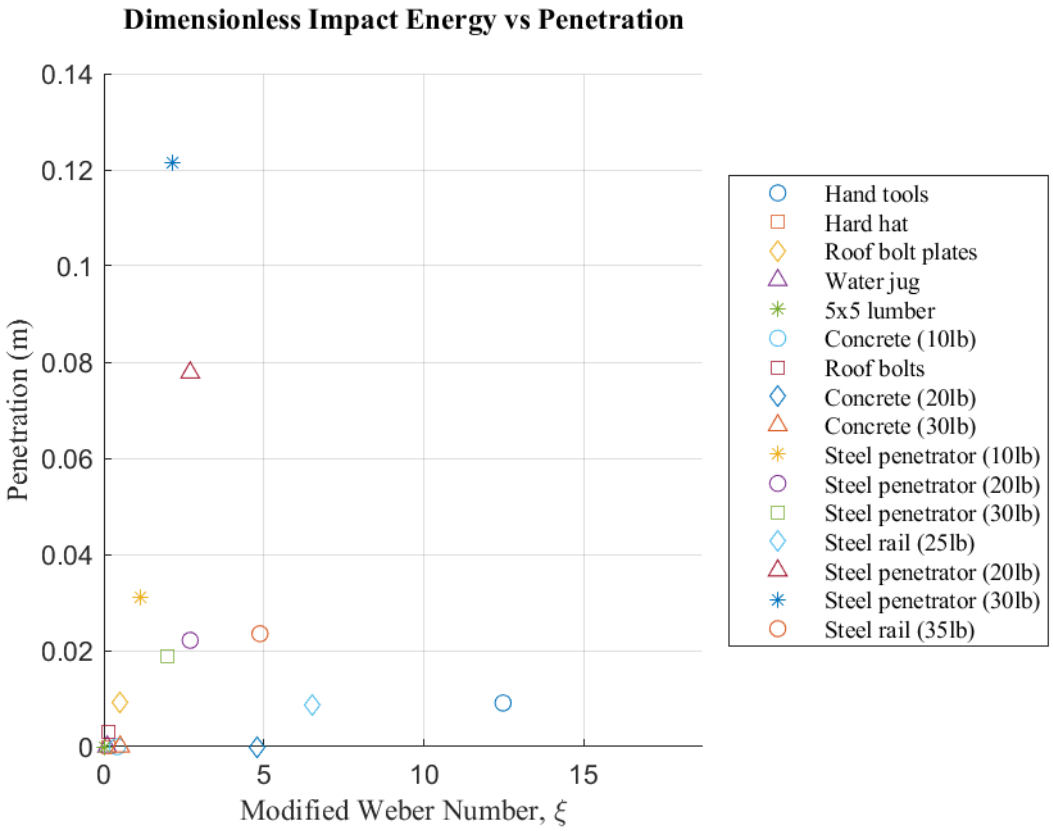


Figure 4.17. Dimensionless impact energy and penetration

The 20 and 30 lb steel penetrator tests conducted on 19 Sep 19 do not appear to fit well with the measured penetration data trend for the other test cases and could be attributed to lacking velocity data.

Additionally, the penetration model estimates and the derived penetration are shown in Figure 4.18.

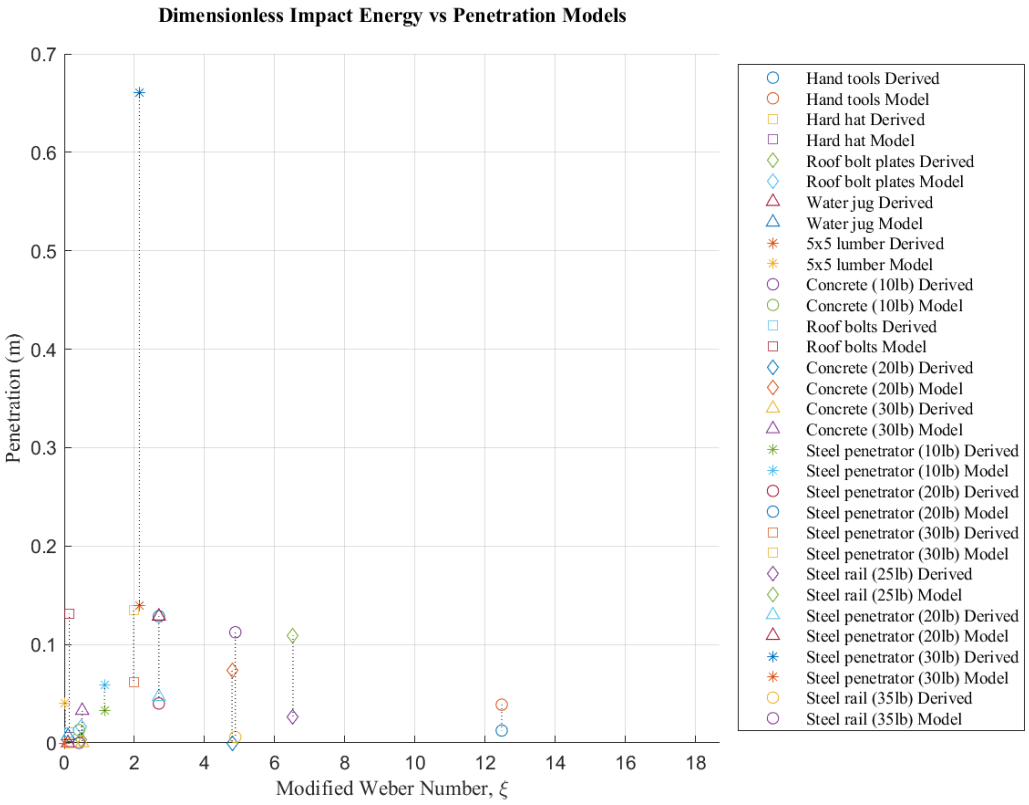


Figure 4.18. Dimensionless impact energy and penetration models

As can be observed from the data in Table 4.6 and comparison between Figure 4.17 and Figure 4.18, the tendency for the penetration models is to overestimate the depth of penetration. There appears to be more scatter in the measured penetration values, as shown in Figure 4.17, than when compared to the data in Figure 4.11 for the Weber number and stress relation. This may suggest a better correlation of Weber number to stress than for penetration. However, the projectile velocity measurements also have a degree of uncertainty due to the effects of smoke obscuration for the sensors during a number of tests.

An apparent grouping of projectile penetration and structural response can be observed when the rigid body projectile assumption is investigated more closely. There is an observable distinction between the penetration and rear face stress or strain state and deformable or non-deformable projectiles. The deformable, non-rigid body projectiles[14], show a general grouping by ξ and penetration within family. Projectiles which showed little or no deformation are classified as rigid body, primarily the steel objects. Rigid body projectiles exhibited more cohesive penetration behavior in the family of projectiles. The penetration estimates for deforming projectiles are consistently out of agreement with the measured results. This lack of agreement is to be expected since the penetration models used assume a rigid body projectile.

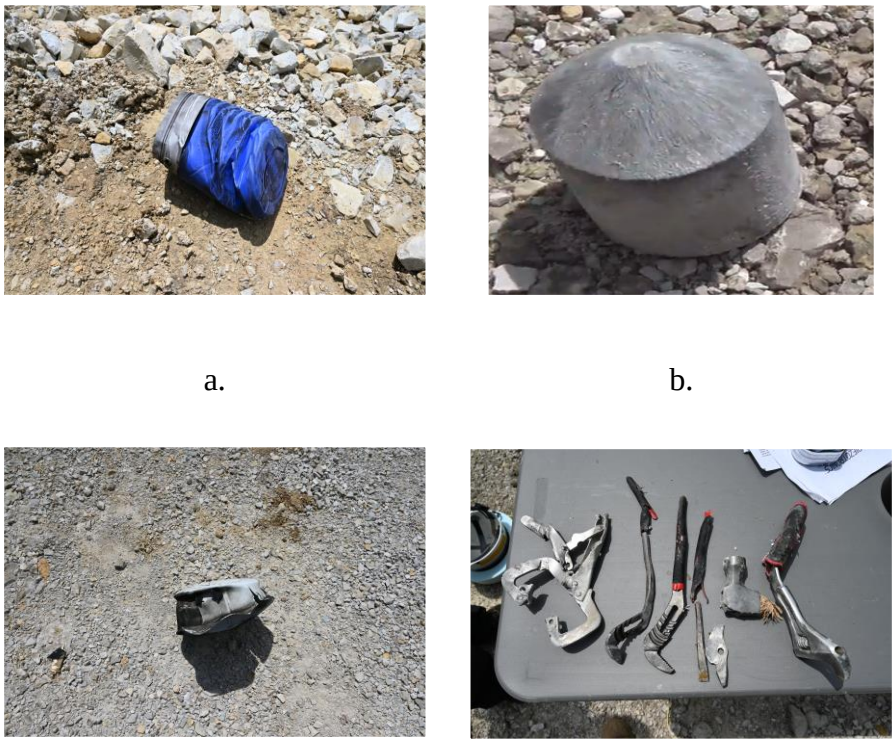
After testing, the projectiles were recovered and visually inspected. There were visible distinctions between deformable and non-deformable projectiles, as seen in Figure 4.19. Additional photographs of the projectiles used in testing, including before and after photographs, can be found in APPENDIX A. Projectiles with high relative shock impedances had a decreased tendency to exhibit deformation. All steel penetrators were recovered with no observed deformation, but the recovered hand tools exhibited observable deformation.

Conversely, there were also test cases with rigid body projectiles, such as the roof bolts and rails, which did not fit well with the penetration models. This is likely due to shape factor, as these projectiles had relatively small diameters to their length (which correlates to projectile mass). Thus, the projectiles, which impact nose-on in the

---

[14] For example, hard hat, water jug, lumber, concrete.

penetration models, would exert high pressure on a localized area resulting in target failure. However, during testing, these long slender projectiles had a tendency to impact side-on, which would distribute the impact load to a significantly greater effective diameter, thus decreasing the impact load.



a. b.

c. d.

Figure 4.19. Deformable and non-deformable projectiles recovered after testing, a. Water jug, visible deformation, b. Steel penetrator (10 lb), no observable deformation, c. Har hat, visible deformation, d. Hand tools, minimal deformation

Not all projectiles were able to be recovered after testing for inspection, in some cases, the projectiles were unrecoverable. An interesting projectile response is observed

during testing with the concrete projectiles, which exhibited perfectly deformable behavior. This can be seen in the sequence of images shown in Figure 4.20.

a.

b.

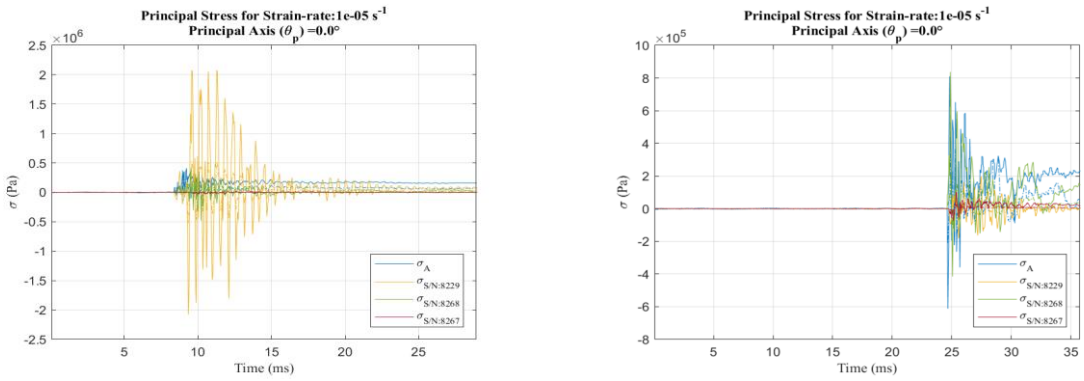
c.

d.

Figure 4.20. Concrete (30 lb) totally deformable projectile impact

The sequence portrayed in Figure 4.20 sheds light on the structural response observed in the target, particularly when comparing the magnitudes of the stresses

developed. It would appear the majority of the impact energy for highly deformable projectiles is absorbed by the projectile instead of being transferred to the target. For comparison, take the 20 lb steel penetrator case and the 20 lb concrete projectile test. The respective dimensionless impact energies are 2.7 and 4.8. Yet, the induced stress from the steel penetrator (299.4 psi) test is nearly four times that of the resultant stress from the 20 lb concrete test (75.6 psi). The stress plots for these two tests are shown in Figure 4.21.



a. b.

Figure 4.21. Relative stresses for a rigid body and non-rigid body projectile, a. Steel penetrator (20 lb), rigid body b. Concrete (20 lb), non-rigid body

This supports preliminary modelling, namely that rigid body projectiles will induce greater stress in the target and ultimately present a greater risk for seal damage than non-rigid body projectiles. The data suggests, that for comparable impact velocities, or dimensionless impact energy, the resultant load from a rigid body projectile can be

multiple times that of a non-rigid body projectile. In addition, data suggests that failure in the seal will have a reducing effect on the stress observed in the rear face (i.e. compare 20 lb steel penetrator cases). The relative stress for each test case, using the derived stress from the concrete (10 lb) test case as the reference value, can be seen in Figure 4.22.

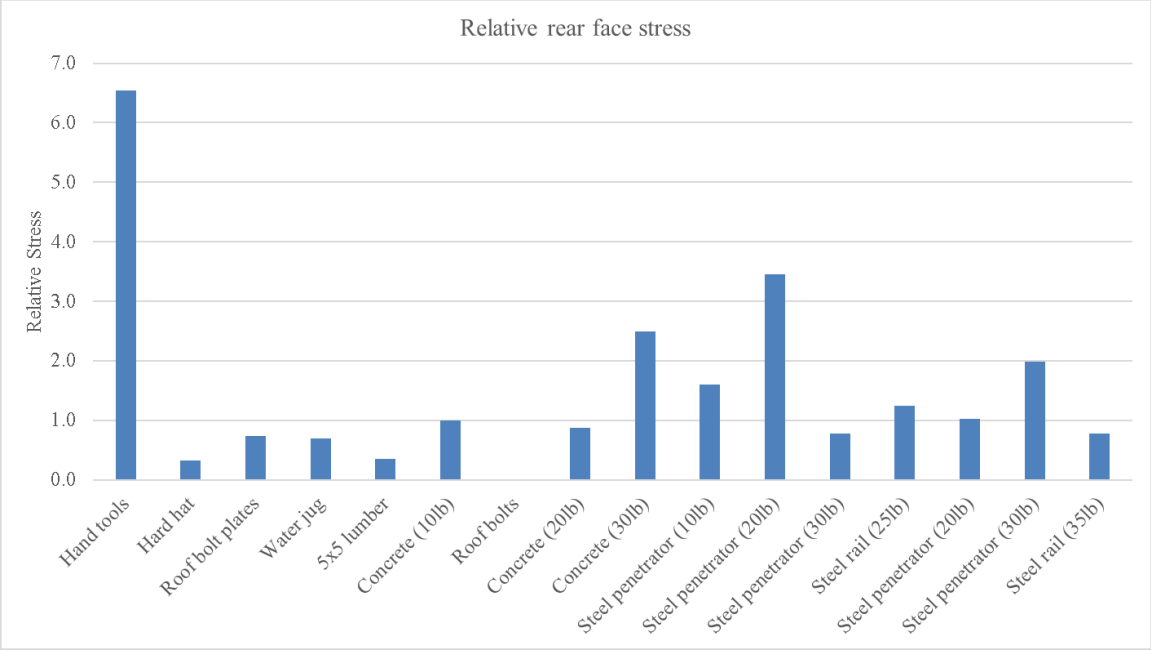


Figure 4.22. Relative rear face stress by test case

4.7. COMPARISON WITH ANALYTICAL MODELS

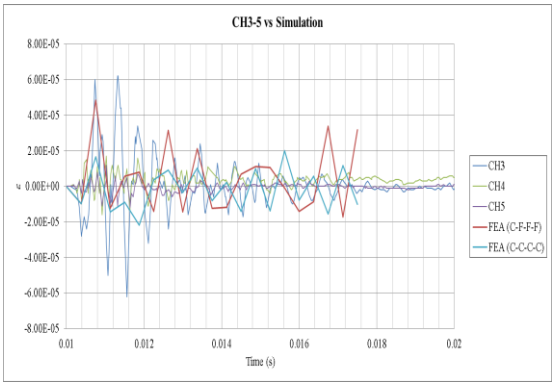
In comparing the FEA model with the empirical data, the initial assumptions must first be reviewed. Simplified forms were developed for each of the projectiles, to reduce solution times it was necessary to reduce the number of elements. This simplification increased the coarseness of the prediction, resulting in a loss of fidelity for fine prediction

but it was hypothesized that the magnitudes of the stresses and strains would be representative and not necessarily exact.

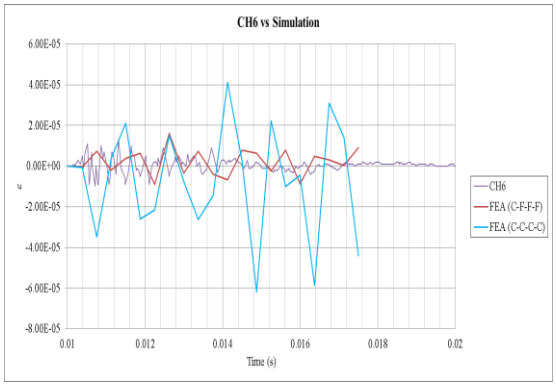
A finite element simulation was developed for each impact case to estimate the stress state in the seal. For each test case, the impact was modeled using simplified geometries of the projectile with projectile mass as the driving parameter of the simulated projectile. Each test simulated a normal, planar impact. The seal was modeled as both a fixed-free-free-free plate and a fixed-fixed-fixed-fixed plate in order to compare empirical data with the test configuration (fixed-free-free-free) and then to use the model to extrapolate the response to an operational configuration (fixed-fixed-fixed-fixed). Strain-time plots from the simulation were overlaid the test data for comparison as can be seen in Figure 4.23.

As can be observed from the comparison, the simulation exhibits a tendency to over predict the strain in the seal. Closer inspection suggests this is may be due to modeling the impact as normal and planar, and material model simplifications in the simulation. These modelling simplifications, normal and planar impact, produce the greatest transfer of kinetic energy from the projectile to the seal, and by correlation, the greatest penetration or structural response. A yawed projectile will exhibit a tendency to reflect, or ricochet, from the surface of the target upon impact. As the angle of impact from the surface unit normal increases, the penetration depth decreases. In the tests performed, the angle of impact is not known, but the probability of a true planar and normal impact is inherently negligible due to the test configuration: aiming limitations with the cannon (non-parallel muzzle-target plane), the free space between muzzle and target, and projectile geometries. By extension, it can be assumed that the projectile

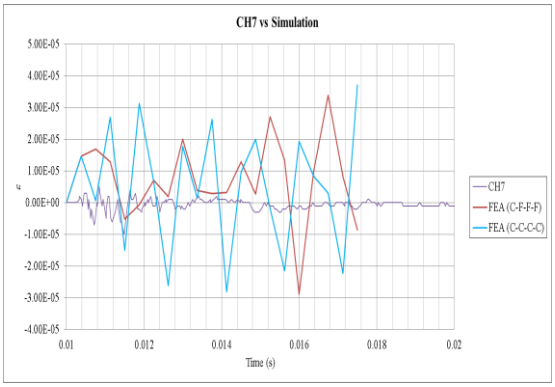
impacts are non-planar/non-normal and hence, the magnitude of the dynamic response of the seal is reduced as a result of decreased energy transfer. It is worthy of note, that a normal, planar impact is not required for this test. In fact, the variability of each test case is more representative of operational conditions. Observing the response of the varied types of projectiles that can be found in an operational underground coal mine is the intent of the test.



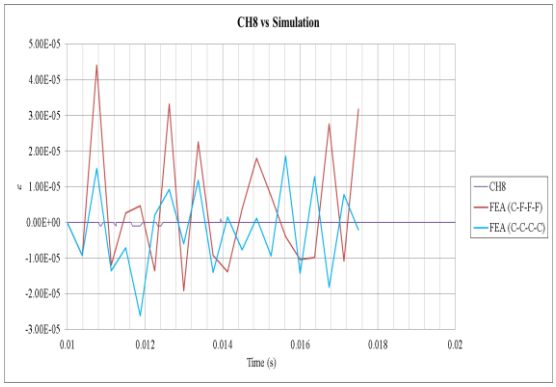
a.



b.



c.



d.

Figure 4.23. FEA strain predictions overlaid with sensor data, steel penetrator (20 lb) test case, test date: 6 Sep 19

In addition, the material model used in the FEA model is also a likely point of departure from the empirical data. This can be seen in the minimal decay of the stress wave from one sensor location to another and at each location for the load duration.

For the test cases analyzed, the FEA analysis was a good predictor of the initial dynamic response in the seal, given specific impact location and projectile data inputs. As time and space from the impact increase, the correlation between the model and empirical data decreases.

A review of the data suggests that the observed damage is representative of impact location and cumulative damage effects may be minimal. The predicted strains were of comparable magnitudes between an undamaged FEA and the damaged test subject. The channel 7 and channel 8 strain data for the 20 and 30 lb steel penetrator tests (which were the final two test cases) exhibited marked deviation from simulation results, however this was common across test cases. For these final two test cases the strain data for channels 3 through 6 showed good correlation with the models for initial response values. In consequence, it appears that for these tests, cumulative damage effects can in large part be neglected and deriving the simulation results from impact location will provide a good degree of fitness with empirical results. This assumption may not hold were any tests to be conducted after the 30 lb steel penetrator due to the major volume loss.

The peak stress at the rear face was also estimated in the simulations for both the test and operational configuration, and these were compared to empirical data. The FEA prediction for a selected test case, the 20 lb steel penetrator, is shown in Figure 4.24.

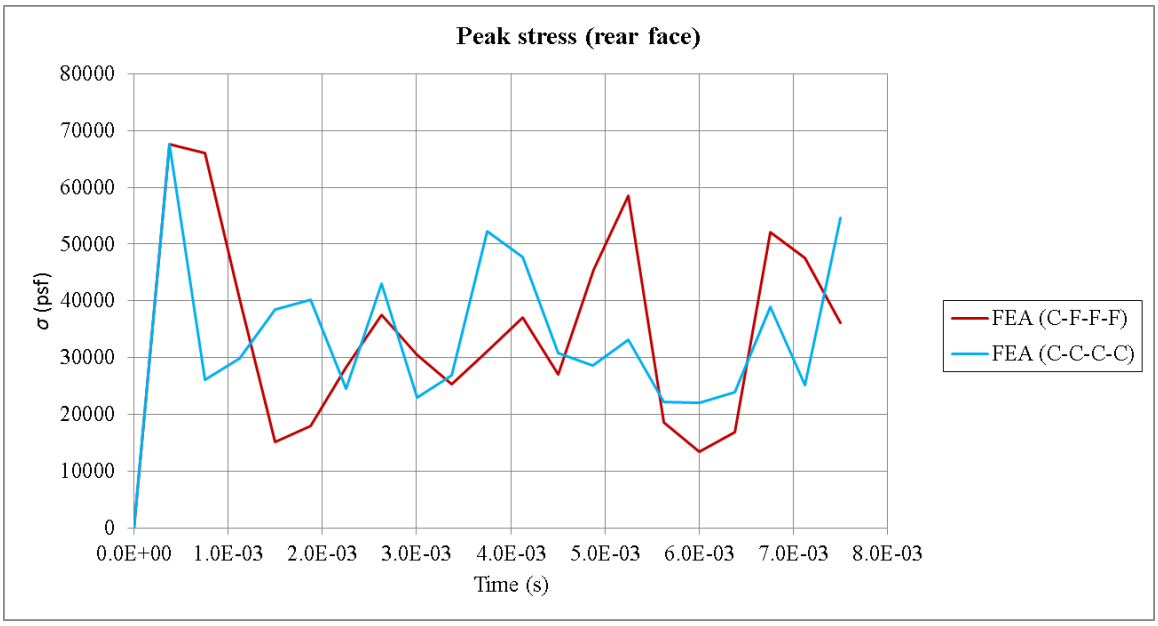


Figure 4.24. Estimated peak rear face stress, steel penetrator (20 lb), test date: 6 Sep 19

For this test case, the derived peak stress at the rear face is 290 psi or 41,771 psf (2 MPa) which is approximately 38% less than the 67,000 psf (3.2 MPa) predicted in the FEA model. Reviewing the simulation data, it is noted that as the stress waves propagate through the seal and interfere with other stress waves and reflect from boundaries, peak stress locations are generated at locations which may not occur at the strain sensor location. Derived peak stresses ranged from 20% to 90% of estimates from simulated values over the entire rear face. These results indicate that the strain sensors may not be located at the peak stress locations.

A critical velocity limit for a projectile type at which the projectile would cause failure in the seal was derived by simulation. For non-rigid body projectiles (i.e. HDPE, concrete) there was no velocity limit determined as the projectile failed prior to to the seal. In rigid body projectiles, failure in the seal was estimated at approximately 550 -

600 ft/s (168 - 183 m/s) for 20 - 30 lb (9 - 14 kg) projectiles. These limits were taken near constraints and are thus expected to be valid for a more mechanically advantageous, central impact. The predicted failure mode was rear face cracking for a fixed-fixed-fixed-fixed plate, based on maximum normal stress criteria. In the simulation, when the projectile began to exceed 1,000 ft/s (305 m/s) and into the 1,500 ft/s (457 m/s) range, the rigid body projectile was predicted to fail (i.e. similar to a Taylor test). This projectile behavior is not expected for a concrete target and steel projectile based on observations from testing and research, and could be attributed to edge effects (fixed constraint) in simulation. The expected behavior is projectile penetration, spallation, cracking, and possibly fragmentation at sufficient impact velocities, as observed during testing.

5. CONCLUSION

The task of assessing impact events on mine seals is a multifaceted problem. From the beginning step of the problem space, estimating projectile generation resulting from blast waves, to acceleration in the flow field, then to impact, response, and failure prediction. Indeed, where current research regarding methane gas DDT in underground coal mines is still producing new insights on the real conditions during accidental explosions, there is much that can be learned in a study of this area. The value of further study is evident when one considers the direct correlation between internal blast wave velocities and resultant projectile velocities.

For the test cases in this experiment, the impact velocities observed generated measurable strain on the rear face of the seal. Surface cracking on the rear, side, and top faces were observed during some test cases; in addition, significant surface cratering and cracking was observed on the impact face. Projectile mass had a greater impact on damage, and volume loss, than initial estimates. Projectile velocities attained were within the expected range for projectiles generated from an internal methane explosion, however, a more robust method of velocity measurement is recommended to increase data capture success. Projectiles used in the test cases ranged from 1 to 34.5 lb (0.5 to 15.6 kg) with velocities ranging from 122 to 642 ft/s (37 to 196 m/s). Catastrophic failure of the seal, with major volume loss and through-thickness cracking, was observed during testing with heavy projectiles.

The test data exhibited inconsistent correlation with the FEA models developed. For each test case, the empirical data for that test case and strain estimates from the FEA

model replicating the impact for that specific test case were compared. The FEA models tended to fit empirical data in the initial response peaks better and the strain amplitudes generally did not decay as quickly as the test data. Observed damage was deemed representative of impact location and cumulative damage was determined to contribute a negligible effect to the test cases. The dynamic response between empirical data and simulation results exhibited good initial correlation, even for later tests where cumulative damage would have the most effect. The fixed-fixed-fixed-fixed seal FEA model did not appreciably differ from fixed-free-free-free FEA results for the range of projectile velocities and masses tested. This suggests that the observed damage during testing could be representative of damage in an operational mine seal. The primary deviation between fixed-free-free-free and fixed-fixed-fixed-fixed plates dynamic response involves edge effects for impacts occurring near a free edge. In this case, the test configuration is expected to suffer greater damage than the operational configuration.

To address the original question, whether the lack of projectile damage criteria is a concern for current seal designs, the data suggests there is potential risk for significant and catastrophic damage to seals within the expected limits of projectile mass and velocity resulting from an accidental explosion. The data and numerical simulation suggest rigid body projectiles with masses in the range of 20 to 30 lb (9 to 14 kg) and velocities in the range of 550 to 600 ft/s (168 to 183 m/s) could result in significant penetration and structural failure. In addition, visual inspection of the non-impact face of the seal may not be sufficient to identify potentially critical damage in the structure, as damage, including cracking may not be visually discernable. As a result, impact damage

could go undetected. An in-situ non-destructive testing method, such as ultrasonic testing or Rayleigh-wave diffraction, may adequately identify the extent of damage.

In summary, this research identified that monitored 50 psi mine seals may suffer structural failure resulting from impact, particularly when the projectiles are massive and rigid-body. The characteristics of an internal methane explosion are anticipated to propel common mining equipment and material to velocities sufficient to induce failure in a mine seal. Since the atmosphere behind most sealed areas can be expected to reach explosive limits during some point of their operational life, the risk of mine seal failure due to impact loading is non-negligible. These conditions provide a strong basis for further investigation into the development of mine seal damage criteria.

Finally, the findings associated with this research suggest there is still much to be learned regarding mine seal integrity when subjected to impact loads. Recommendations for further research include: higher velocity impact testing, especially in the 600 ft/s to 1,000 ft/s (183 m/s to 305 m/s) range, steel reinforced mine seal testing, and blast and projectile load coupling, since there are synergistic effects possibly reducing required projectile velocity to induce failure.

APPENDIX A.

TEST DATA

The collected and output data from testing is provided in this appendix for all tests performed. Data is provided in accordance with test sequence.

For figures in this appendix referencing strain sensor serial numbers, the positions and associated serial numbers are given in Figure A.1 and Table A.1.

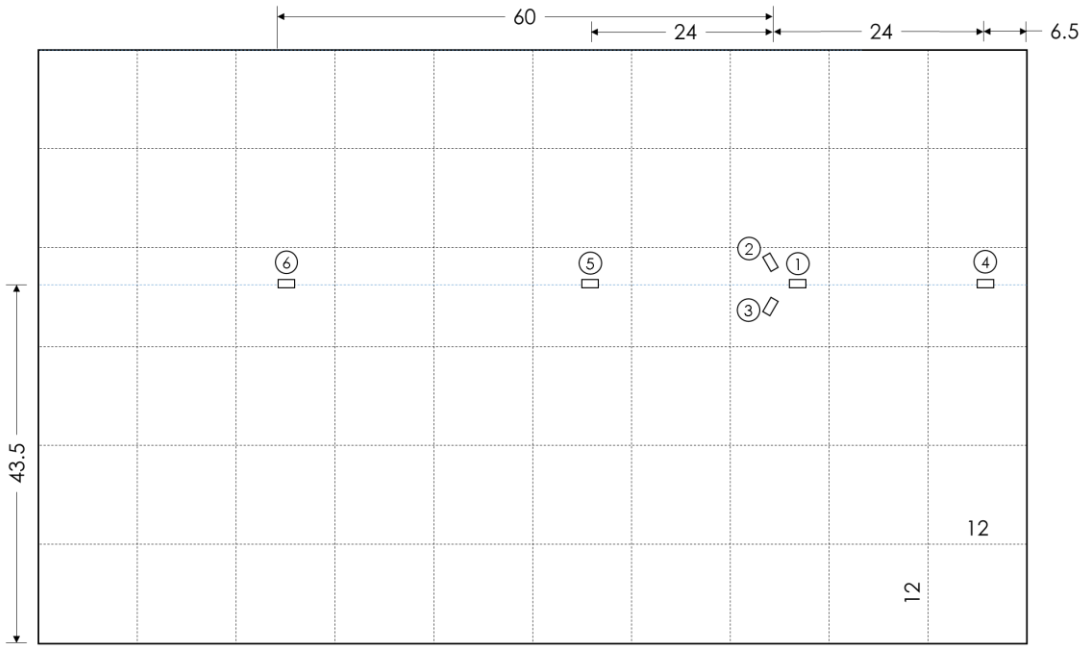


Figure A.1. Strain sensor installation

Table A.1. Sensor ID, data channel, and serial number matrix

Sensor ID (ref. Figure A.1)	Data channel	Serial number (for testing on 31 Jul 19)	Serial number (for all other tests)
1	CH3	8268	8229
2	CH4	8269	8268
3	CH5	8267	8267
4	CH6	8229	8269
5	CH7	8229	8228
6	CH8	8266	8266

Test dates and the sequence of testing is given in Table A.2, with the impact locations projected onto the rear face in Figure A.2.

Table A.2. Test dates, sequence, and total projectile weights

ID	Testing date	Test case	Total projectile weight (lb)
A	31 July 2019	Hard hat	1
B		Water jug	1
C		Hand tools	9
D		Roof bolt plates	17.5
E	13 August 2019	5 x 5 lumber	30
F		Roof bolts	17
G		Concrete (10 lb)	10
H	27 August 2019	Concrete (20 lb)	20
I		Concrete (30 lb)	30
J		Steel penetrator (10 lb)	10
K	6 September 2019	Steel rail (25 lb)	25
L		Steel penetrator (20 lb)	22
M		Steel penetrator (30 lb)	32
N	19 September 2019	Steel rail (35 lb)	34.5
O		Steel penetrator (20 lb)	22
P		Steel penetrator (30 lb)	32

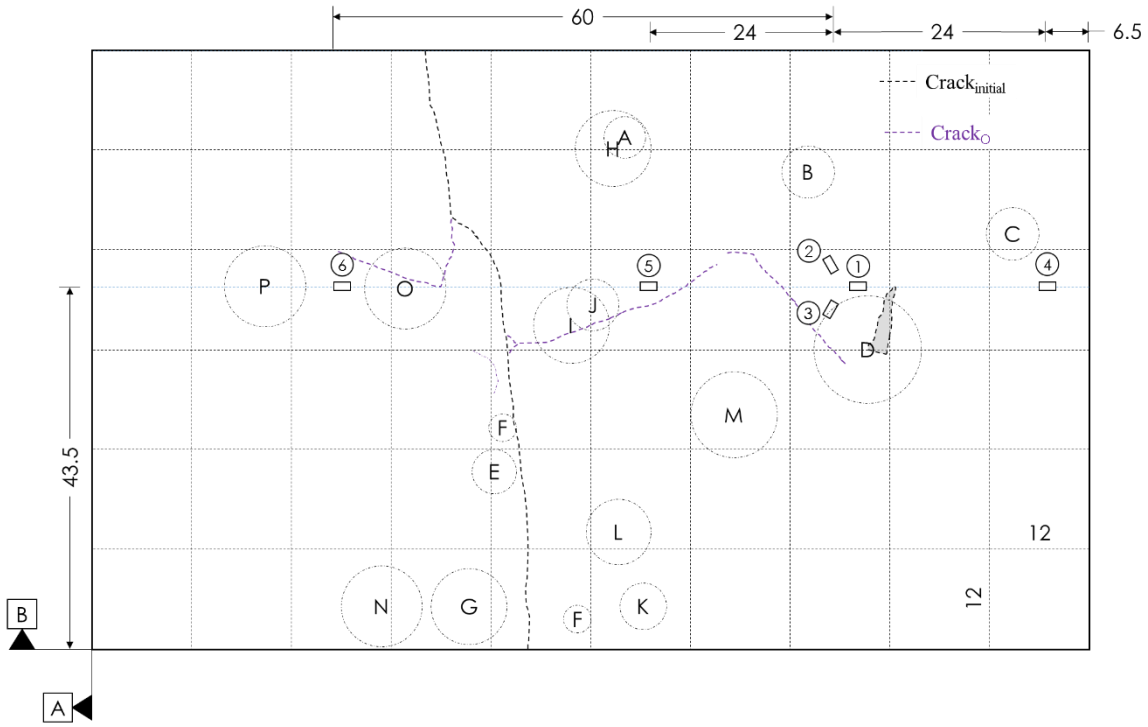
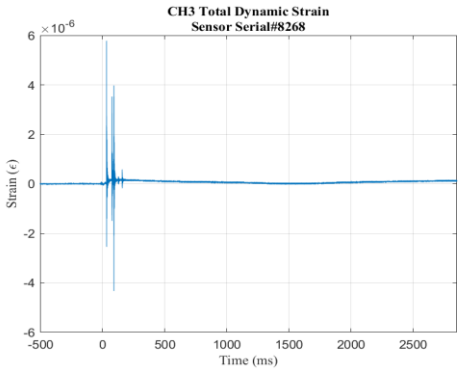


Figure A.2. Projected impact locations, rear face

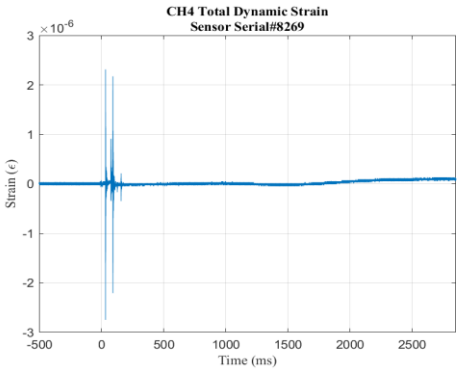
The following figures are given in order of test case.

A, Hard hat, 31 July 2019

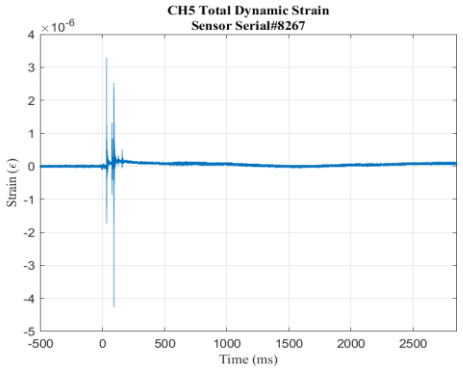
Figure A.3. Hard hat



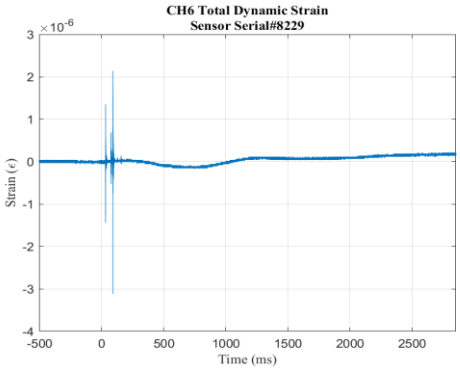
a.



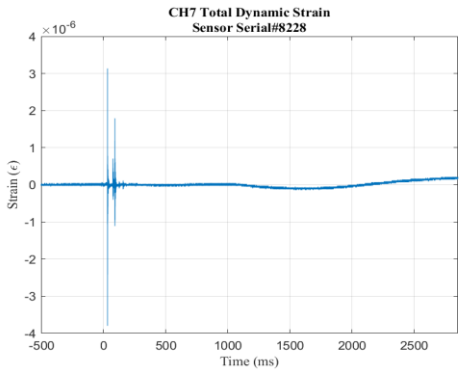
b.



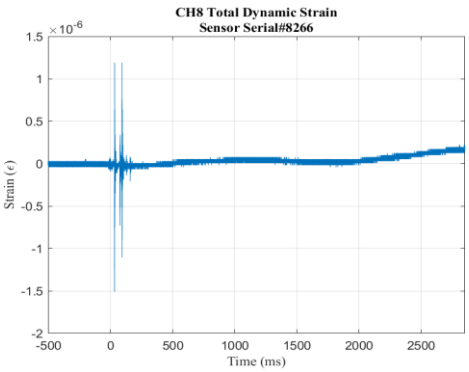
c.



d.



e.



f.

Figure A.4. Strain response, hard hat, test date: 31 Jul 19

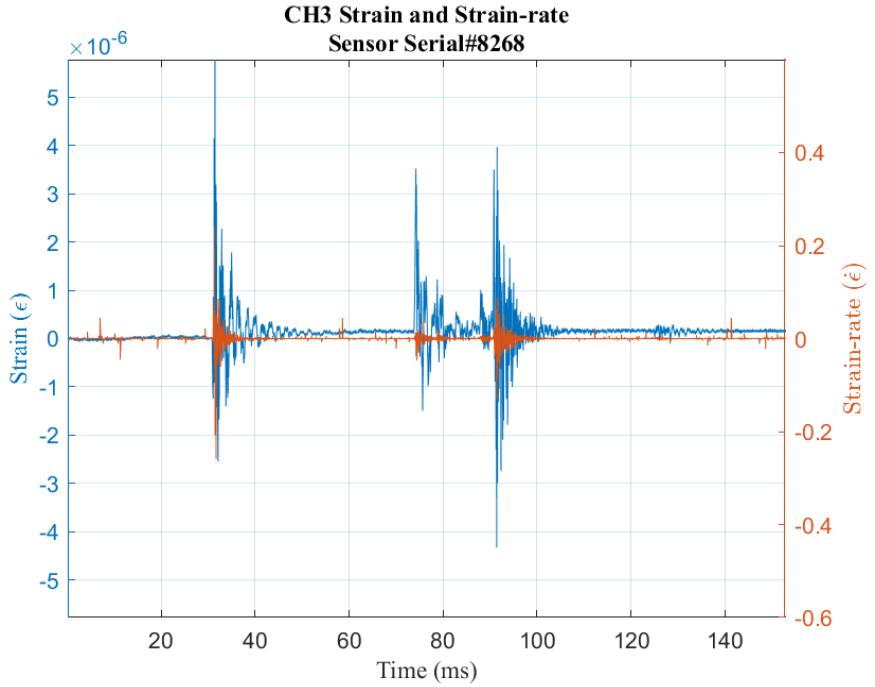


Figure A.5. Strain and strain rate, hard hat, CH3

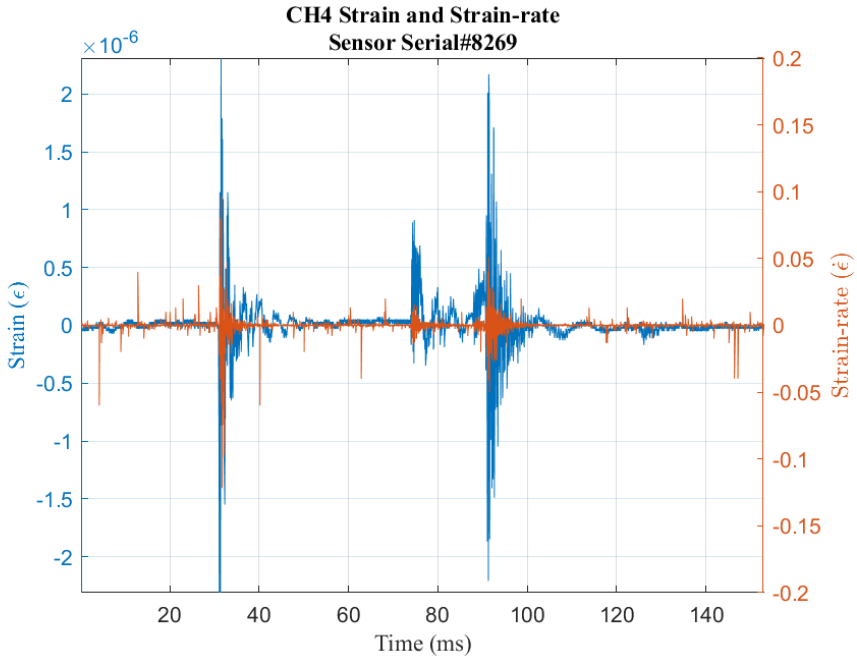


Figure A.6. Strain and strain rate, hard hat, CH4

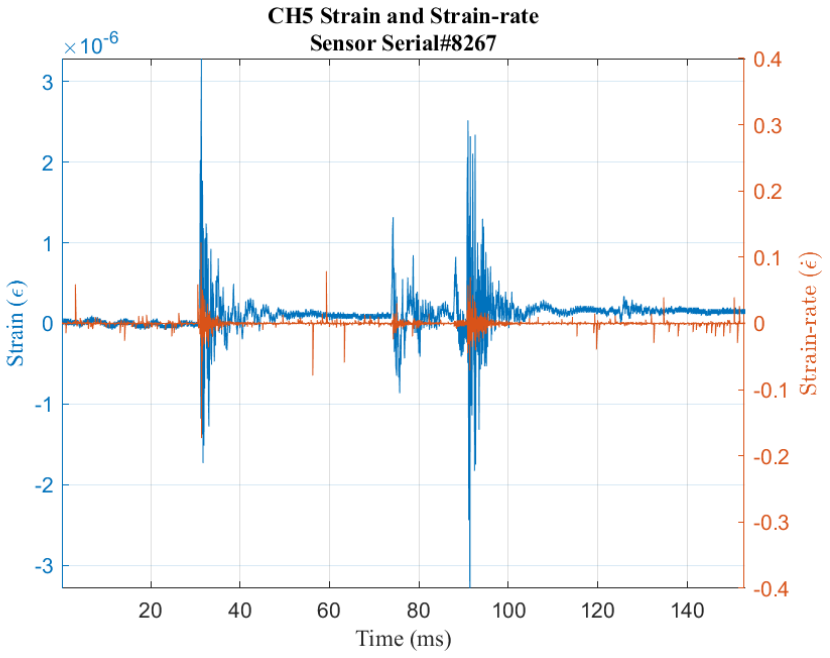


Figure A.7. Strain and strain rate, hard hat, CH5

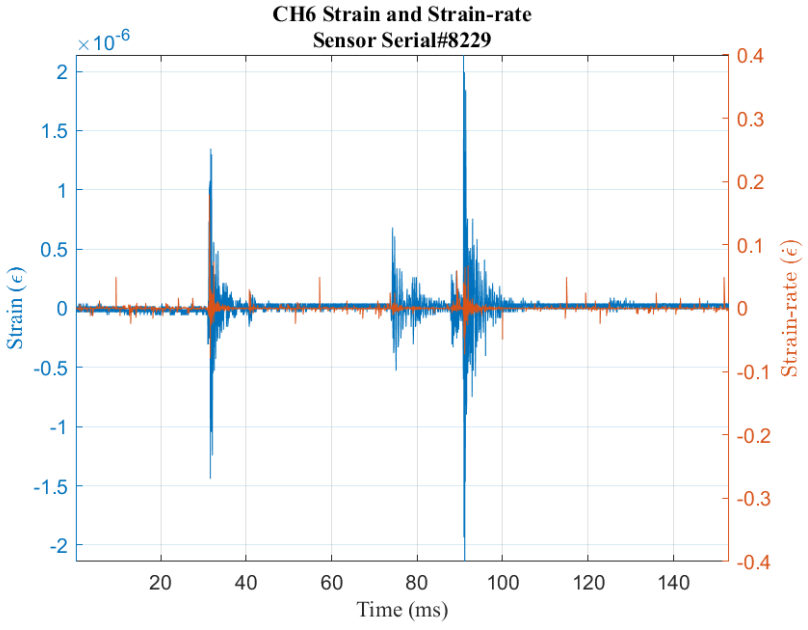


Figure A.8. Strain and strain rate, hard hat, CH6

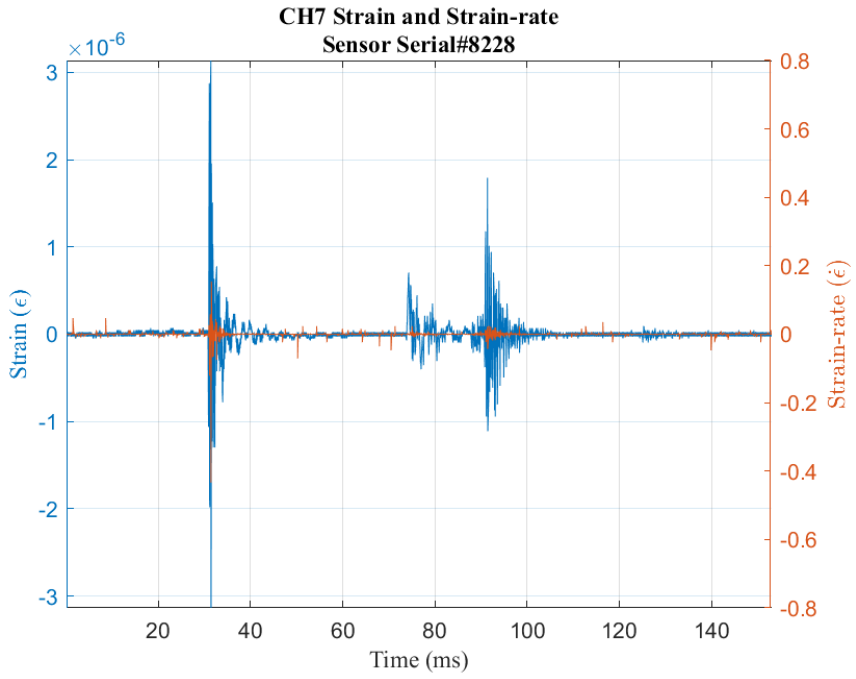


Figure A.9. Strain and strain rate, hard hat, CH7

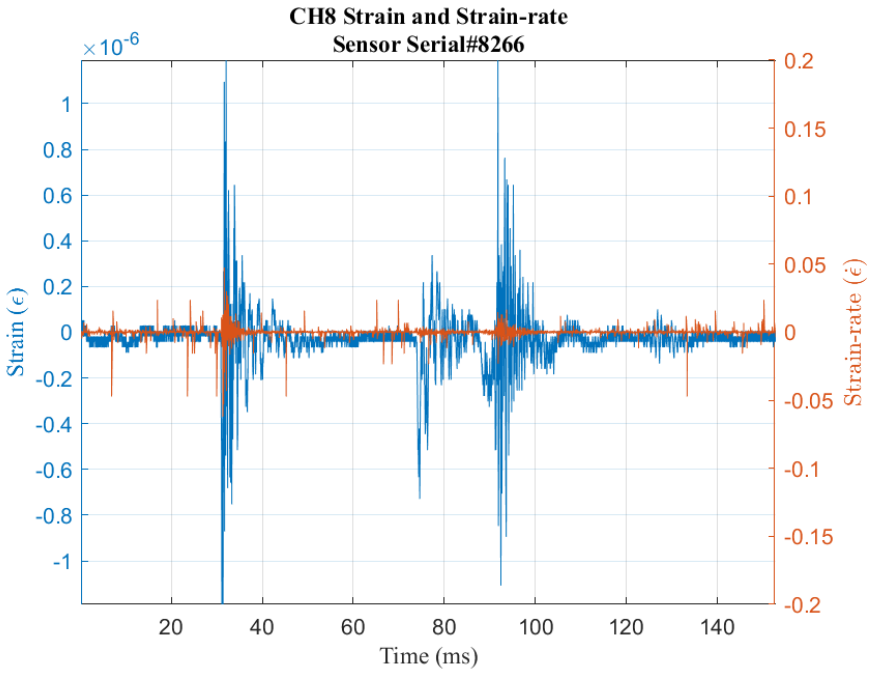
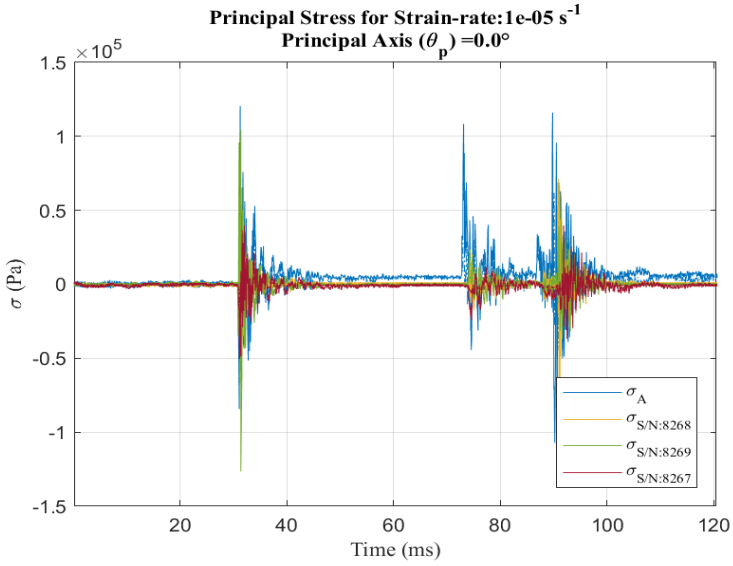
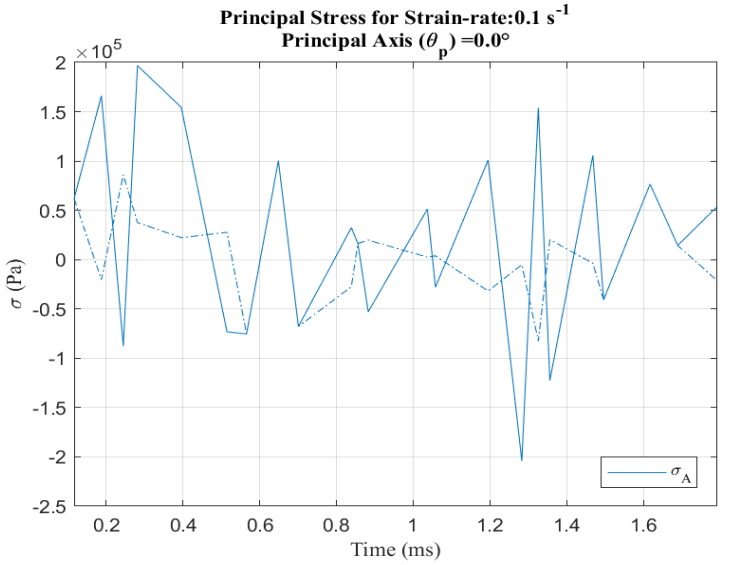


Figure A.10. Strain and strain rate, hard hat, CH8

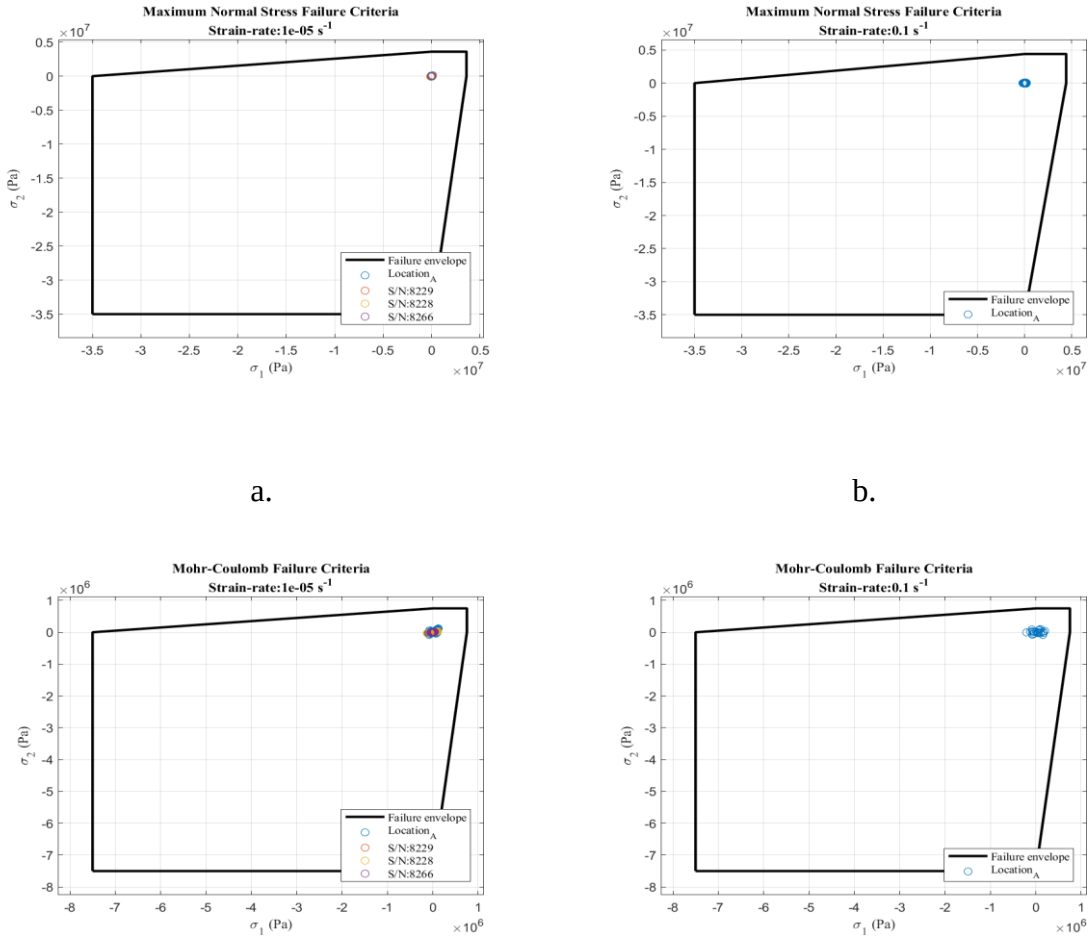


a.



b.

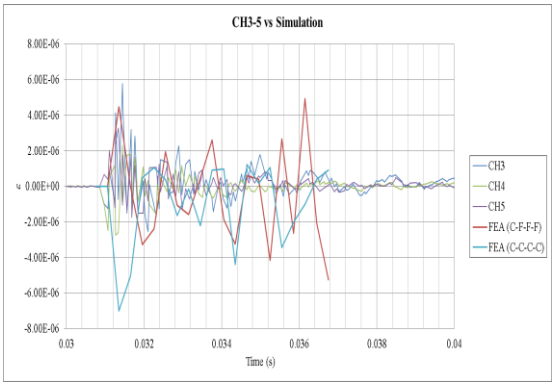
Figure A.11. Principal stress at strain-rate, hard hat, test date: 31 Jul 19



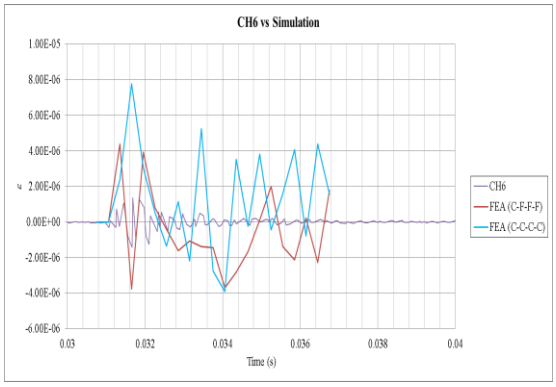
a. b.

c. d.

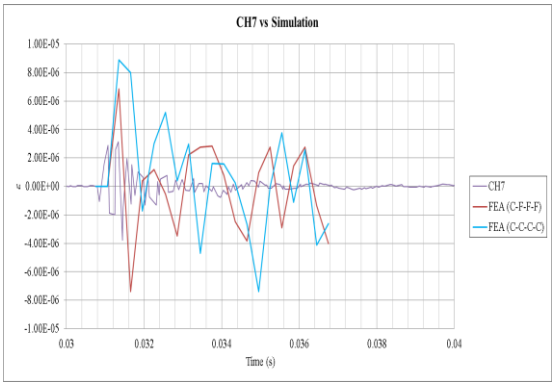
Figure A.12. Failure criterion envelopes, hard hat, test date: 31 Jul 19



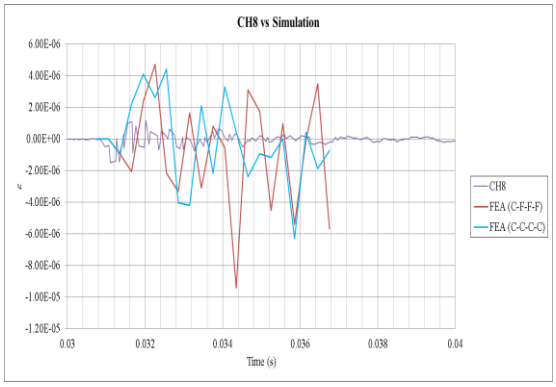
a.



b.



c.



d.

Figure A.13. Strain response and FEA comparison, hard hat, test date: 31 July 19

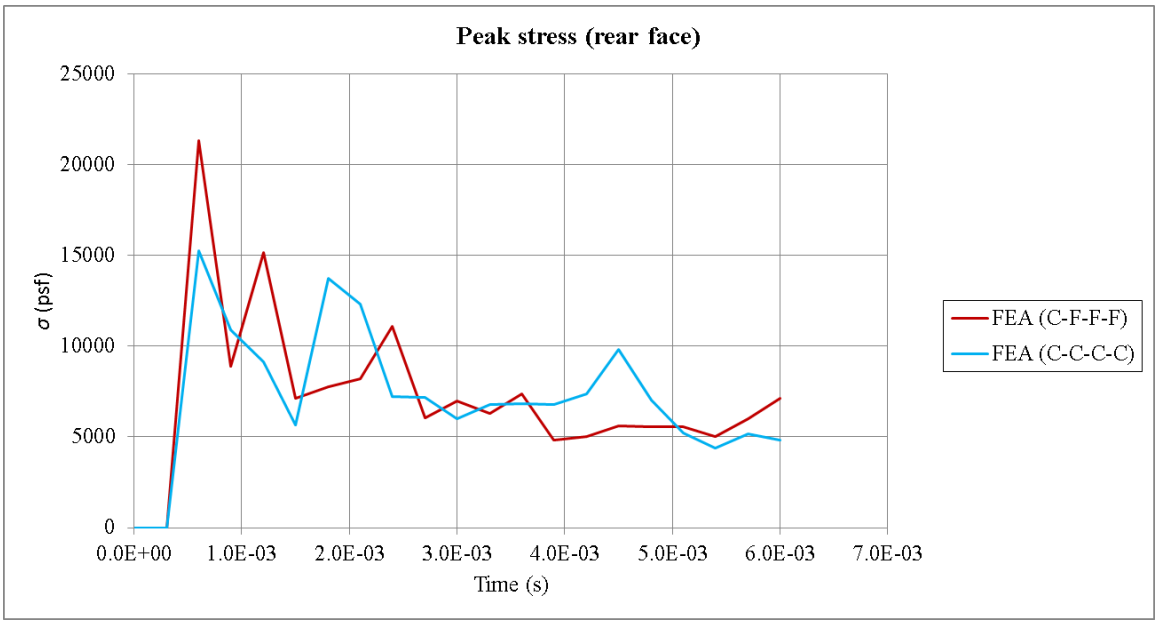
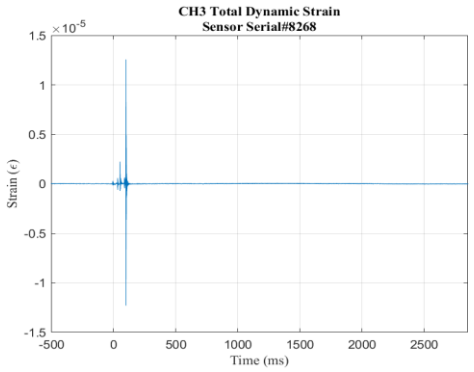


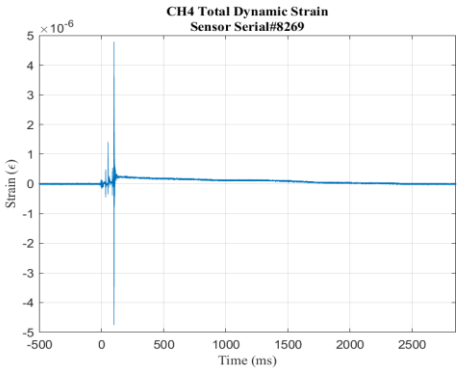
Figure A.14. FEA peak rear face stress, hard hat, test date: 31 Jul 19

B, Water jug, 31 July 2019

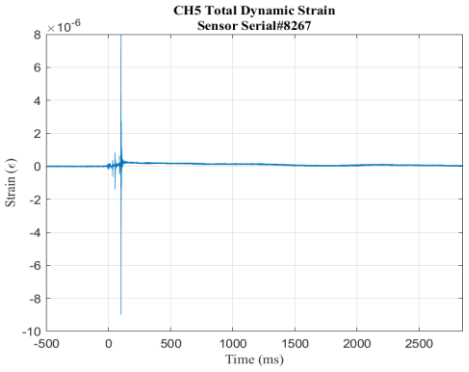
Figure A.15. Water jug



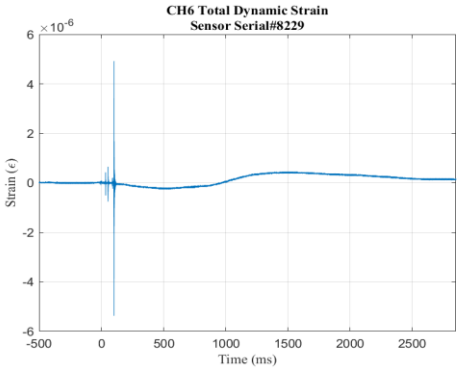
a.



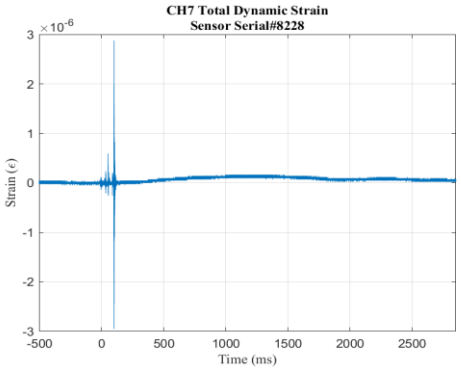
b.



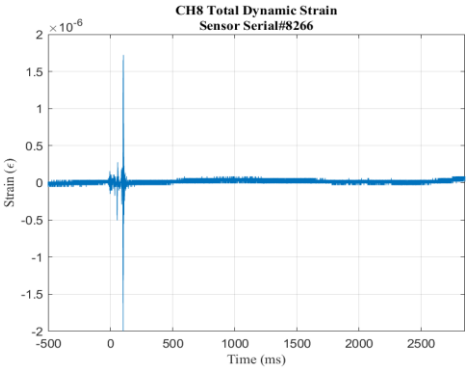
c.



d.



e.



f.

Figure A.16. Strain response, water jug, test date: 31 Jul 19

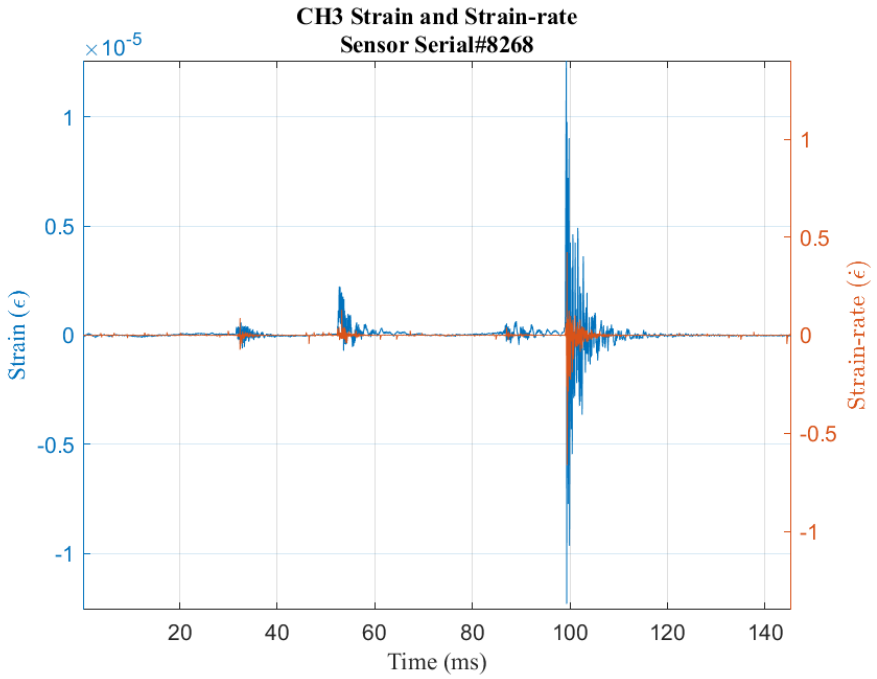


Figure A.17. Strain and strain rate, water jug, CH3

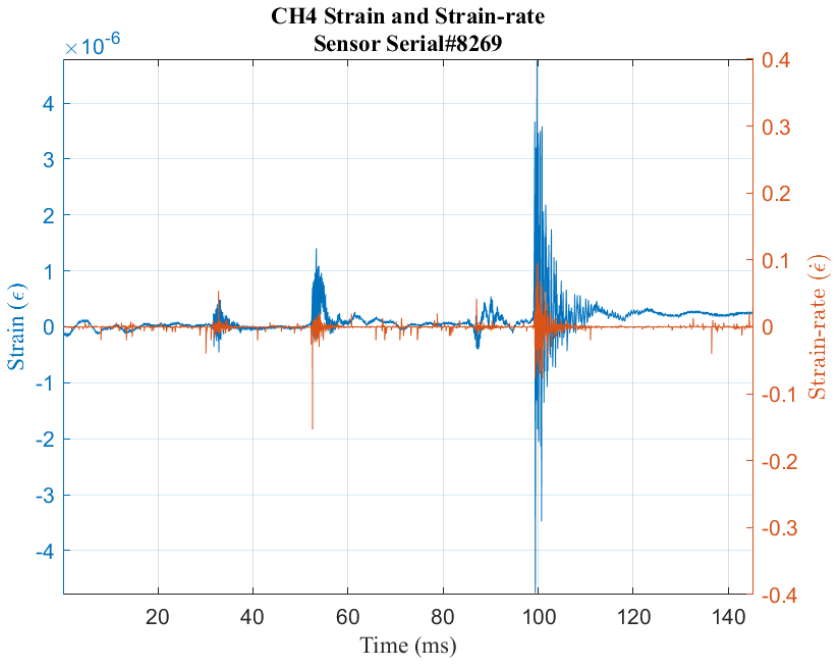


Figure A.18. Strain and strain rate, water jug, CH4

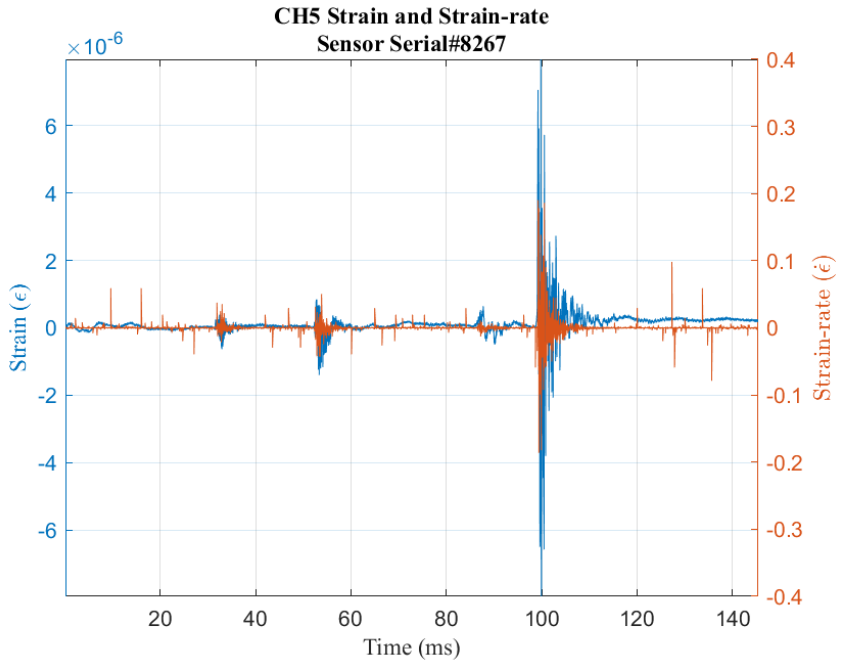


Figure A.19. Strain and strain rate, water jug, CH5

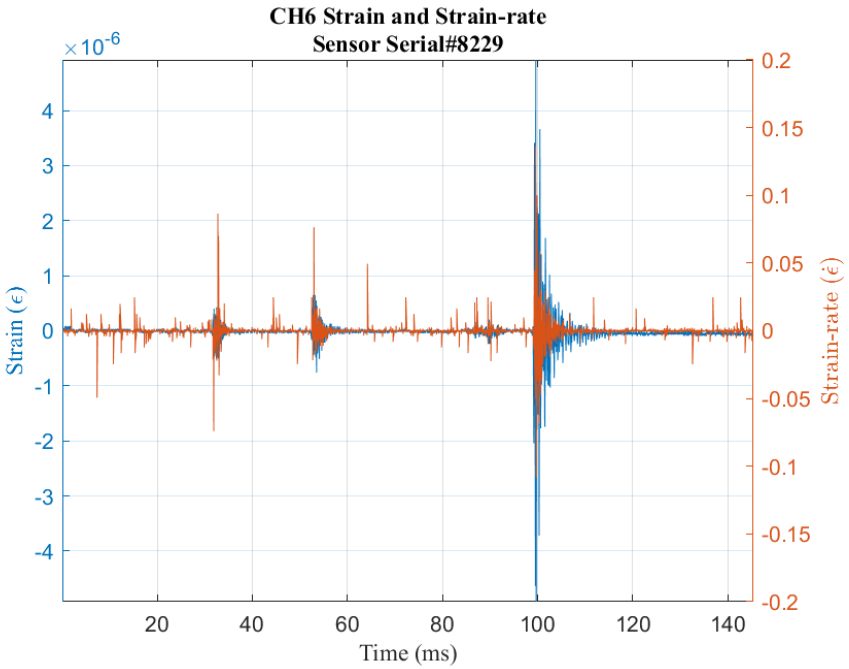


Figure A.20. Strain and strain rate, water jug, CH6

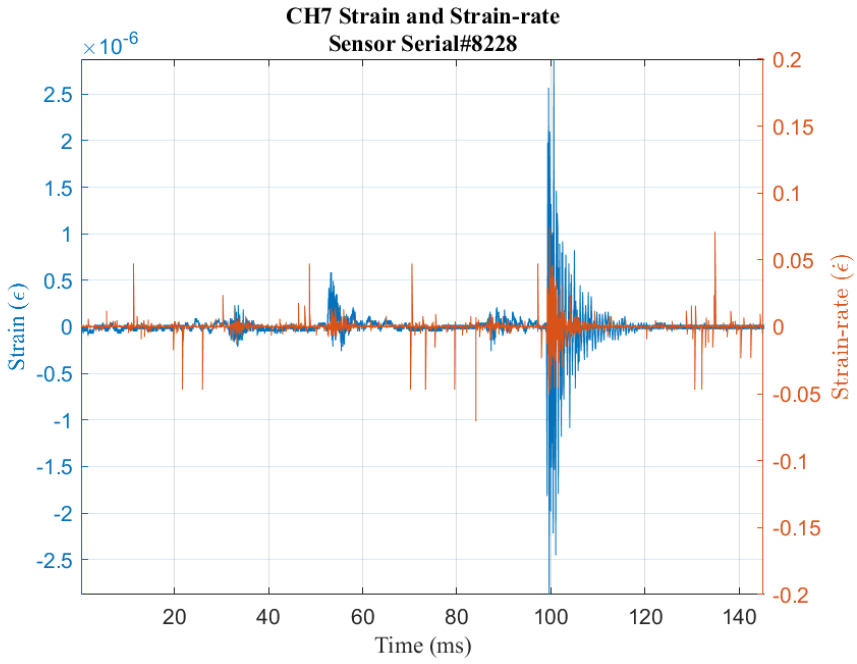


Figure A.21. Strain and strain rate, water jug, CH7

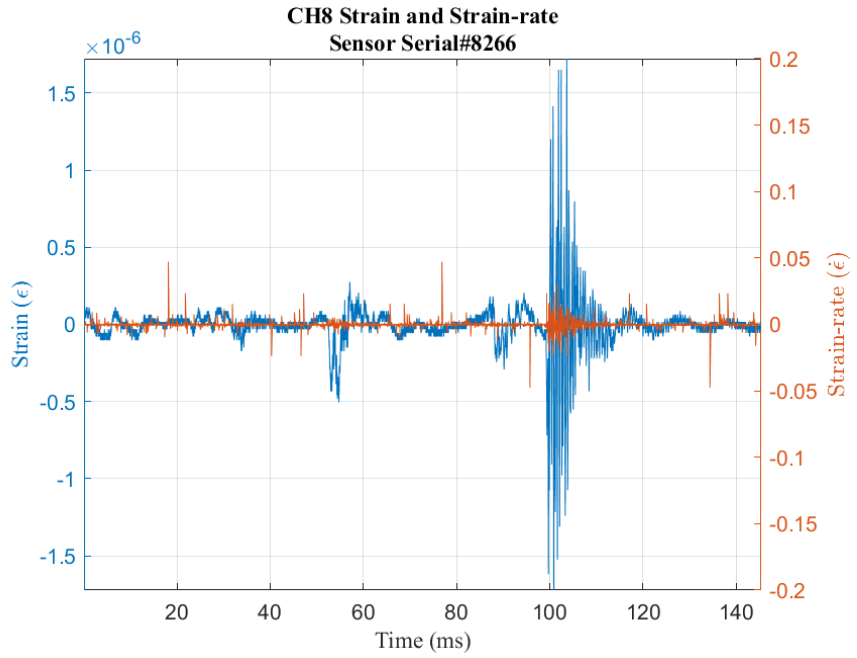
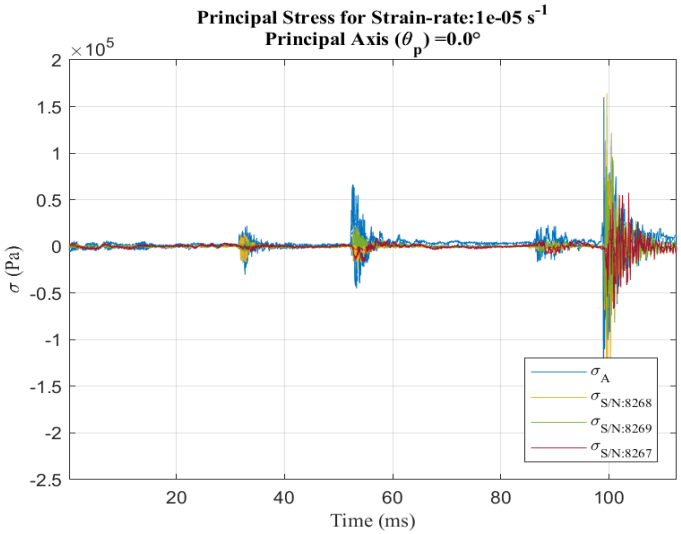
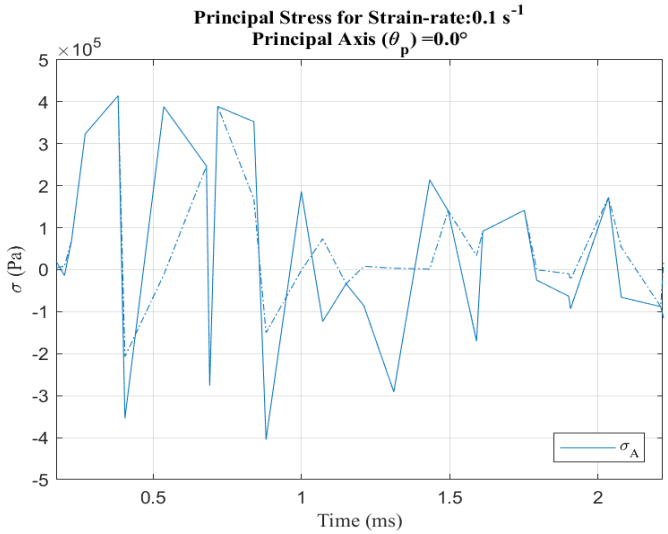


Figure A.22. Strain and strain rate, water jug, CH8

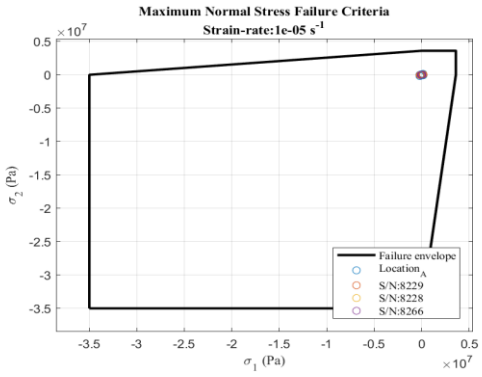


a.

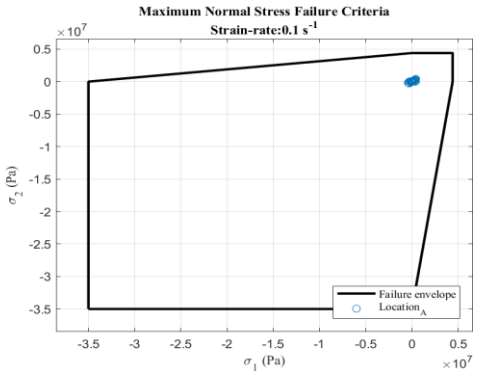


b.

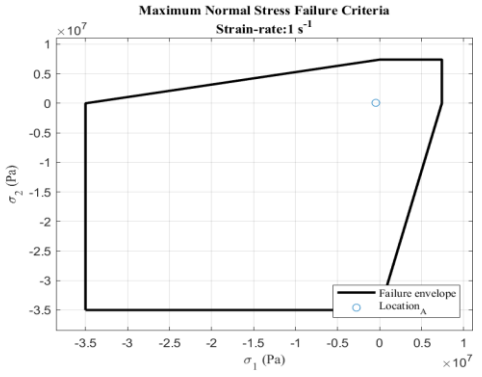
Figure A.23. Principal stress at strain rate, water jug, test date: 31 Jul 19



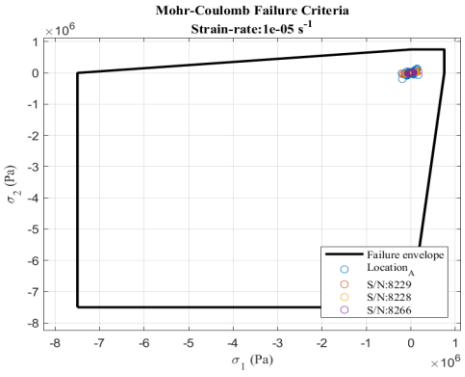
a.



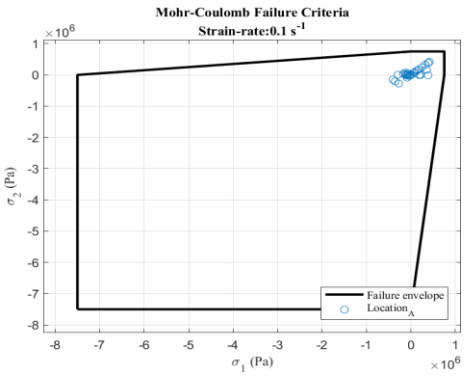
b.



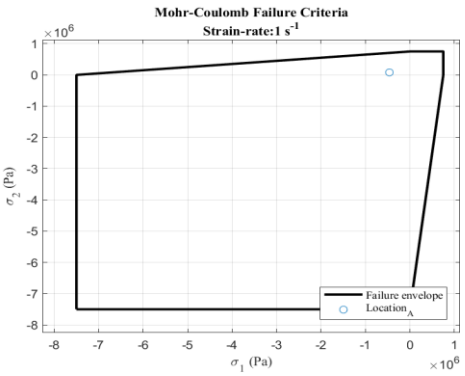
c.



d.

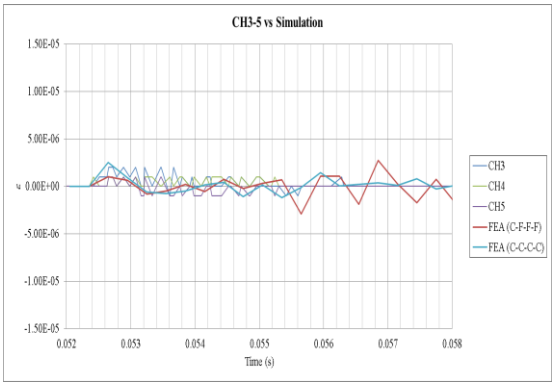


e.

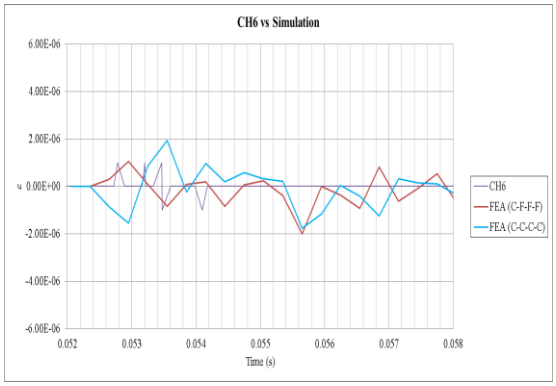


f.

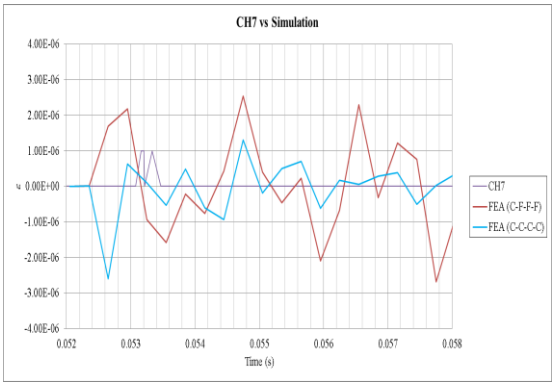
Figure A.24. Failure criterion envelopes, water jug, test date: 31 Jul 19



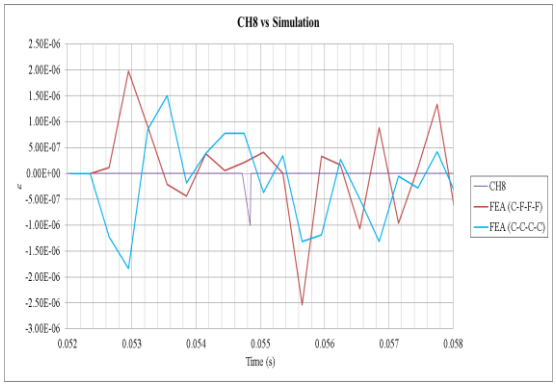
a.



b.



c.



d.

Figure A.25. Strain response and FEA comparison, water jug, test date: 31 Jul 19

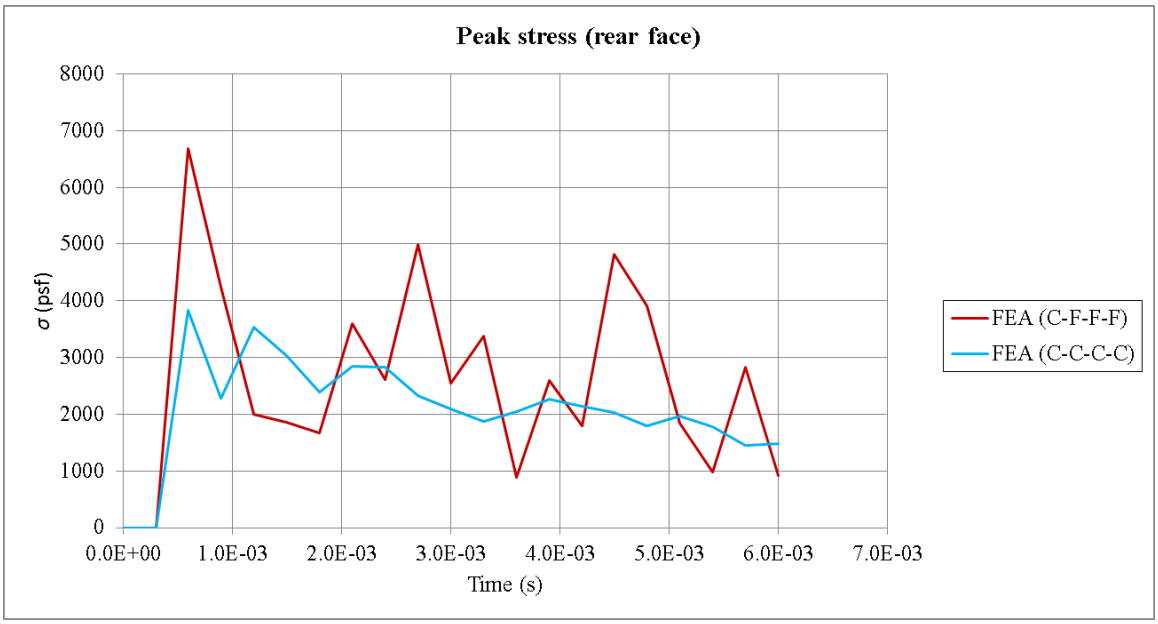
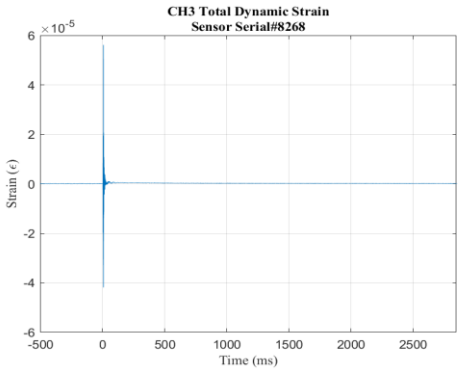


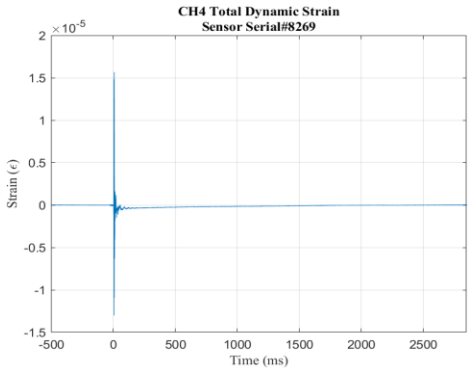
Figure A.26. FEA peak rear face stress, water jug, test date: 31 Jul 19

C, Hand tools, 31 July 2019

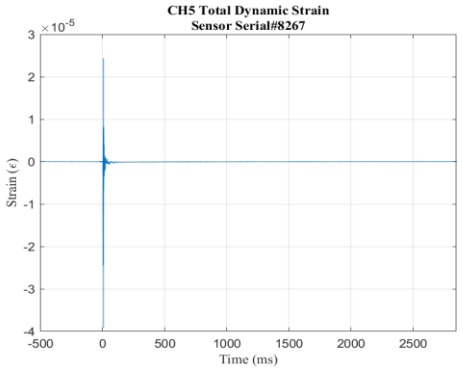
Figure A.27. Hand tools



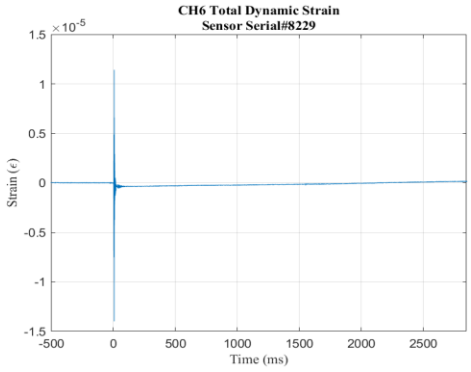
a.



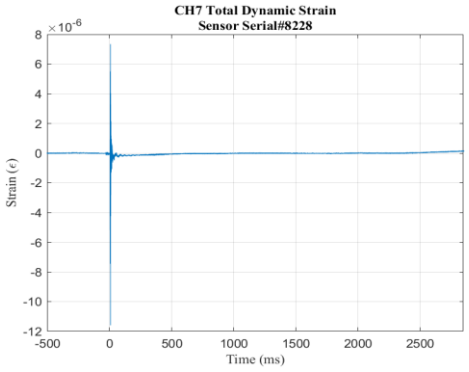
b.



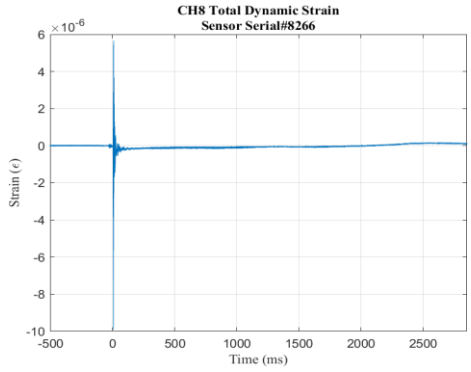
c.



d.



e



f

Figure A.28. Strain response, hand tools, test date: 31 Jul 19

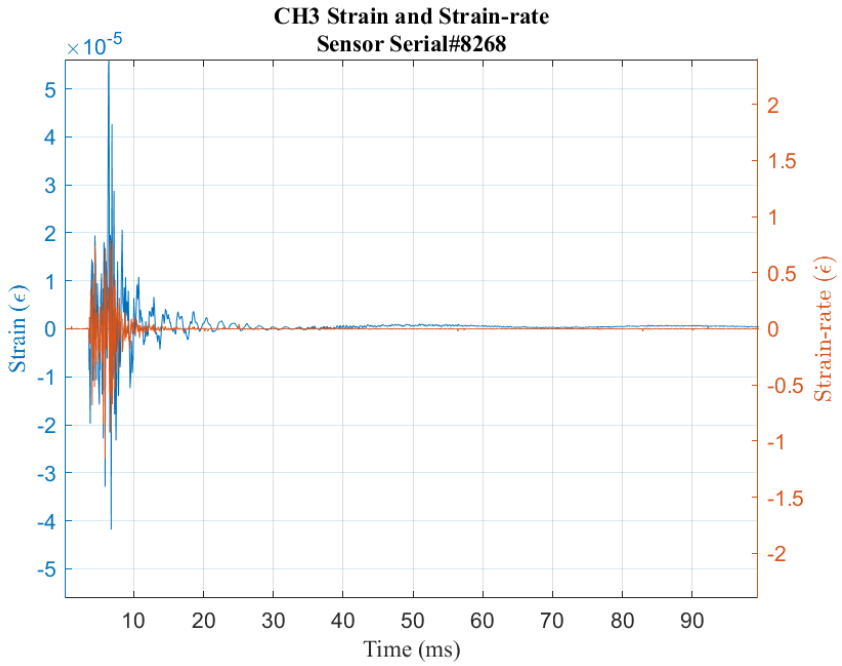


Figure A.29. Strain and strain rate, hand tools, CH3

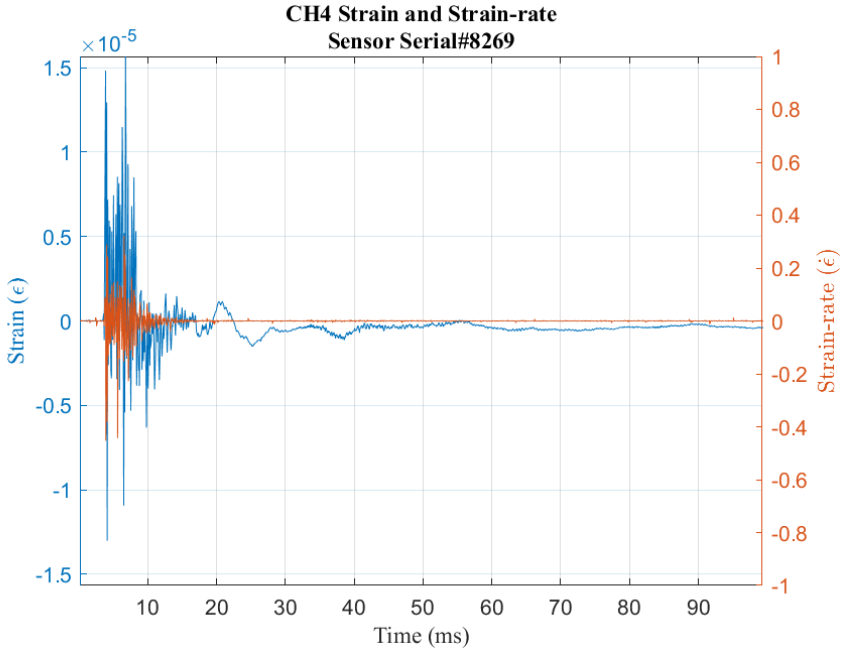


Figure A.30. Strain and strain rate, hand tools, CH4

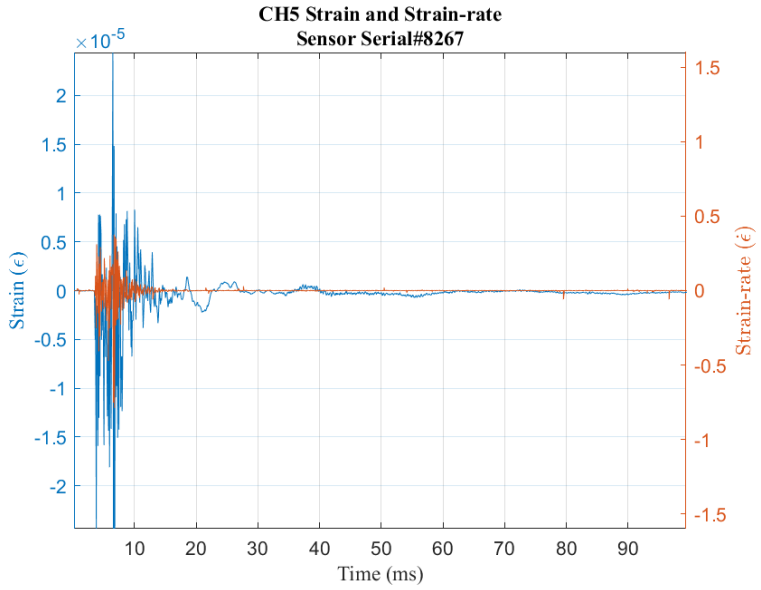


Figure A.31. Strain and strain rate, hand tools, CH5

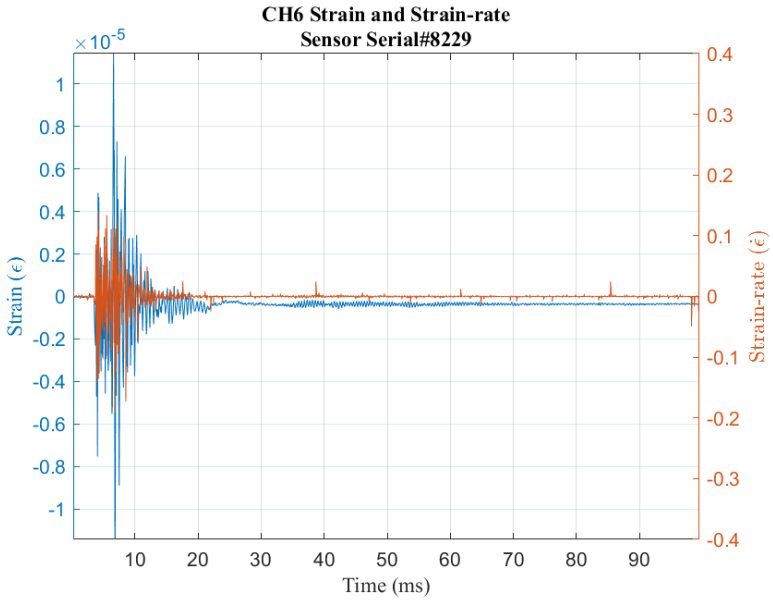


Figure A.32. Strain and strain rate, hand tools, CH6

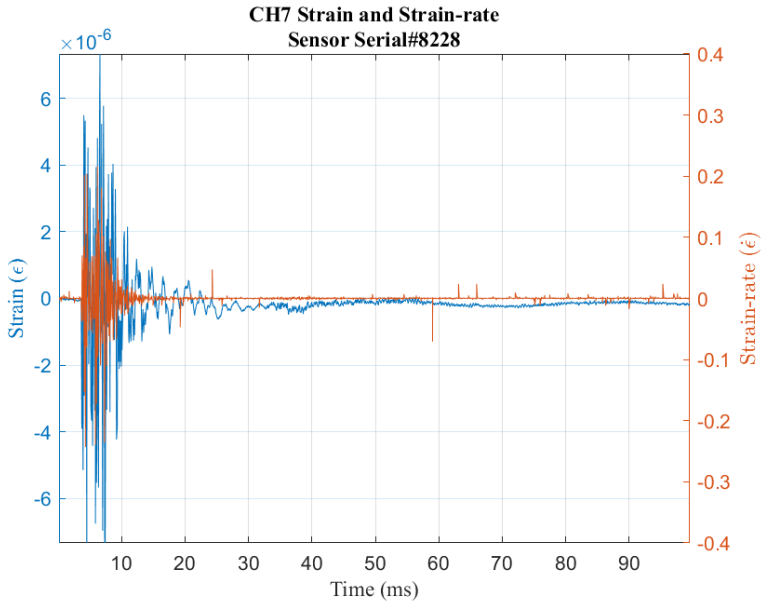


Figure A.33. Strain and strain rate, hand tools, CH3

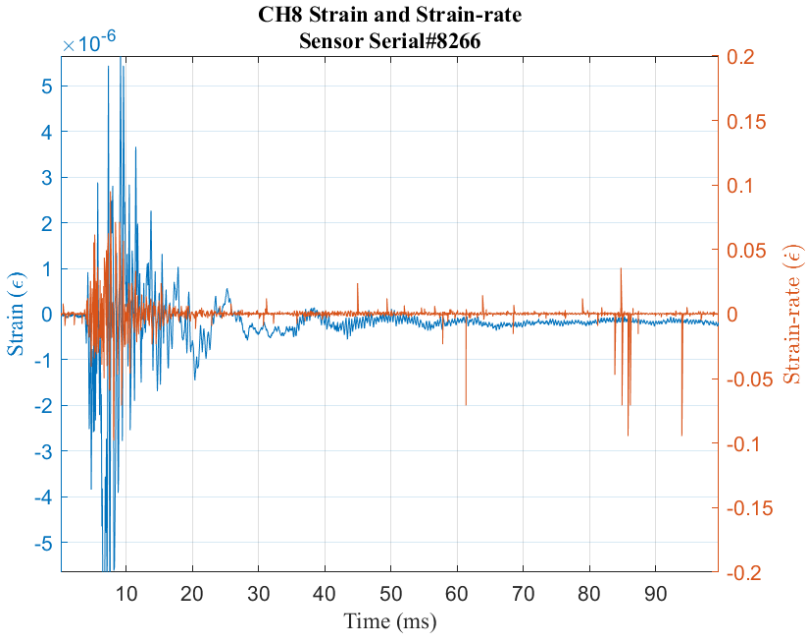
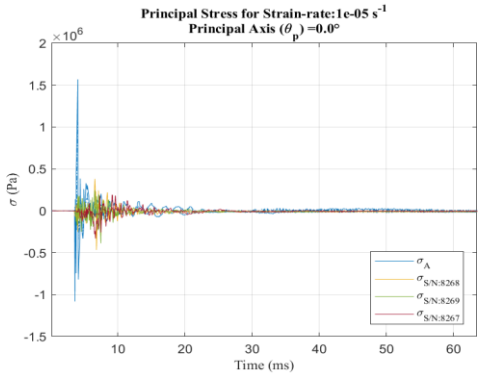
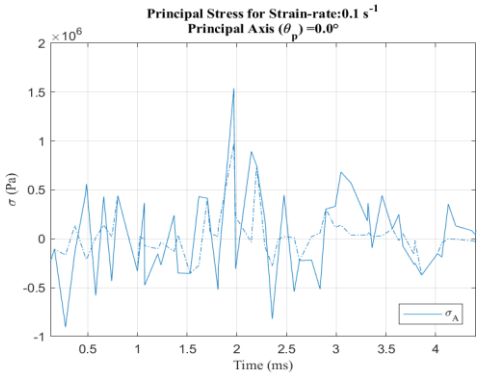


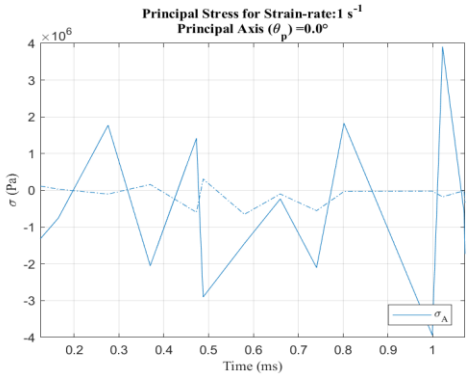
Figure A.34. Strain and strain rate, hand tools, CH8



a.

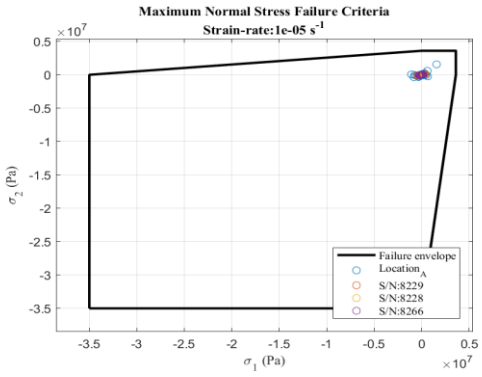


b.

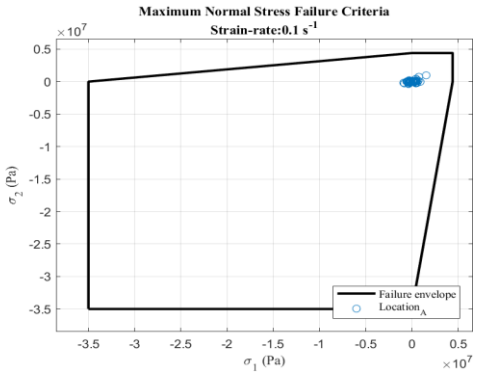


c.

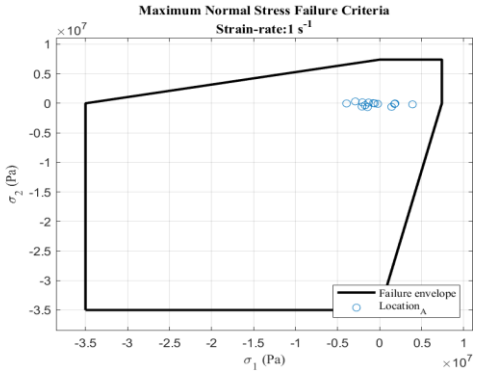
Figure A.35. Principal stress at strain rate, hand tools, test date: 31 Jul 19



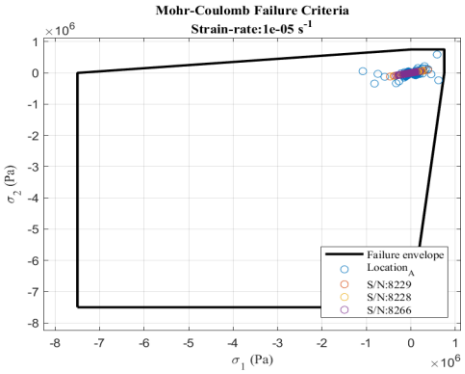
a.



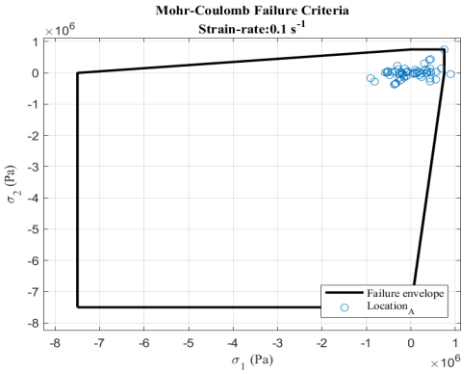
b.



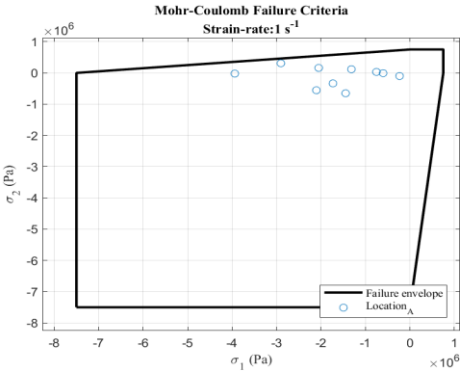
c.



d.

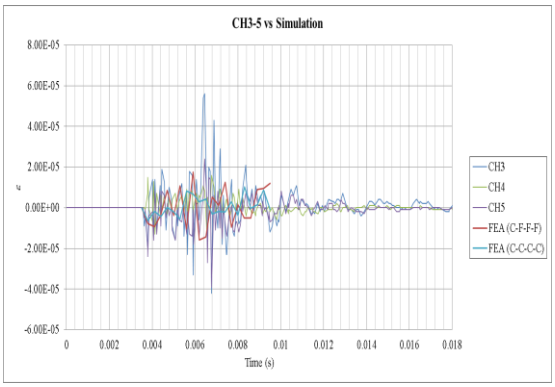


e.

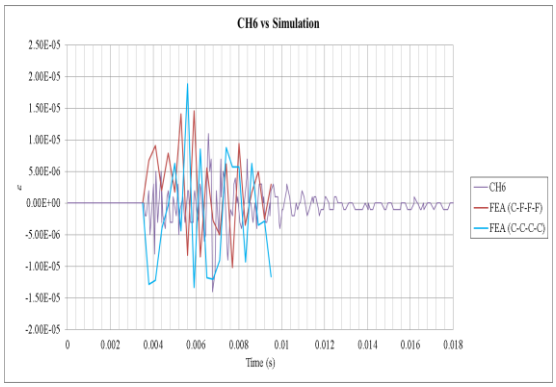


f.

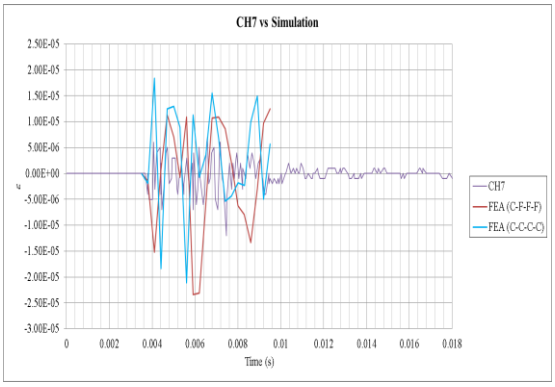
Figure A.36. Failure criterion envelopes, hand tools, test date: 31 Jul 19



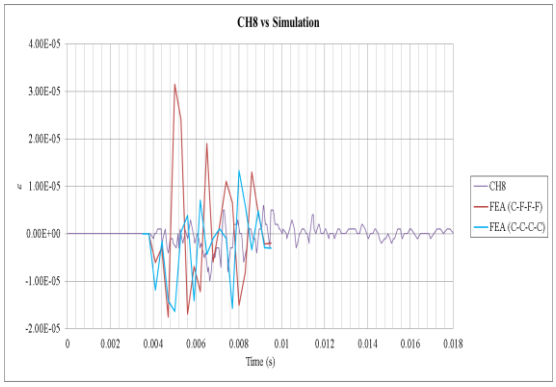
a.



b.



c.



d.

Figure A.37. Strain response and FEA comparison, hand tools, test date: 31 Jul 19

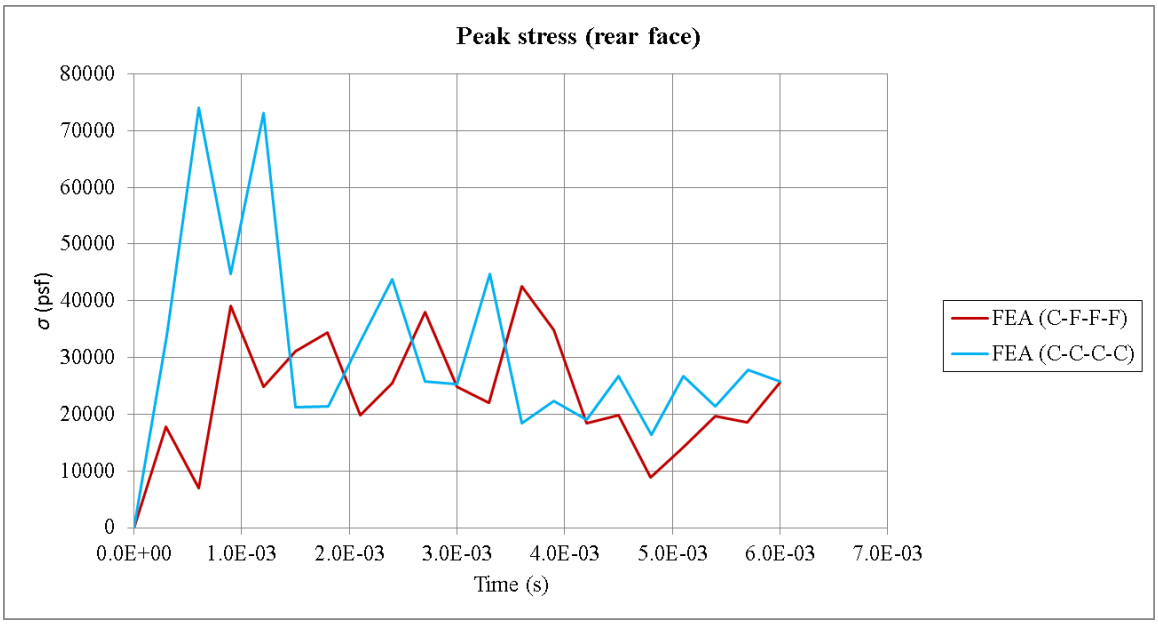
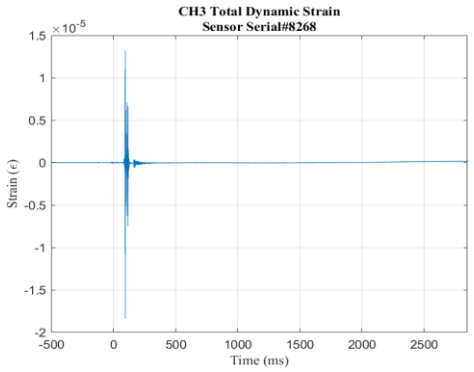


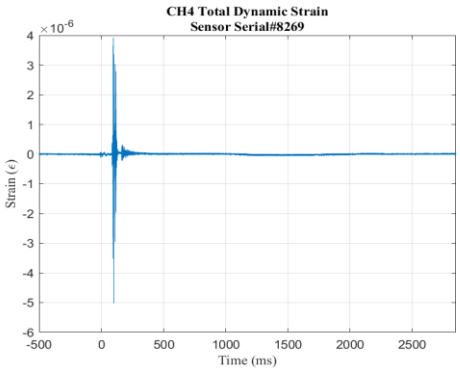
Figure A.38. FEA peak rear face stress, hand tools, test date: 31 Jul 19

D, Roof bolt plates, 31 July 2019

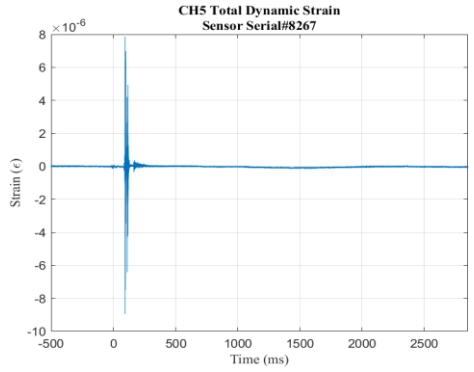
Figure A.39. Roof bolt plates



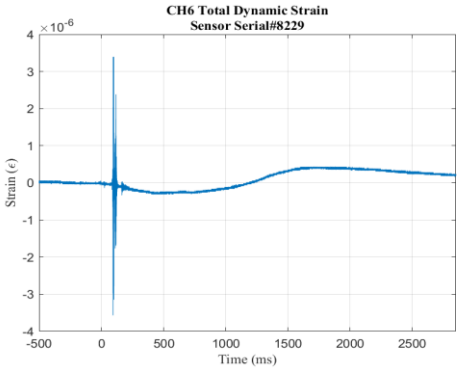
a.



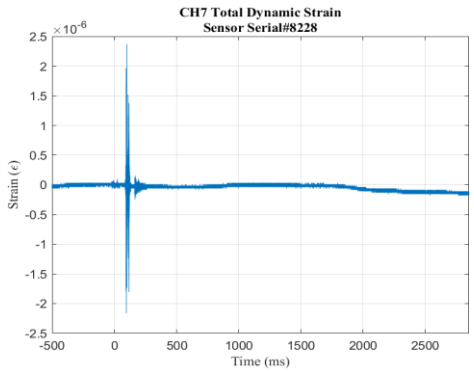
b.



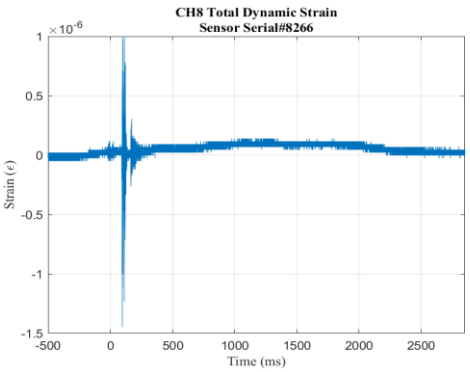
c.



d.



e.



f.

Figure A.40. Strain response, roof bolt plates, test date: 31 Jul 19

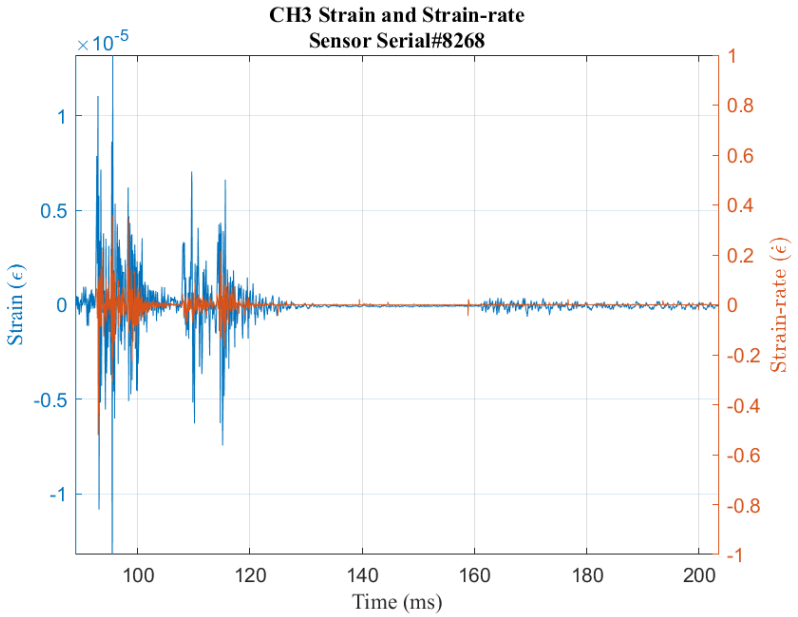


Figure A.41. Strain and strain rate, roof bolt plates, CH3

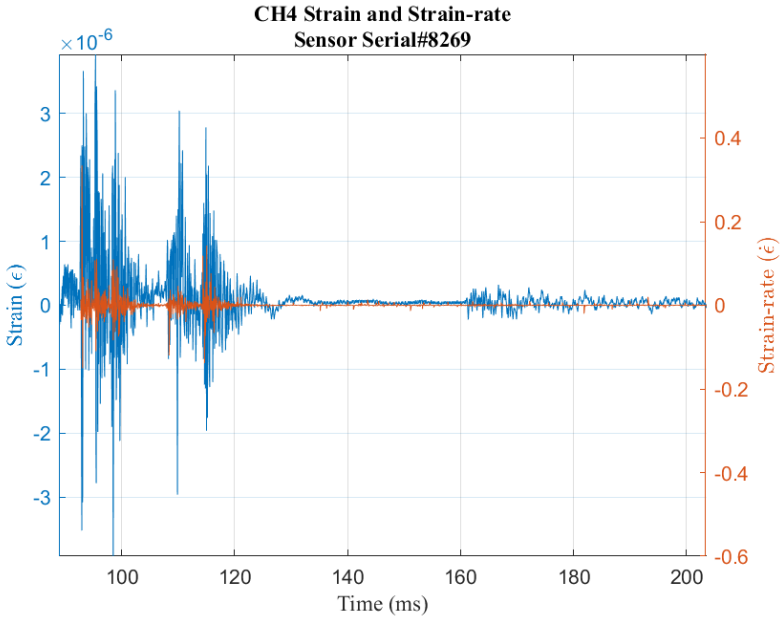


Figure A.42. Strain and strain rate, roof bolt plates, CH4

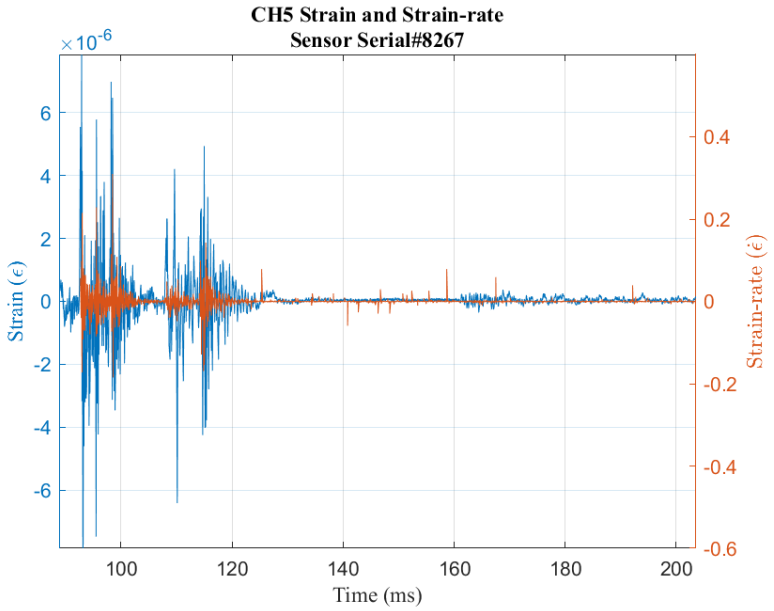


Figure A.43. Strain and strain rate, roof bolt plates, CH5

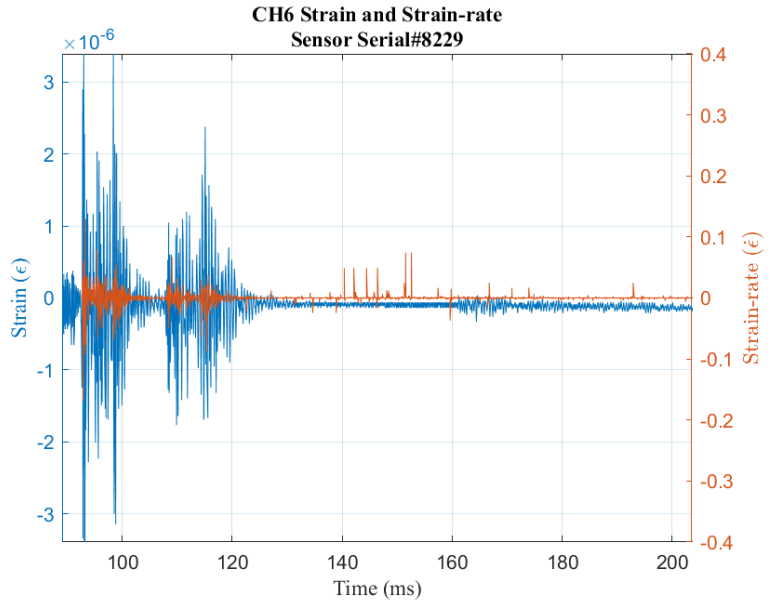


Figure A.44. Strain and strain rate, roof bolt plates, CH6

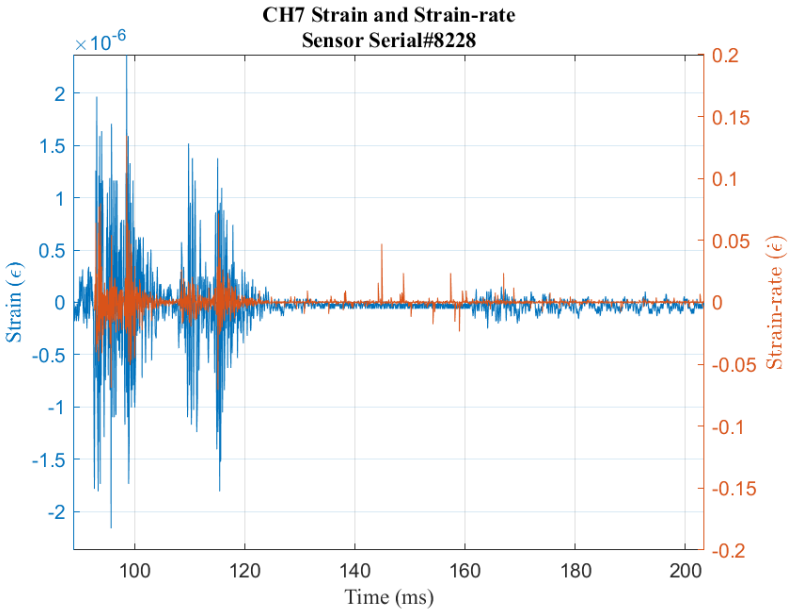


Figure A.45. Strain and strain rate, roof bolt plates, CH7

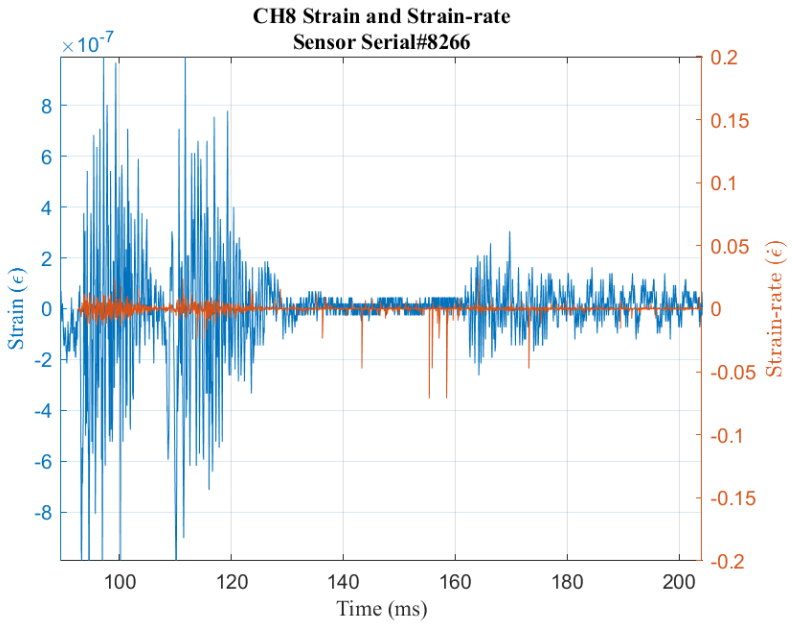
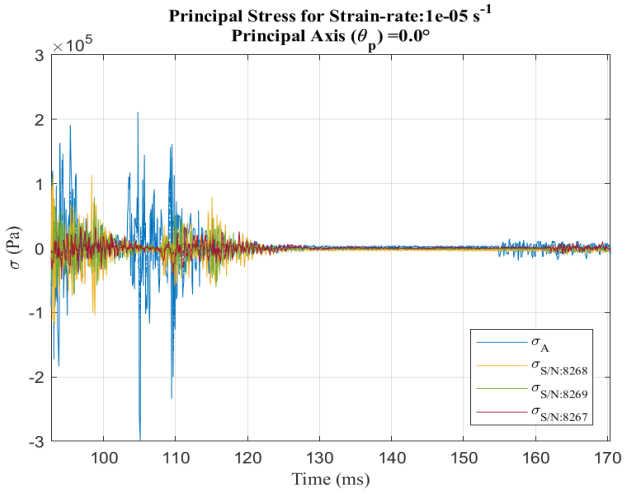
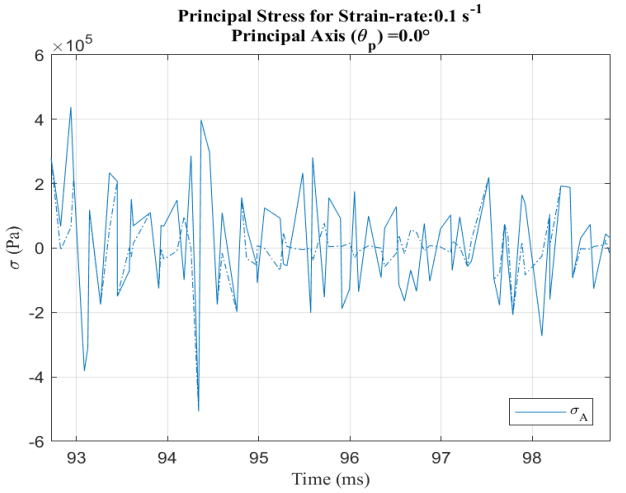


Figure A.46. Strain and strain rate, roof bolt plates, CH8

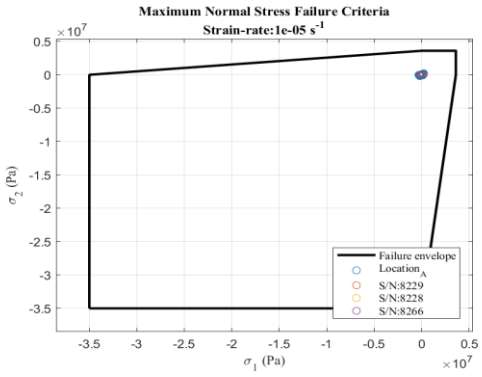


a.

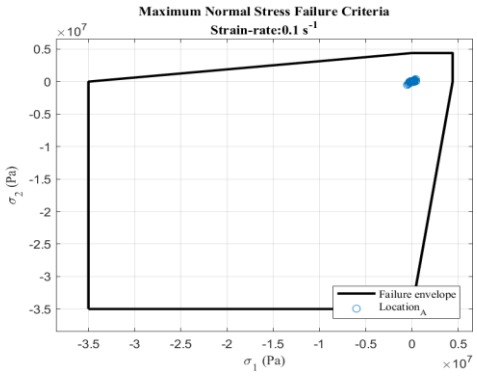


b.

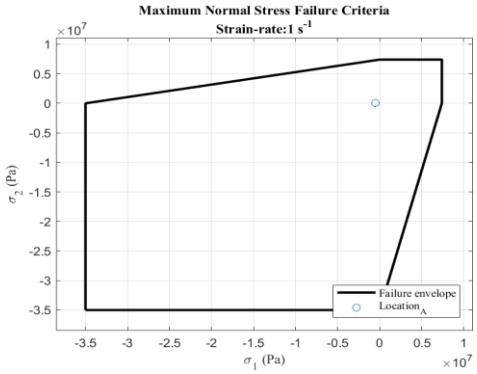
Figure A.47. Principal stress at strain rate, roof bolt plates, test date: 31 Jul 19



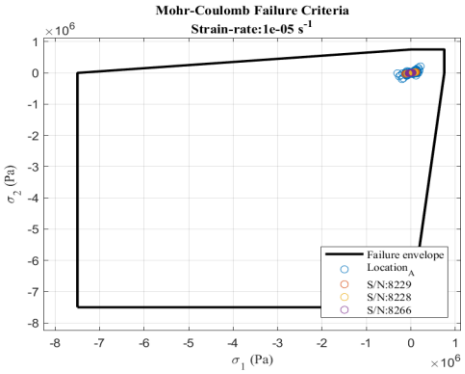
a.



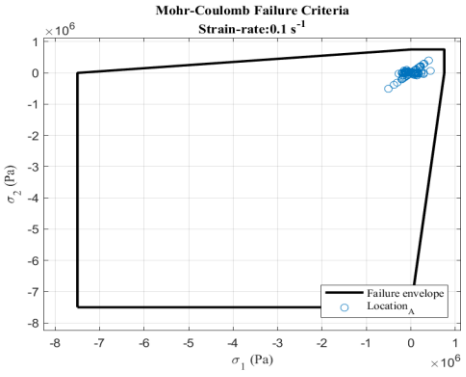
b.



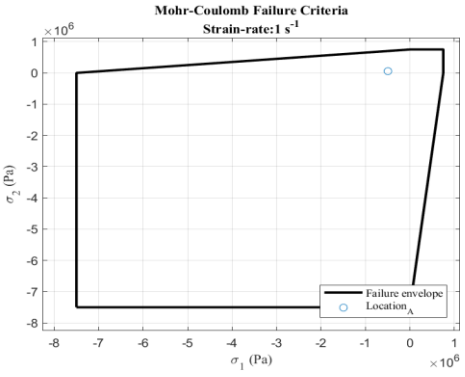
c.



d.

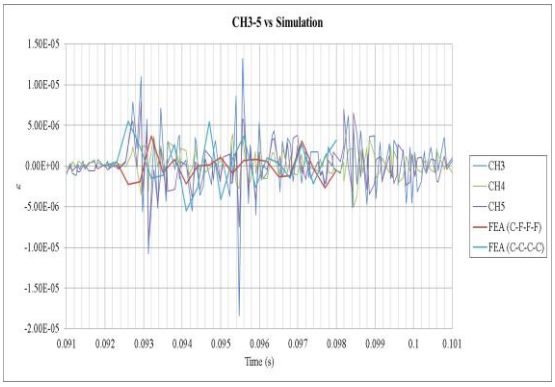


e.

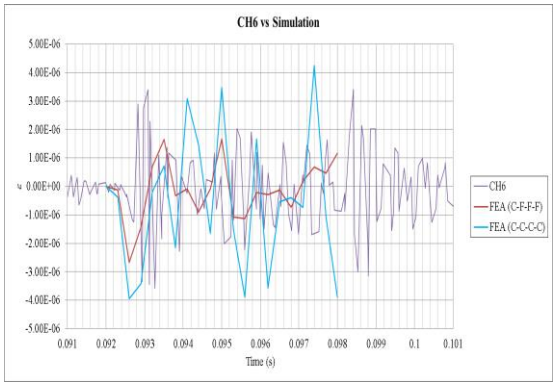


f.

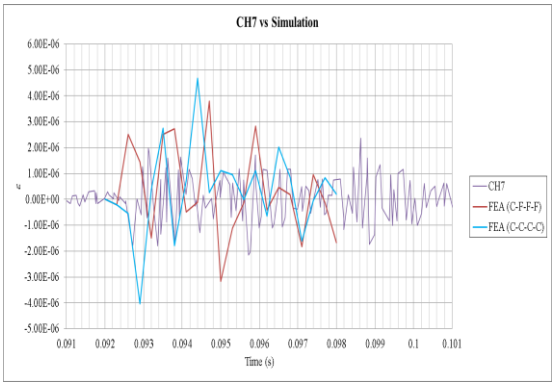
Figure A.48. Failure criterion envelopes, roof bolt plates, test date: 31 Jul 19



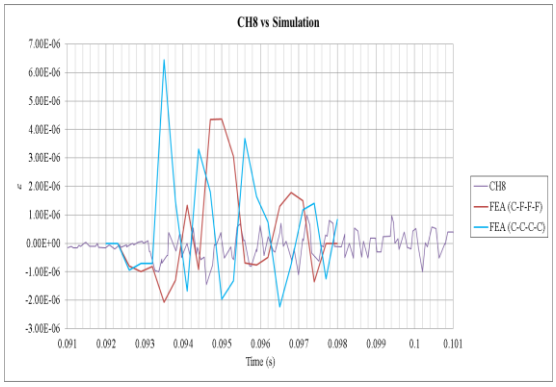
a.



b.



c.



d.

Figure A.49. Strain response and FEA comparison, roof bolt plates, test date: 13 Aug 19

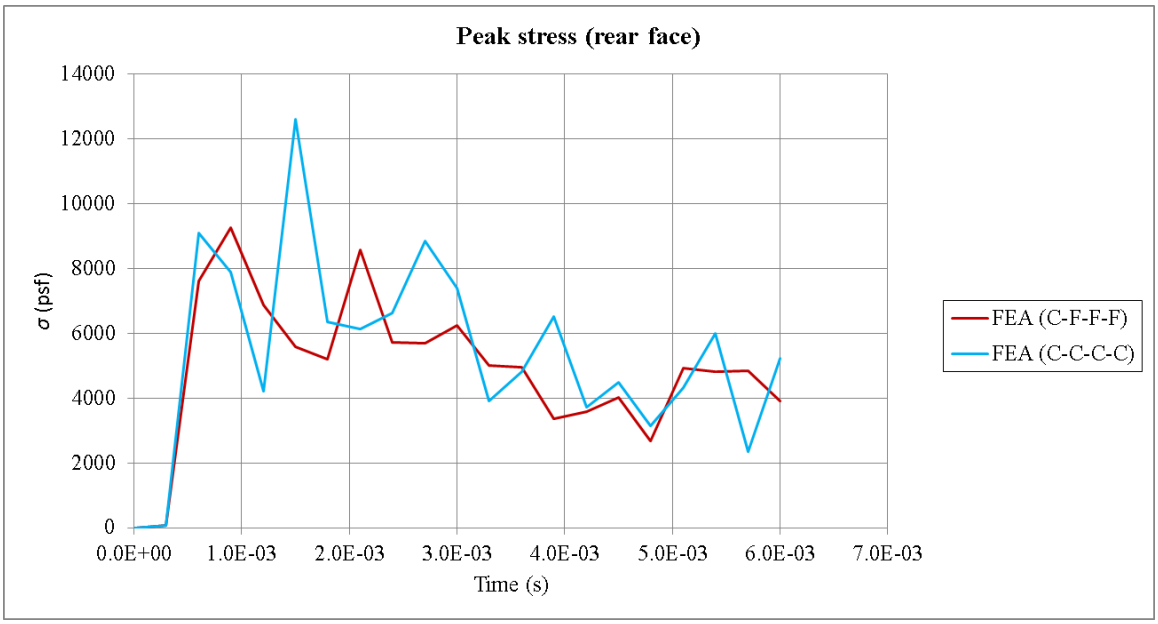
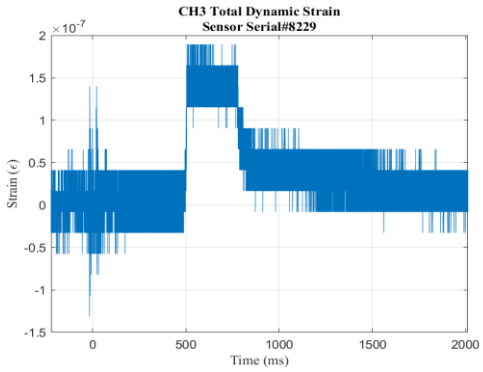


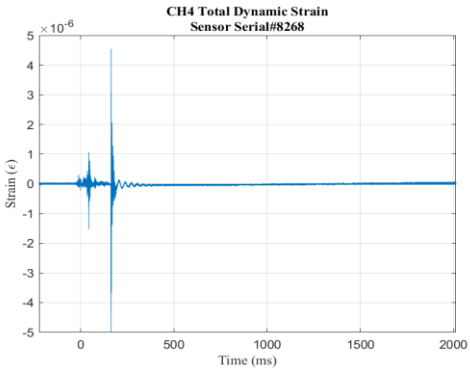
Figure A.50. FEA peak rear face stress, roof bolt plates, test date: 13 Aug 19

E, 5 x 5 lumber, 13 August 2019

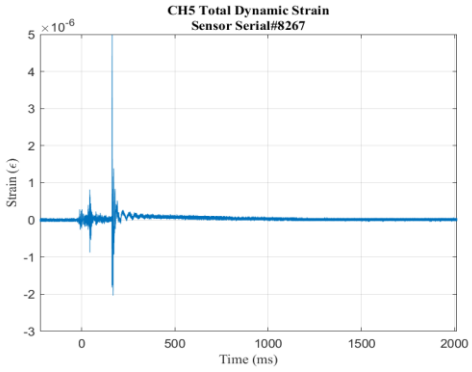
Figure A.51. 5 x 5 lumber



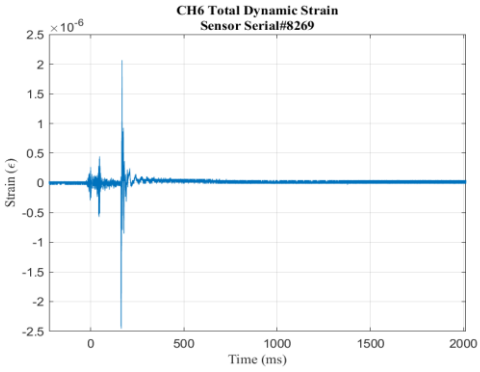
a.



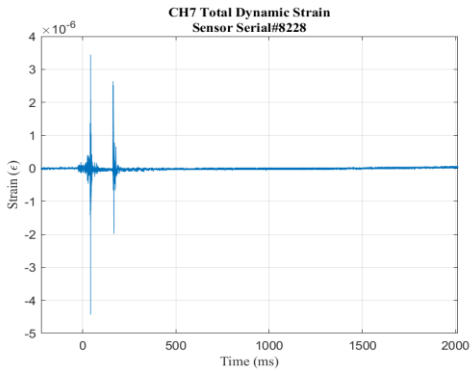
b.



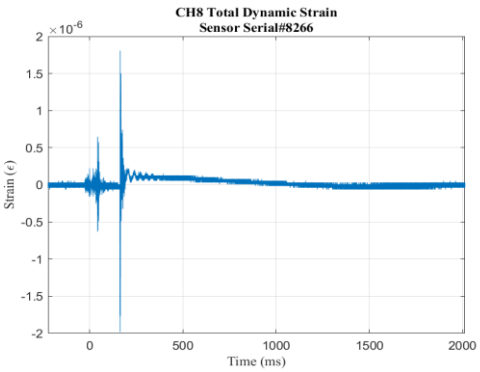
c.



d.



e.



f.

Figure A.52. Strain response, 5 x 5 lumber, test date: 13 Aug 19

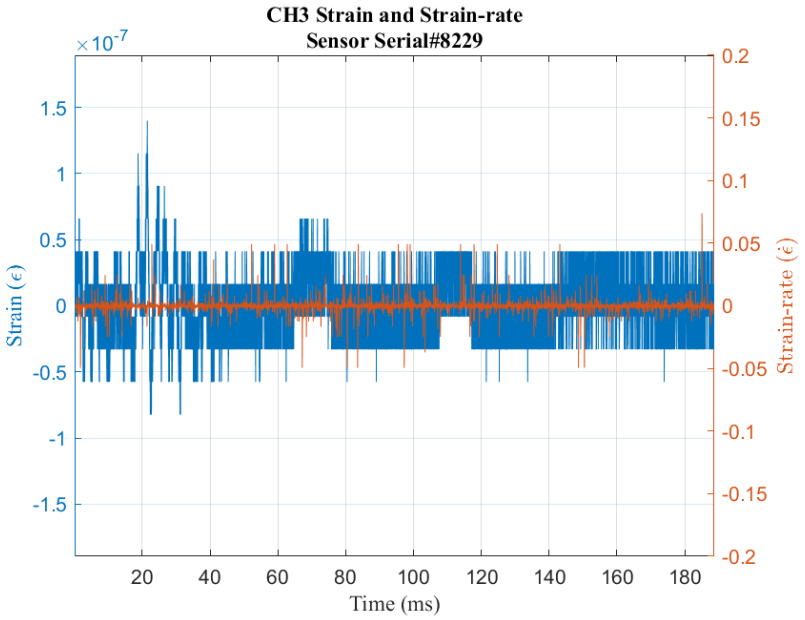


Figure A.53. Strain and strain rate, 5x5 lumber, CH3

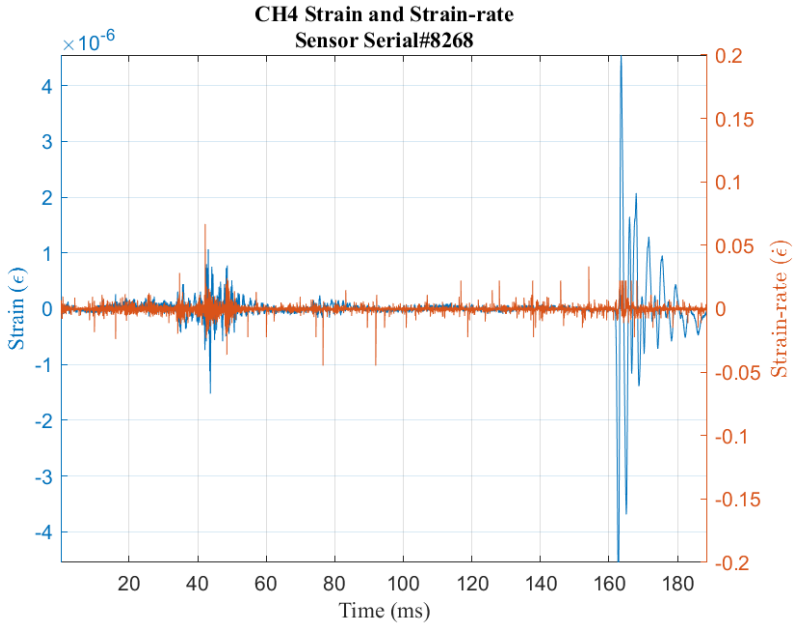


Figure A.54. Strain and strain rate, 5x5 lumber, CH4

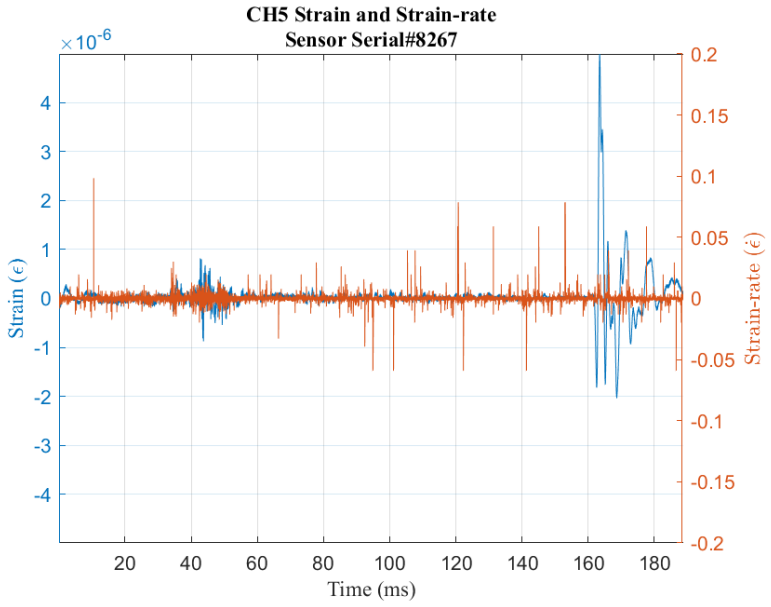


Figure A.55. Strain and strain rate, 5x5 lumber, CH5

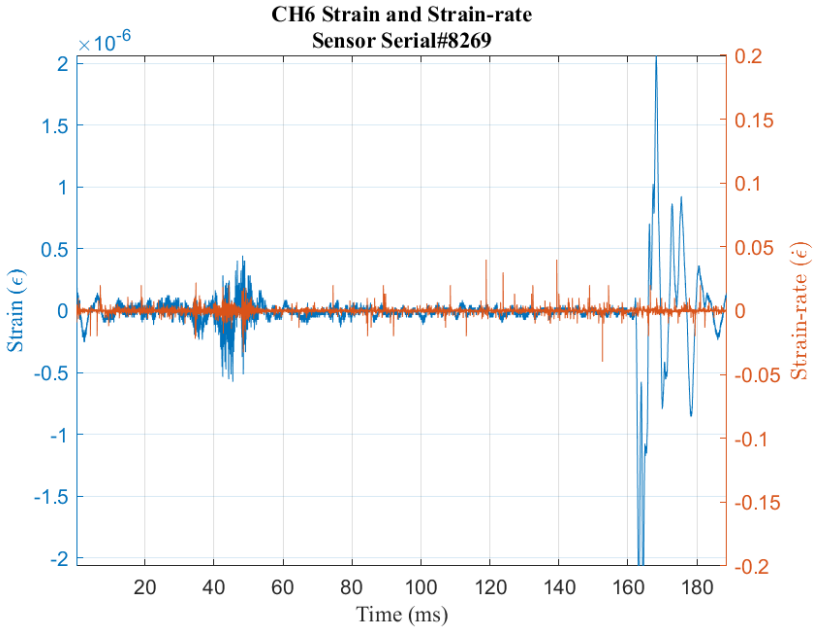


Figure A.56. Strain and strain rate, 5x5 lumber, CH6

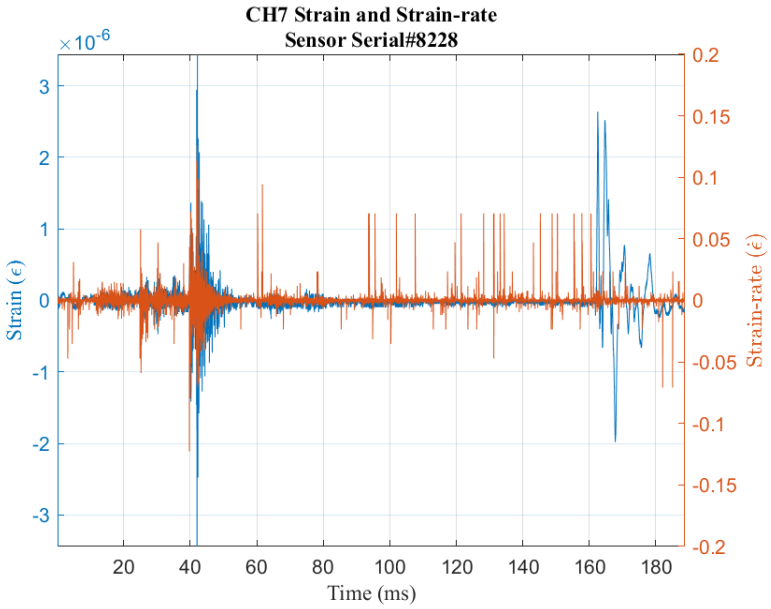


Figure A.57. Strain and strain rate, 5x5 lumber, CH7

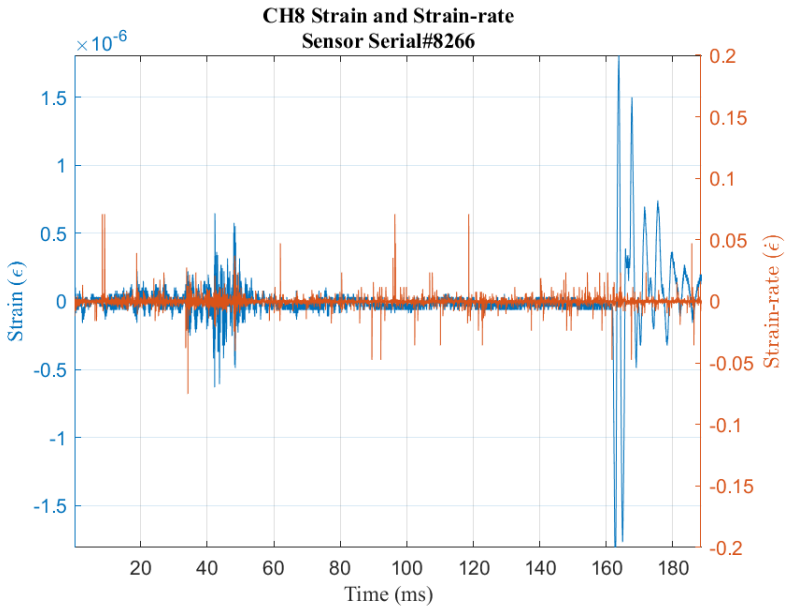
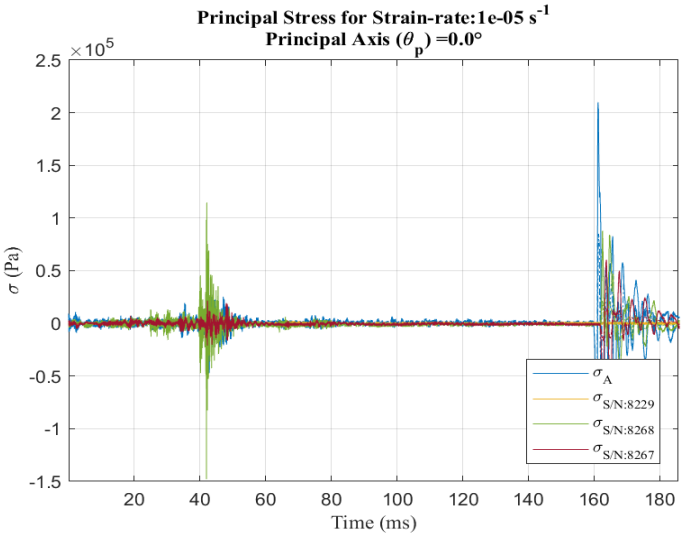
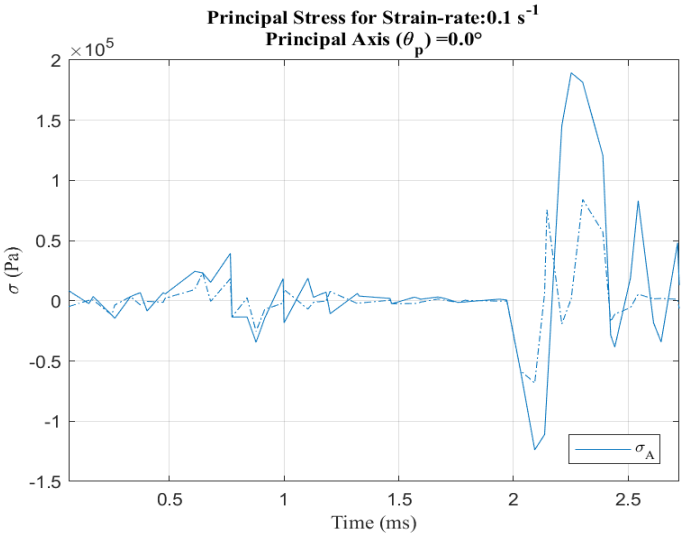


Figure A.58. Strain and strain rate, 5x5 lumber, CH8

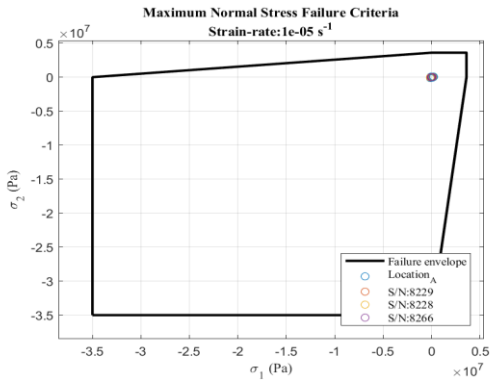


a.

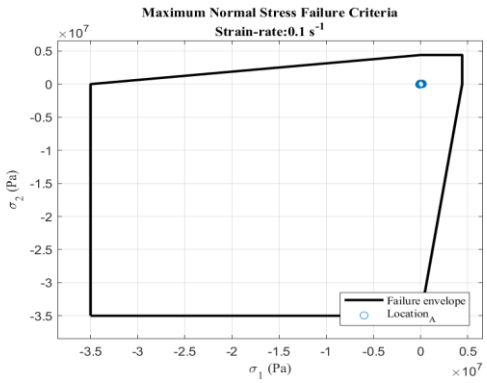


b.

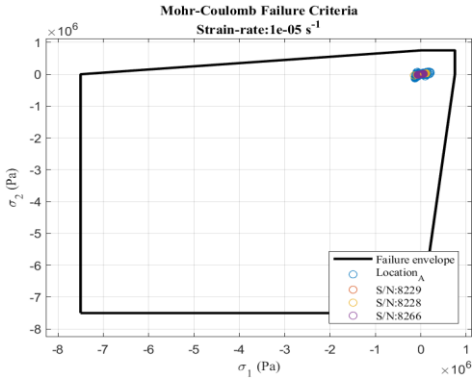
Figure A.59. Principal stress at strain rate, 5 x 5 lumber, test date: 13 Aug 19



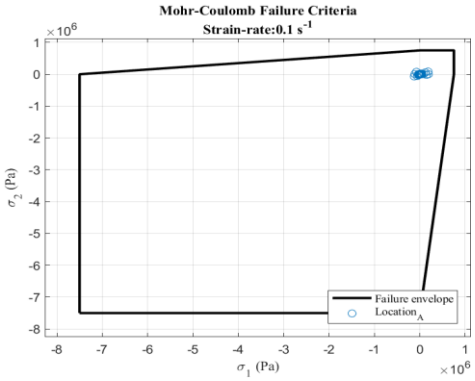
a.



b.

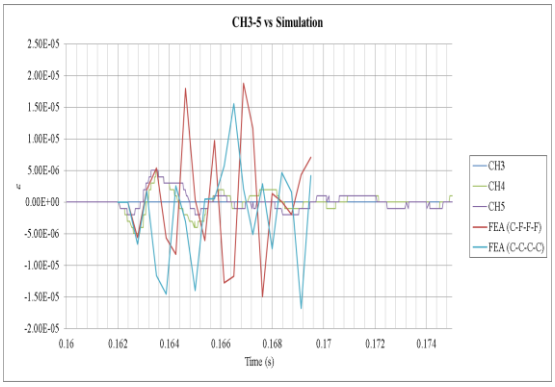


c.

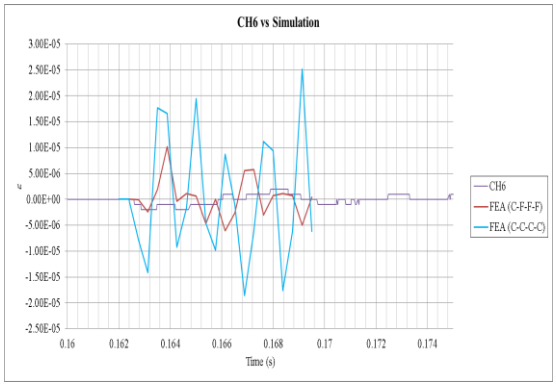


d.

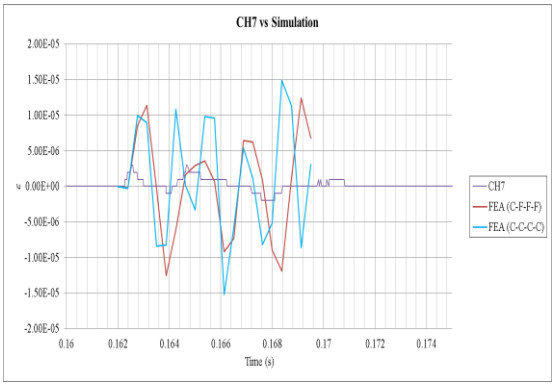
Figure A.60. Failure criterion envelopes, 5 x 5 lumber, test date: 13 Aug 19



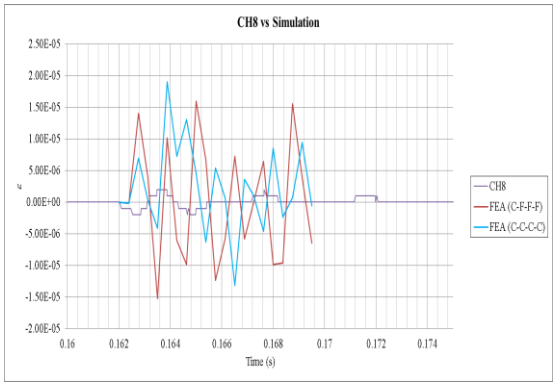
a.



b.



c.



d.

Figure A.61. Strain response and FEA comparison, 5 x 5 lumber, test date: 13 Aug 19

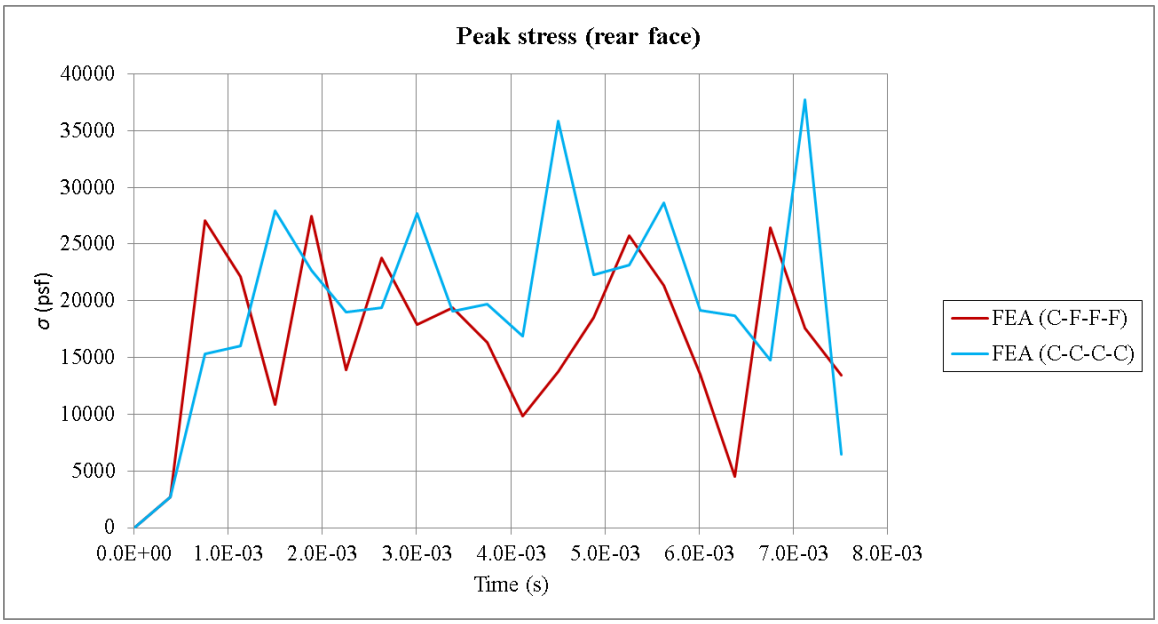
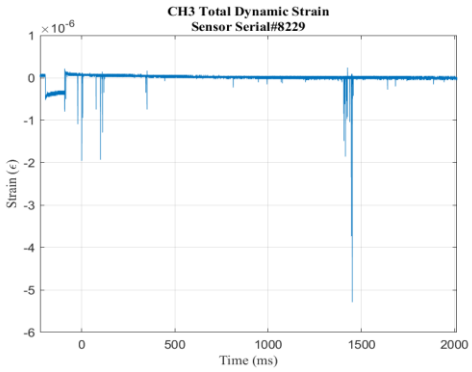


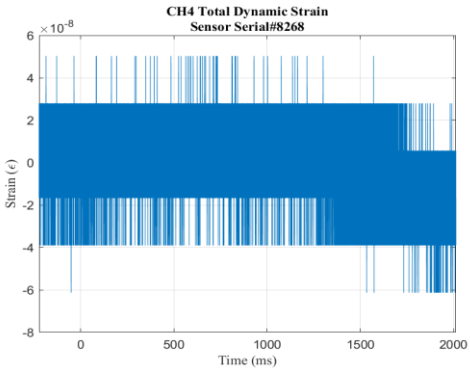
Figure A.62. FEA peak rear face stress, 5 x 5 lumber, test date: 13 Aug 19

F, Roof bolts, 13 August 2019

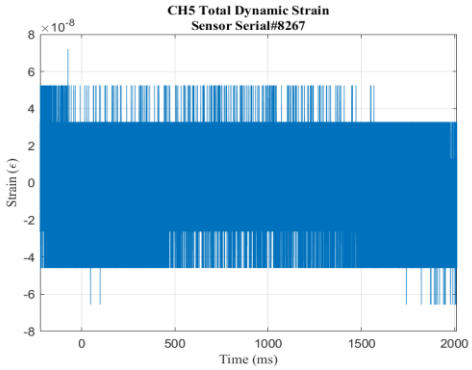
Figure A.63. Roof bolts



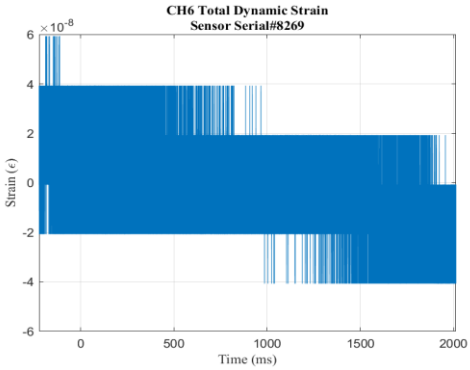
a.



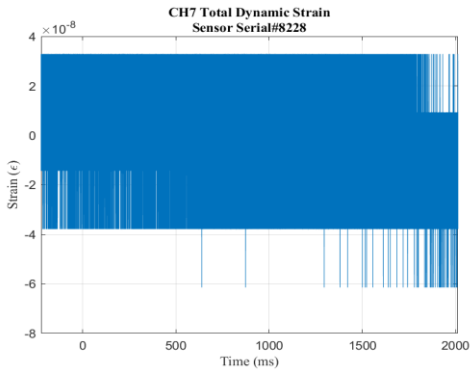
b.



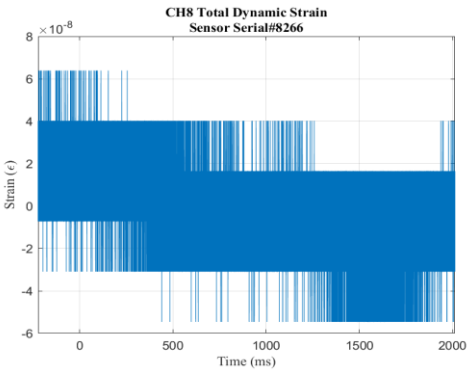
c.



d.



e.



f.

Figure A.64. Strain response, roof bolts, 13 Aug 19

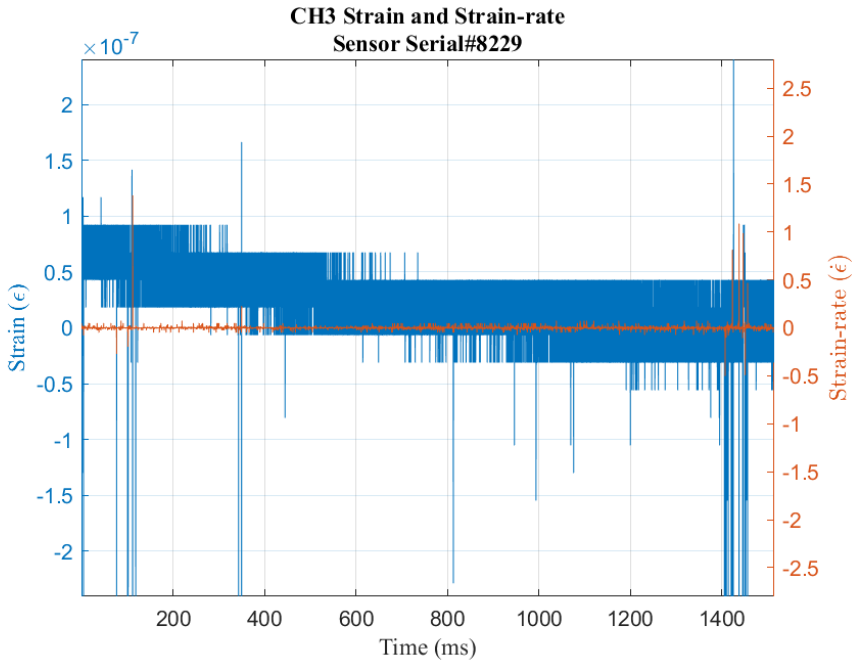


Figure A.65. Strain and strain rate, roof bolts, CH3

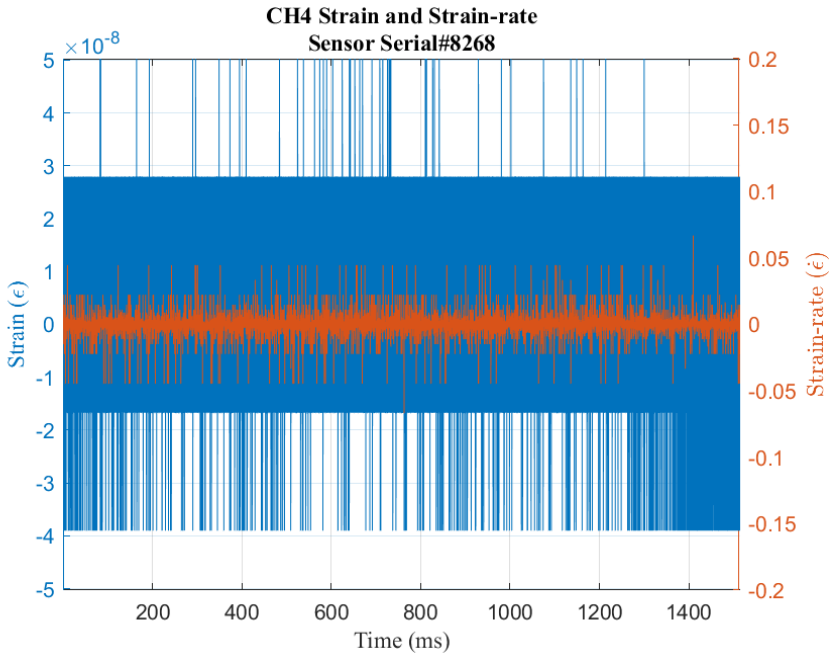


Figure A.66. Strain and strain rate, roof bolts, CH4

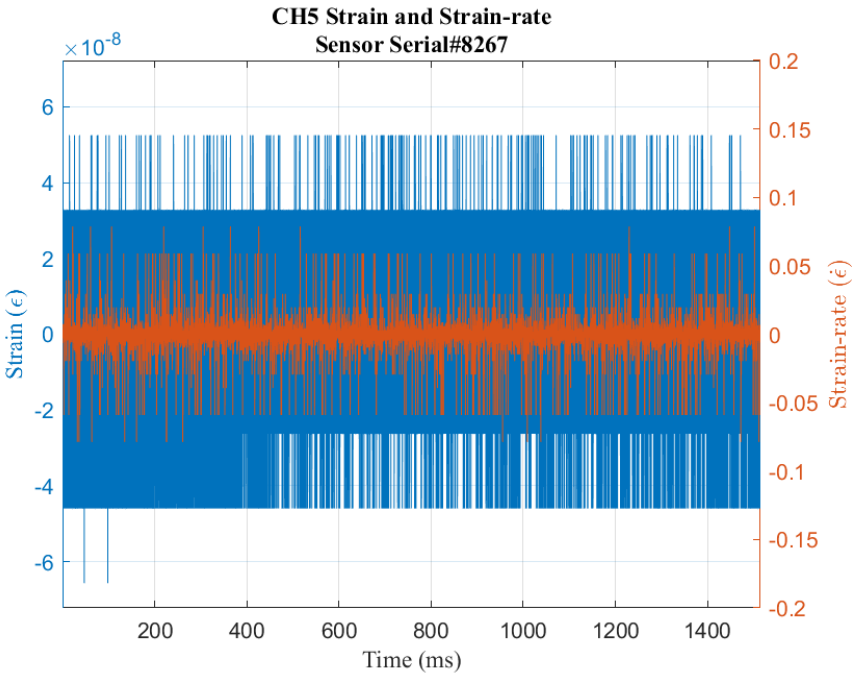


Figure A.67. Strain and strain rate, roof bolts, CH5

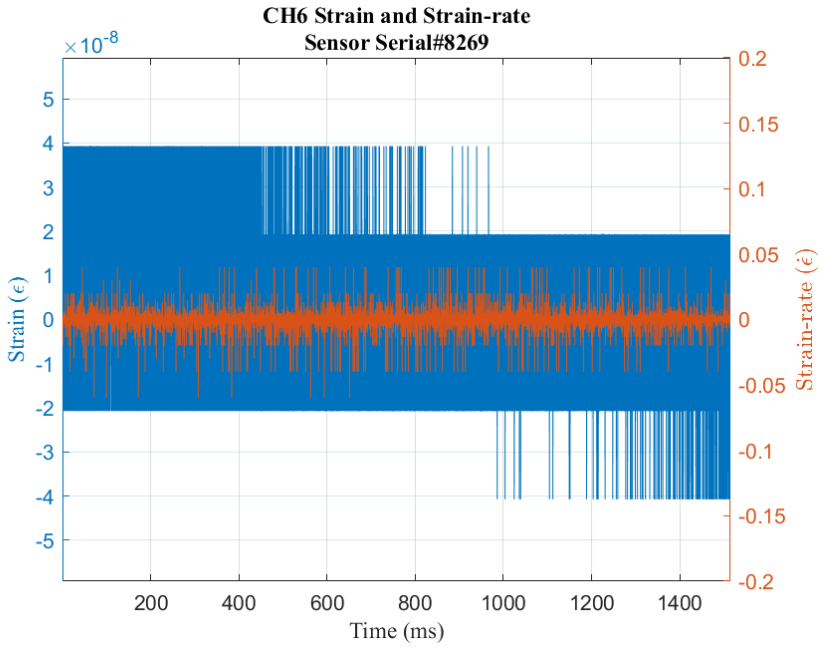


Figure A.68. Strain and strain rate, roof bolts, CH6

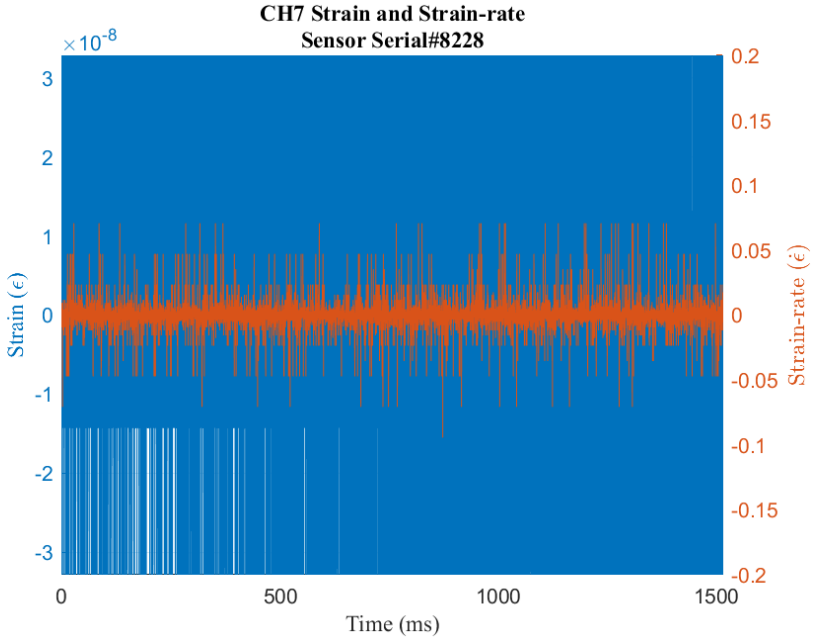


Figure A.69. Strain and strain rate, roof bolts, CH7

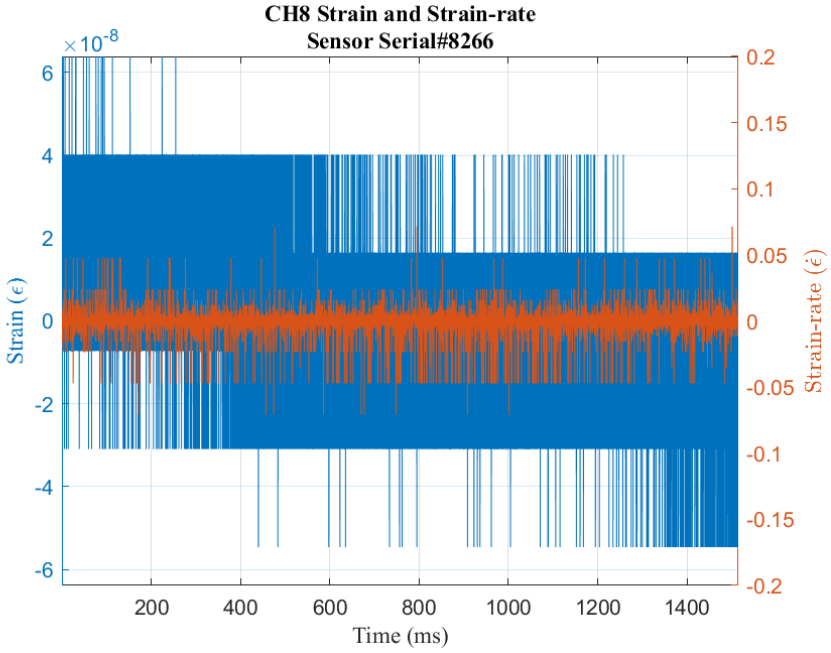
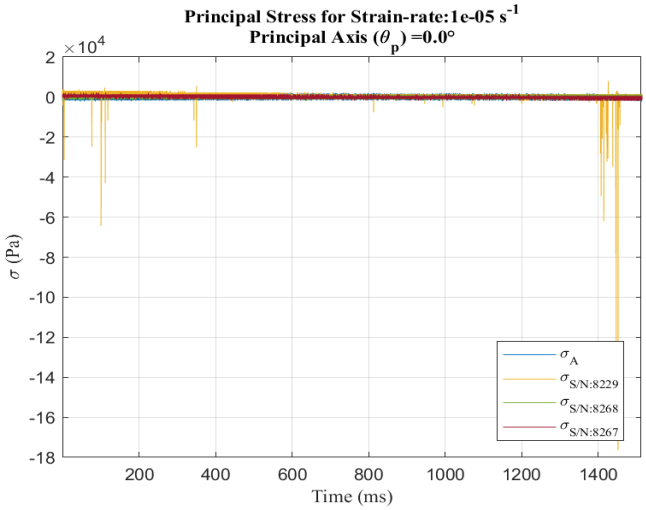
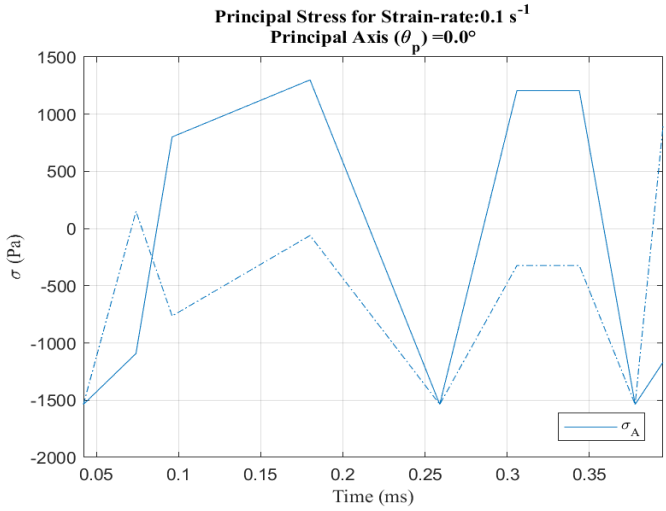


Figure A.70. Strain and strain rate, roof bolts, CH8

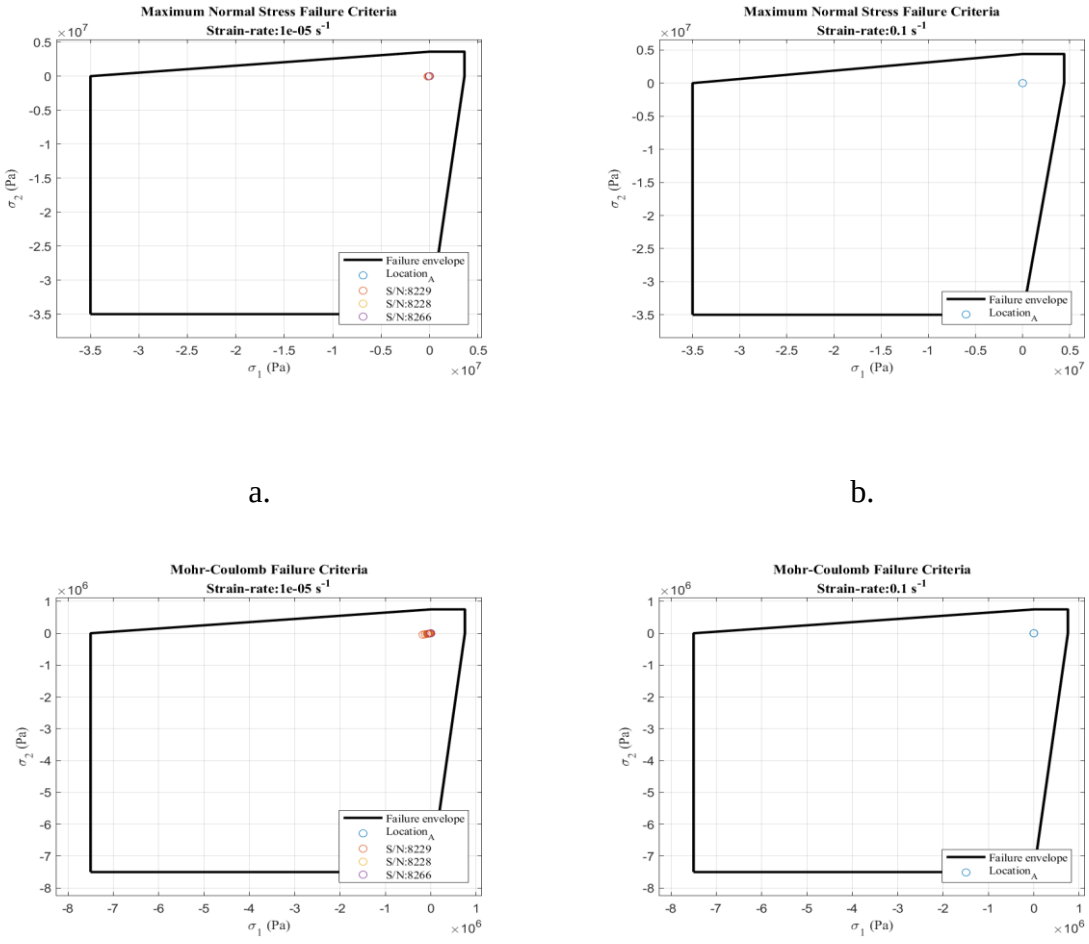


a.



b.

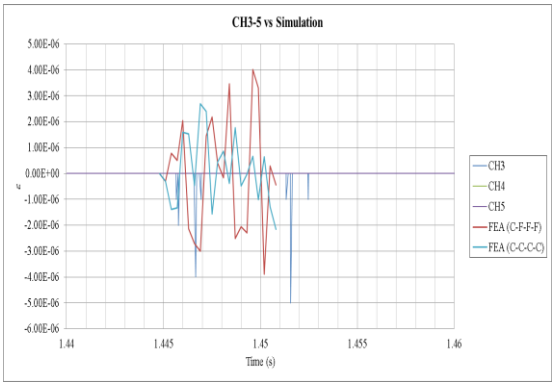
Figure A.71. Principal stress at strain rate, roof bolts, test date: 13 Aug 19



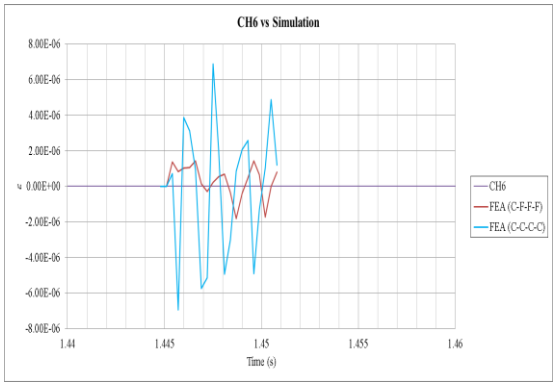
a. b.

c. d.

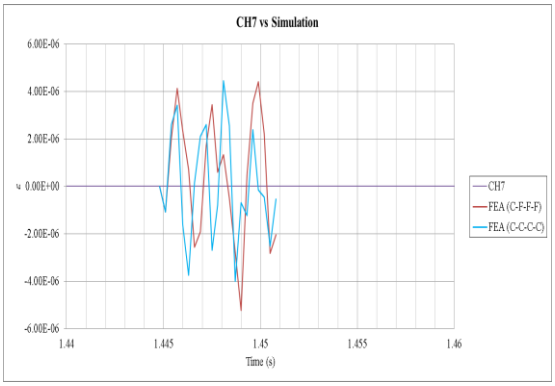
Figure A.72. Failure criterion envelopes, roof bolts, test date: 13 Aug 19



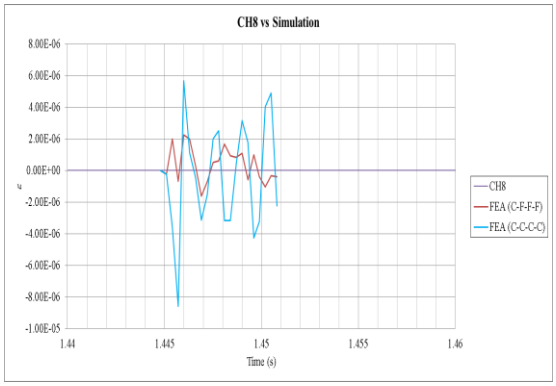
a.



b.



c.



d.

Figure A.73. Strain response and FEA comparison, roof bolts, test date: 13 Aug 19

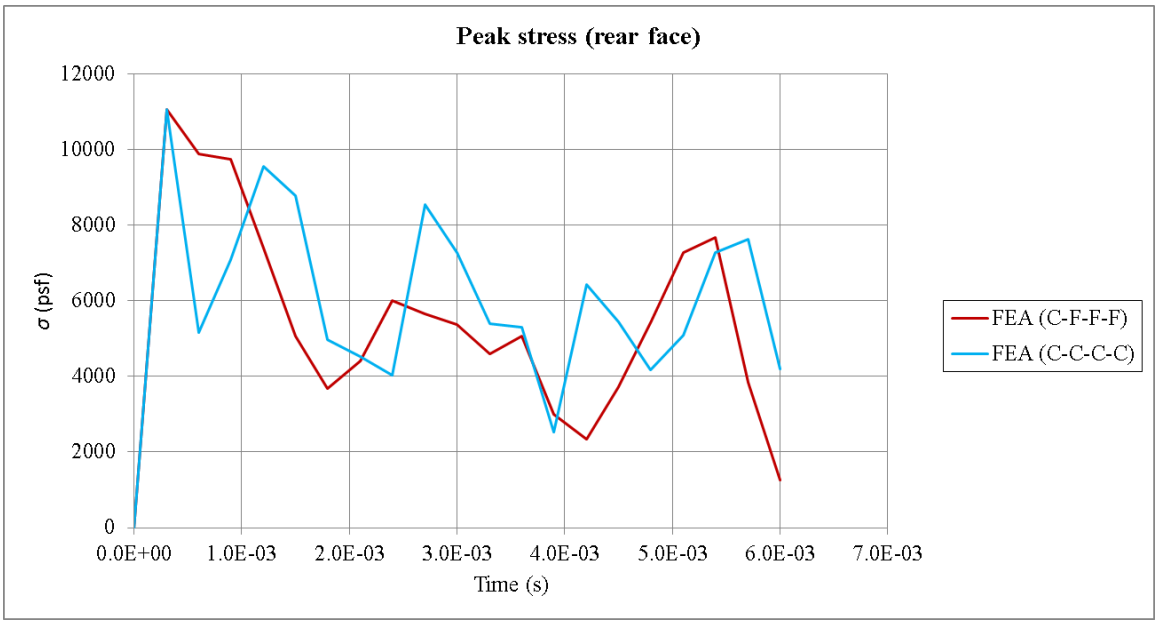
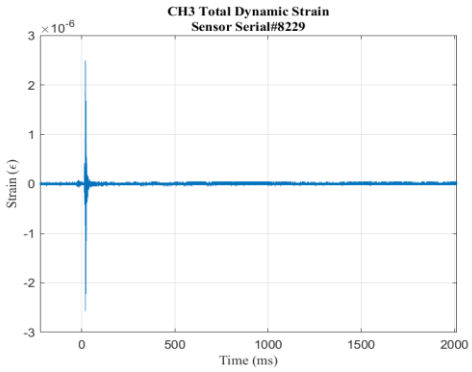


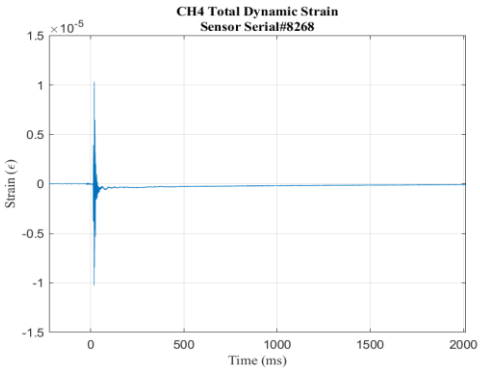
Figure A.74. FEA peak rear face stress, roof bolts, test date: 13 Aug 19

G, Concrete (10 lb), 13 August 2019

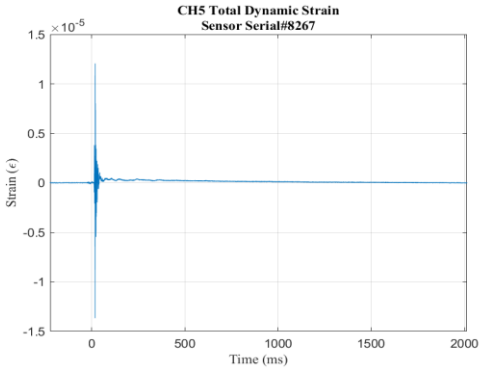
Figure A.75. Concrete (10 lb)



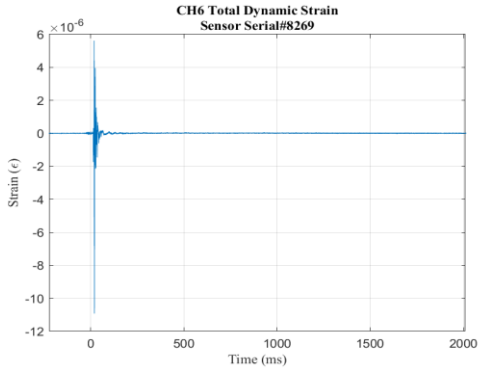
a.



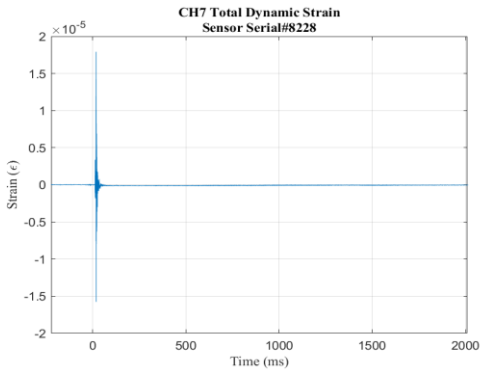
b.



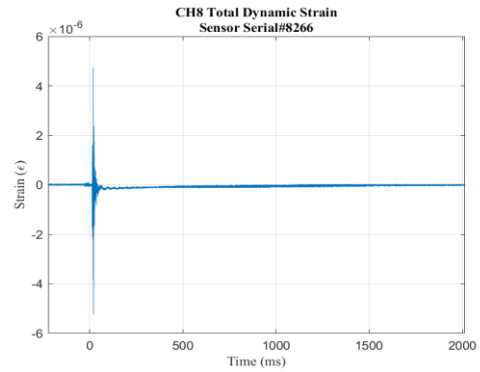
c.



d.



e.



f.

Figure A.76. Strain response, concrete (10 lb) test date: 13 Aug 19

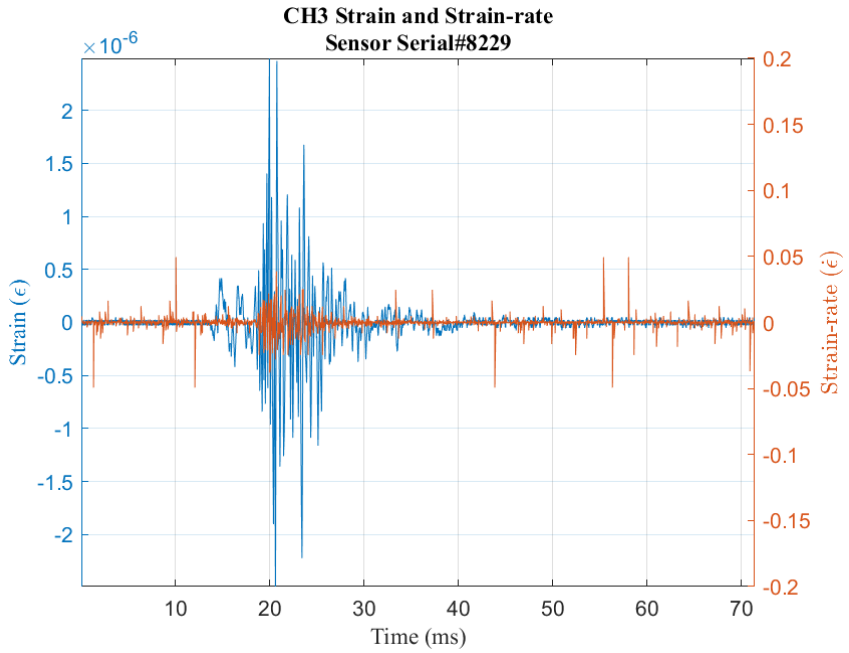


Figure A.77. Strain and strain rate, concrete (10 lb), CH3

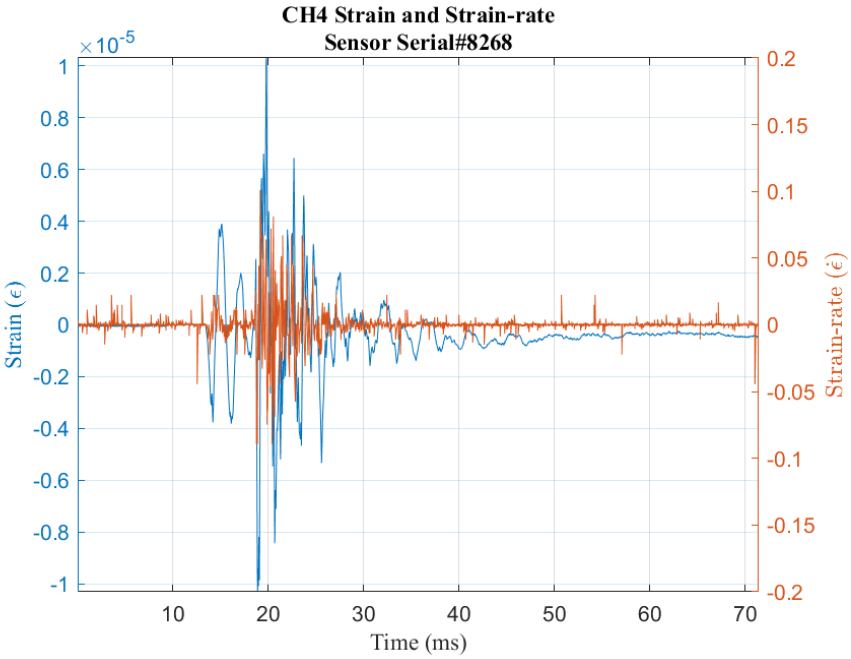


Figure A.78. Strain and strain rate, concrete (10 lb), CH4

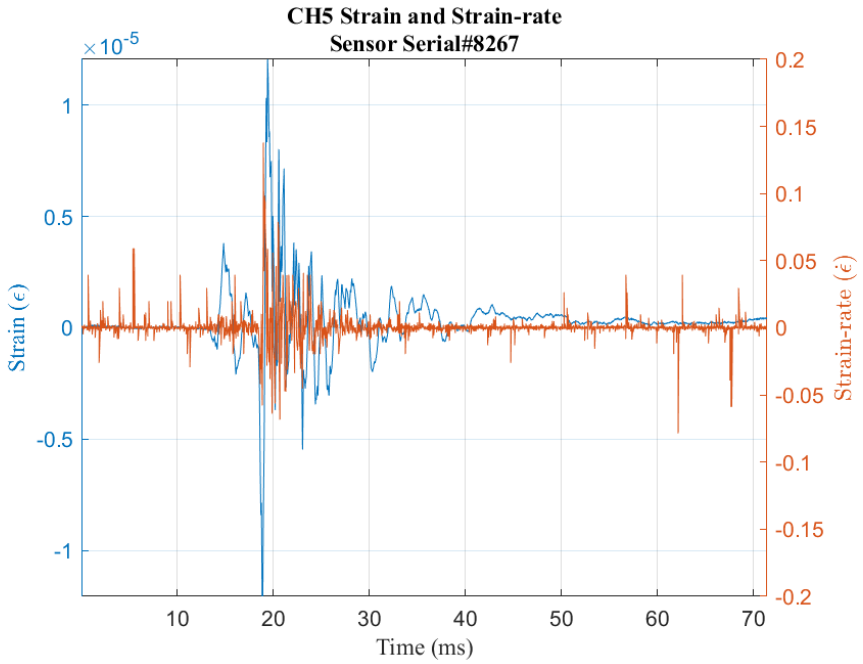


Figure A.79. Strain and strain rate, concrete (10 lb), CH5

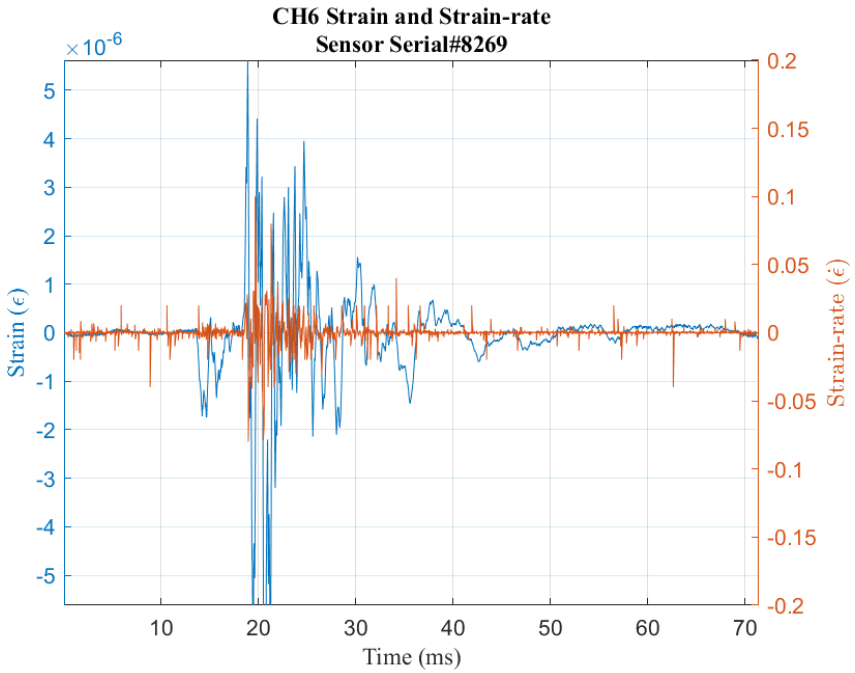


Figure A.80. Strain and strain rate, concrete (10 lb), CH6

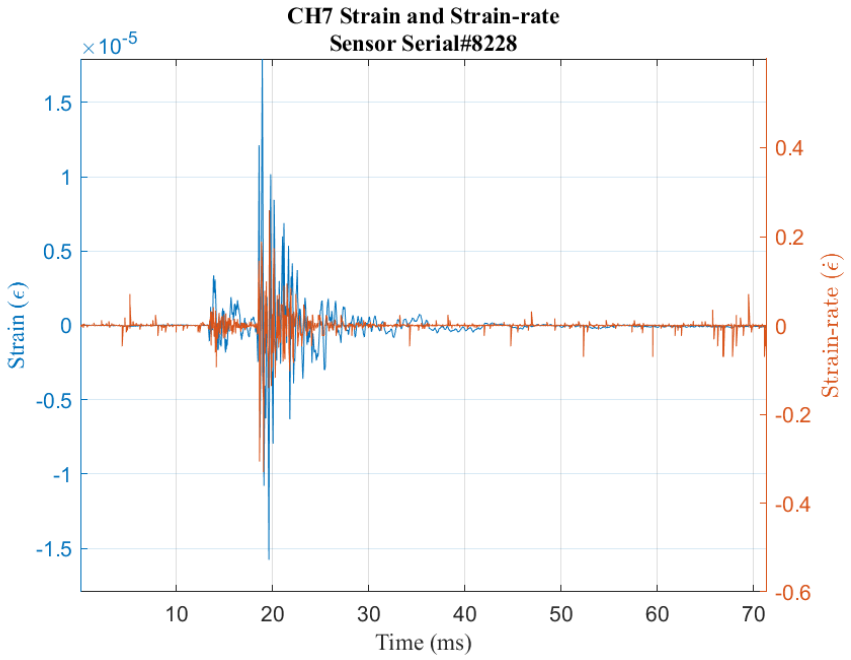


Figure A.81. Strain and strain rate, concrete (10 lb), CH7

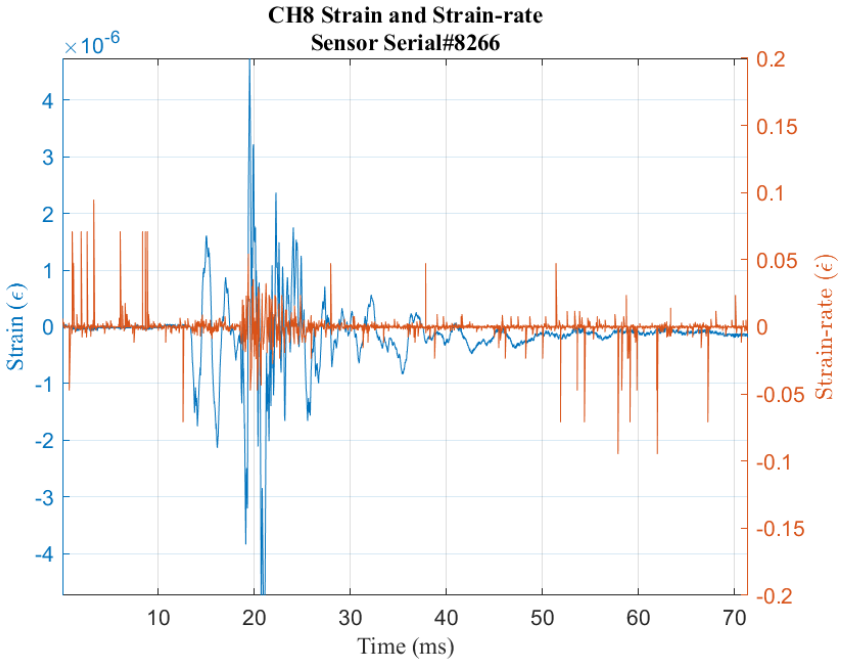
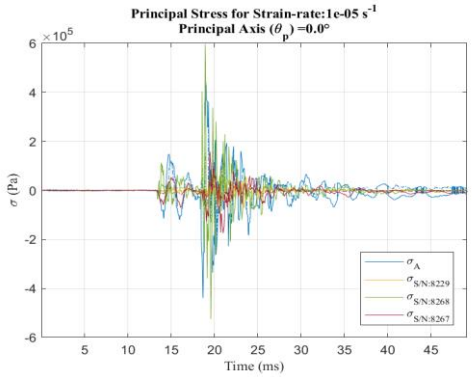
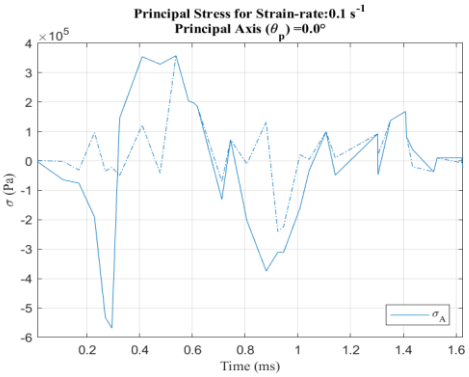


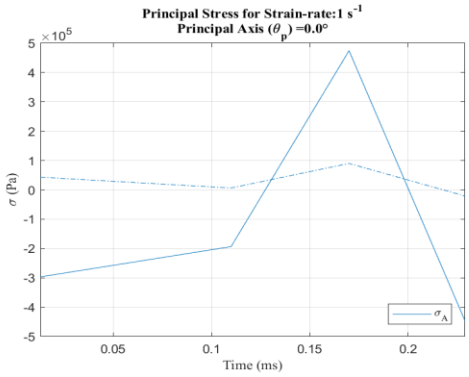
Figure A.82. Strain and strain rate, concrete (10 lb), CH8



a.

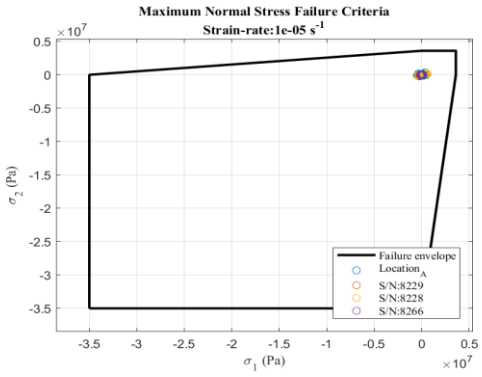


b.

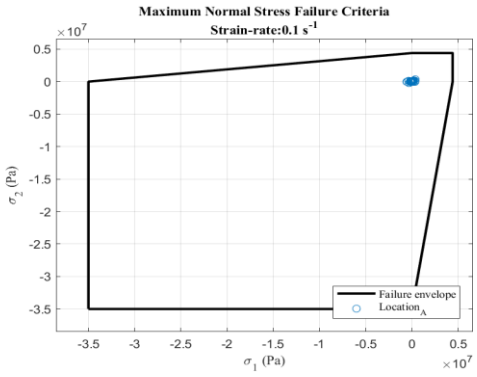


c.

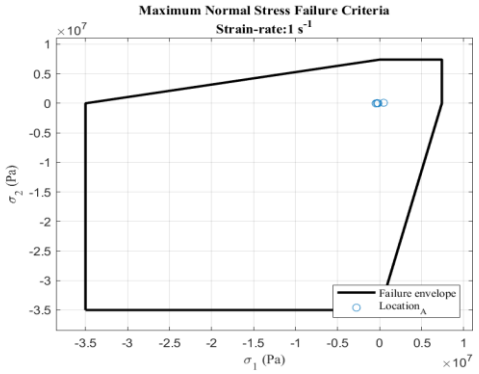
Figure A.83. Principal stress at strain rate, concrete (10 lb), test date: 13 Aug 19



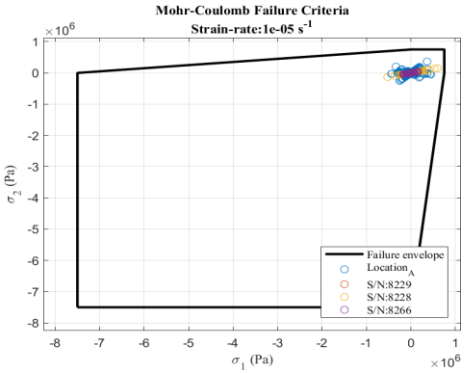
a.



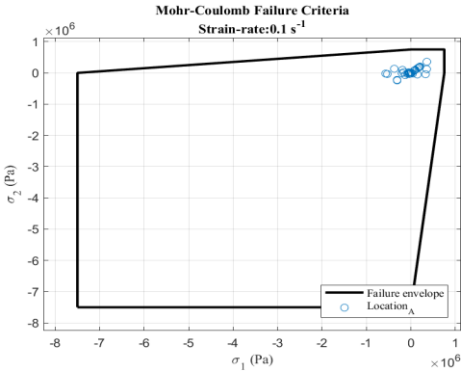
b.



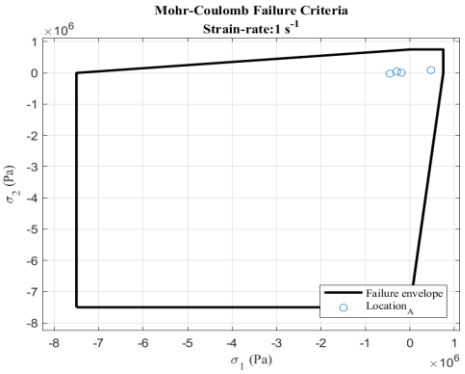
c.



d.

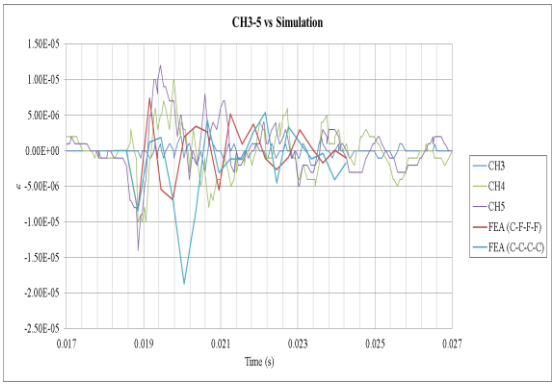


e.

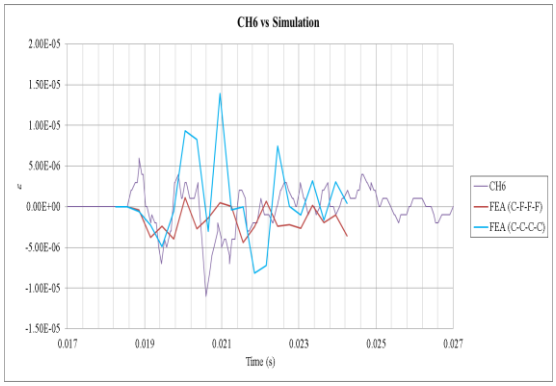


f.

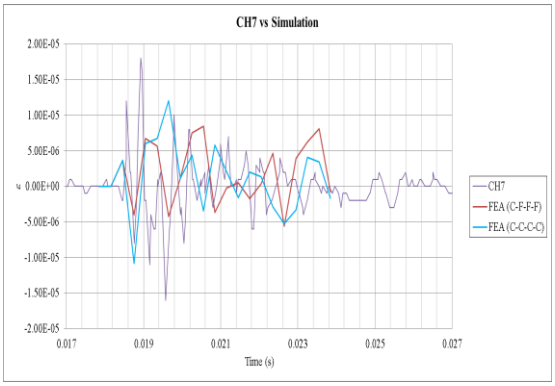
Figure A.84. Failure criterion envelopes, concrete (10 lb), test date: 13 Aug 19



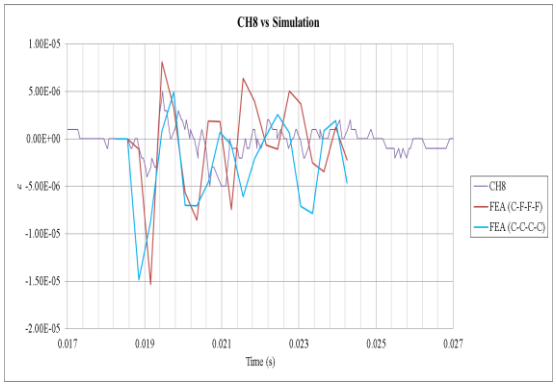
a.



b.



c.



d.

Figure A.85. Strain response and FEA comparison, concrete (10 lb), test date: 13 Aug 19

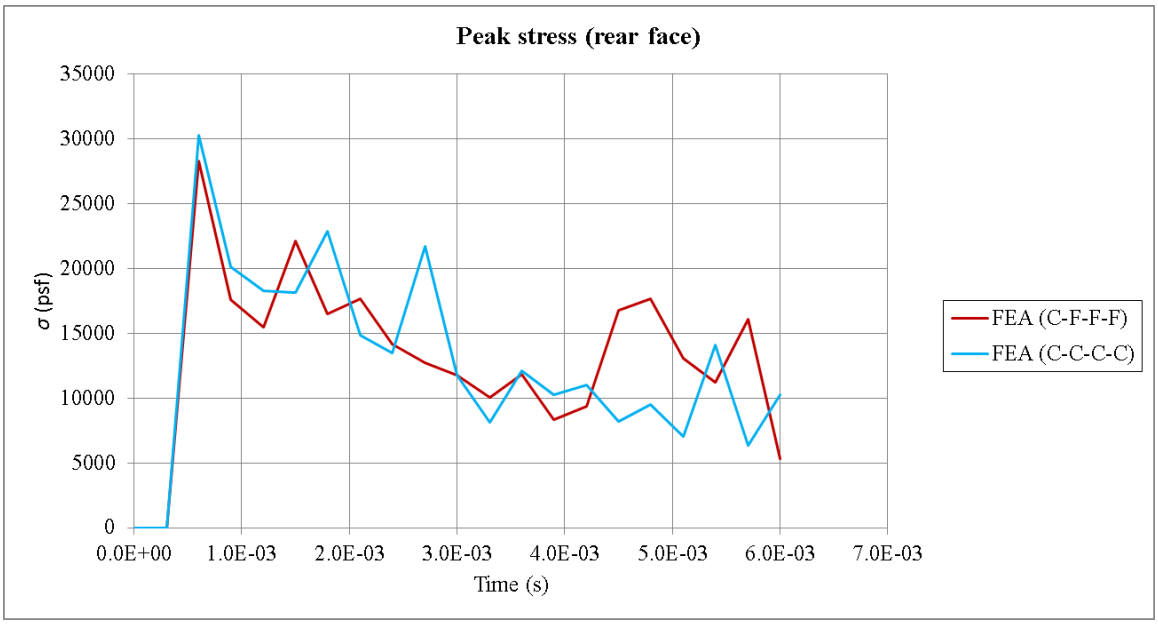
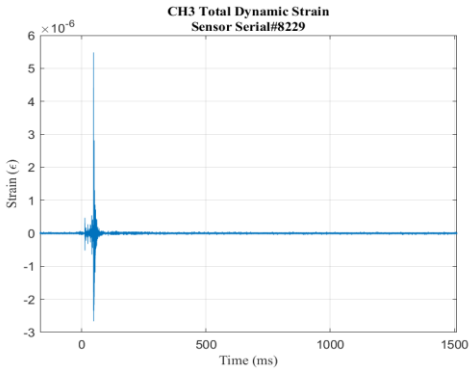


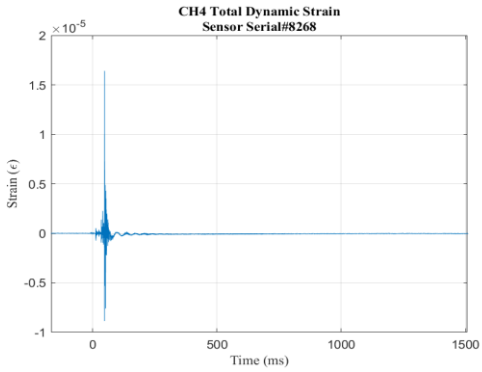
Figure A.86. FEA peak rear face stress, concrete (10 lb), test date: 13 Aug 19

H, Concrete (20 lb), 27 August 2019

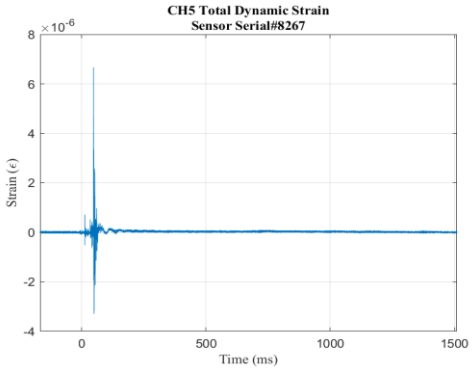
Figure A.87. Concrete (20 lb)



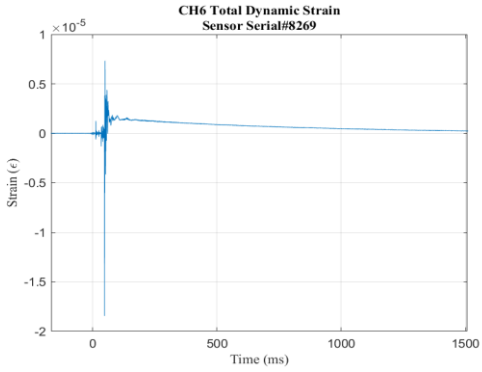
a.



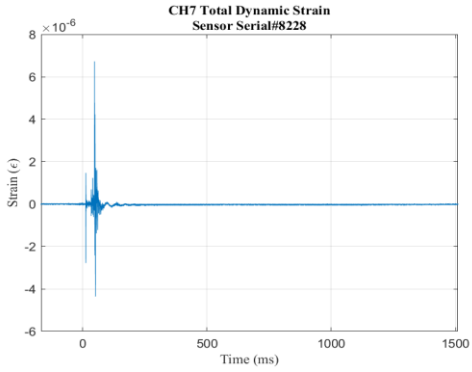
b.



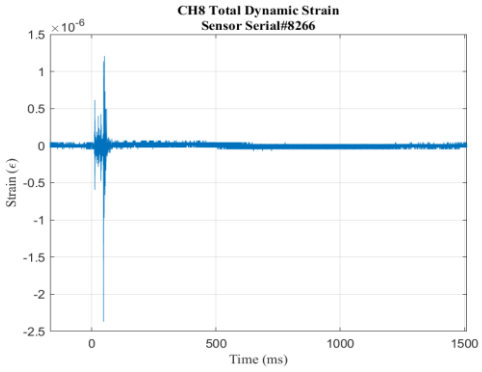
c.



d.



e.



f.

Figure A.88. Strain response, concrete (20 lb), test date: 27 Aug 19

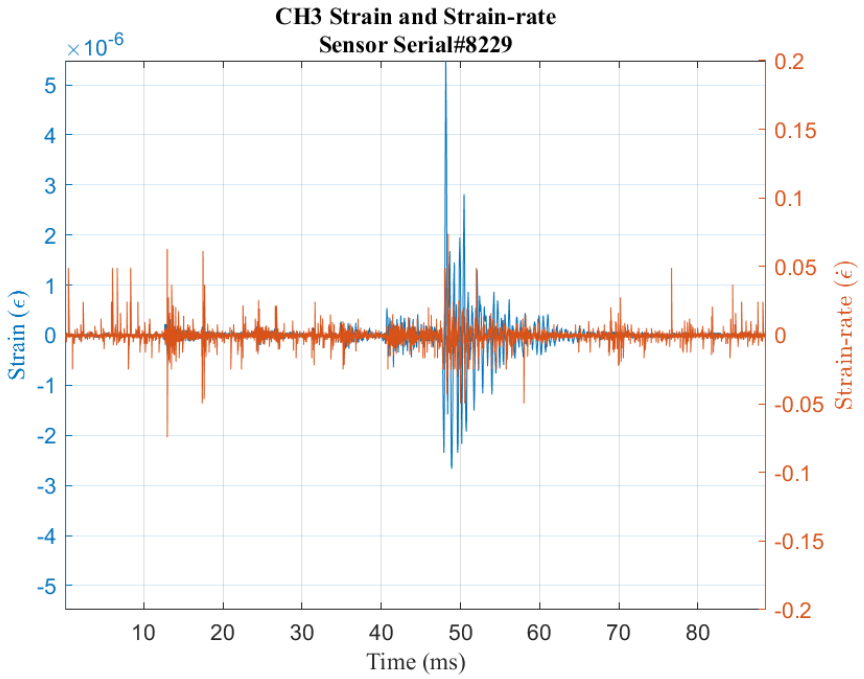


Figure A.89. Strain and strain rate, concrete (20 lb), CH3

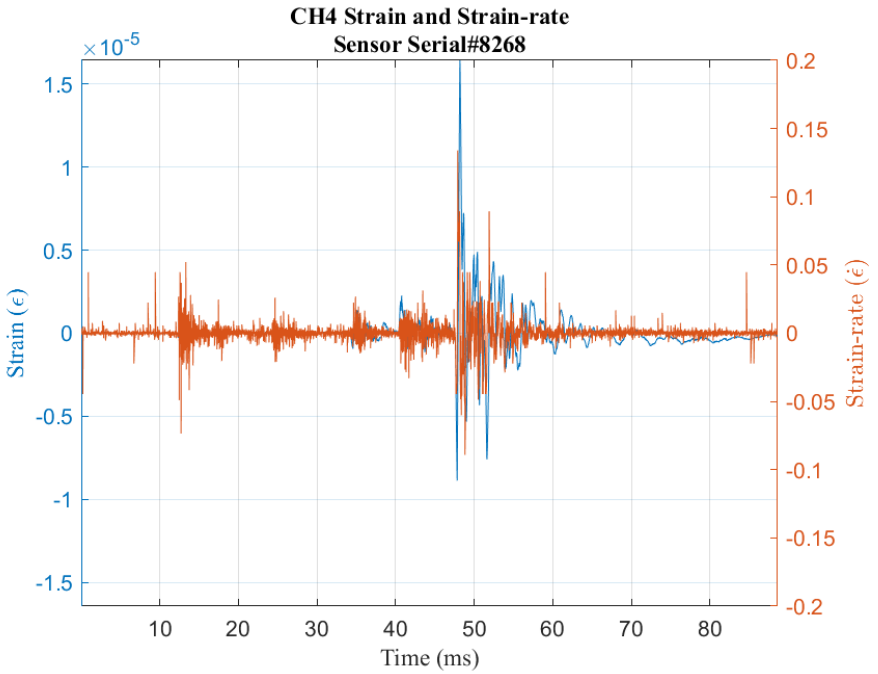


Figure A.90. Strain and strain rate, concrete (20 lb), CH4

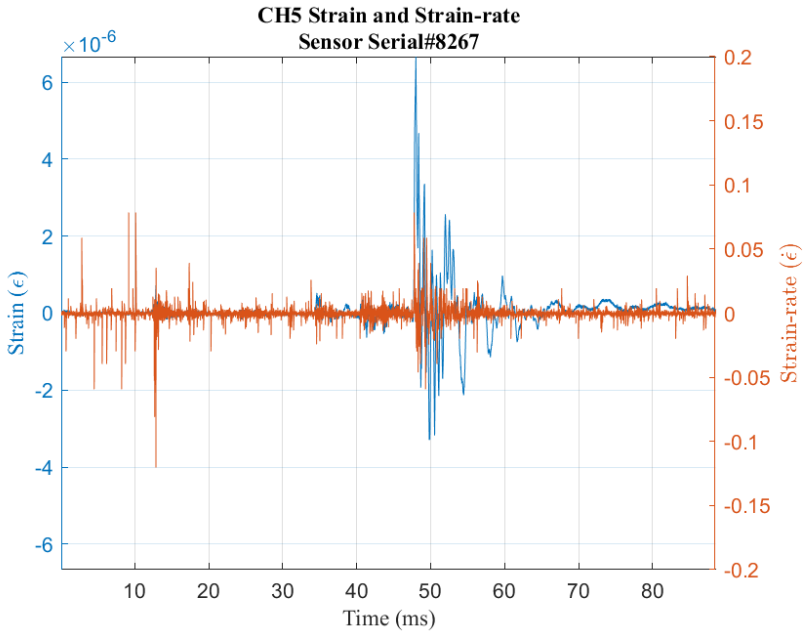


Figure A.91. Strain and strain rate, concrete (20 lb), CH5

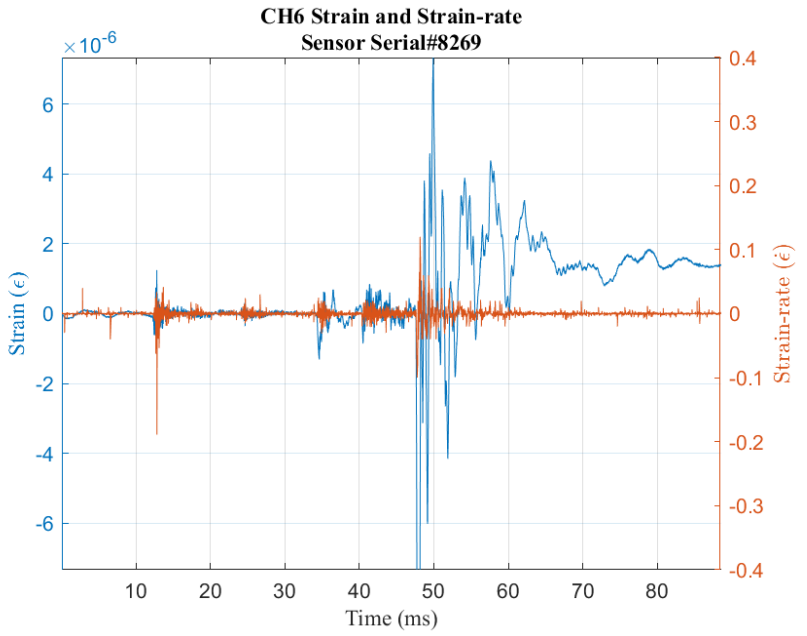


Figure A.92. Strain and strain rate, concrete (20 lb), CH6

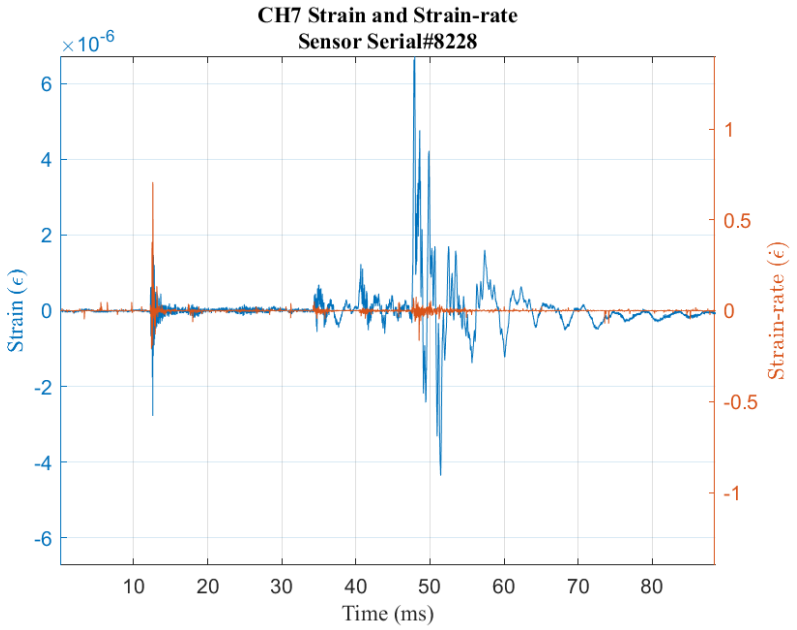


Figure A.93. Strain and strain rate, concrete (20 lb), CH7

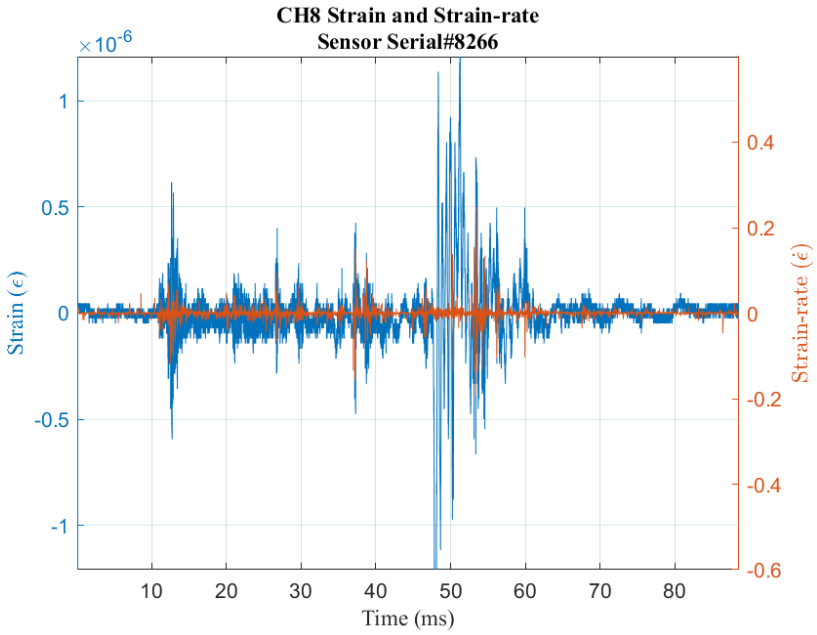
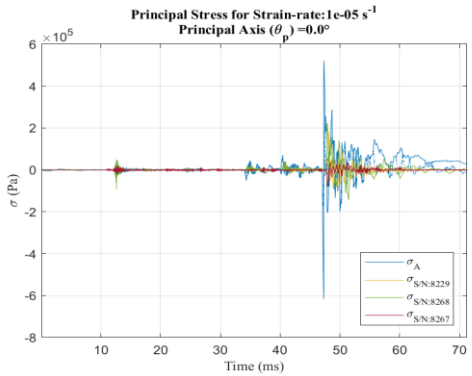
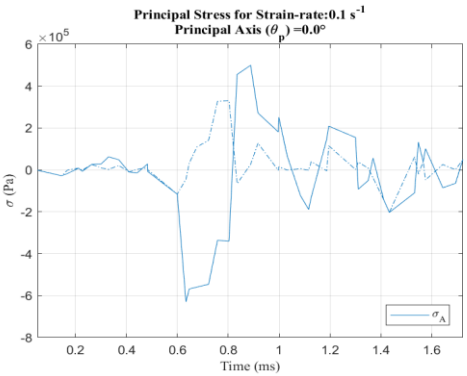


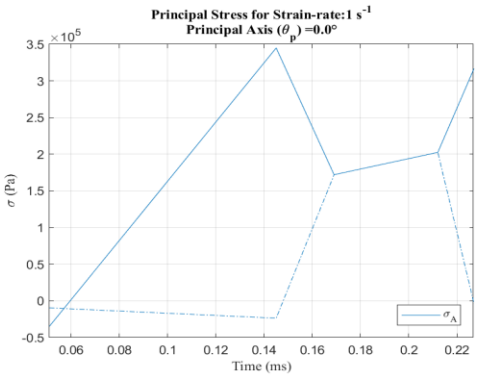
Figure A.94. Strain and strain rate, concrete (20 lb), CH8



a.

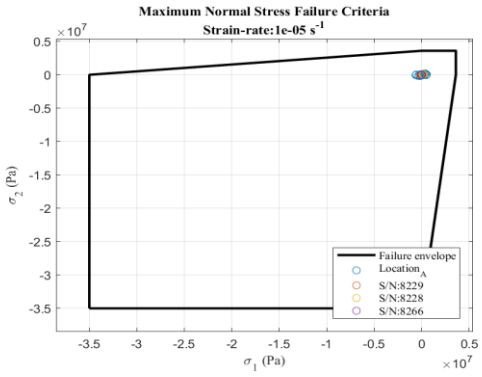


b.

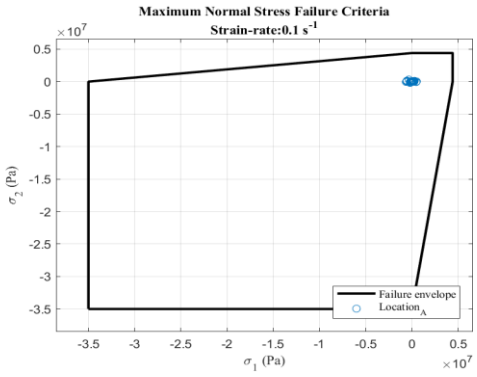


c.

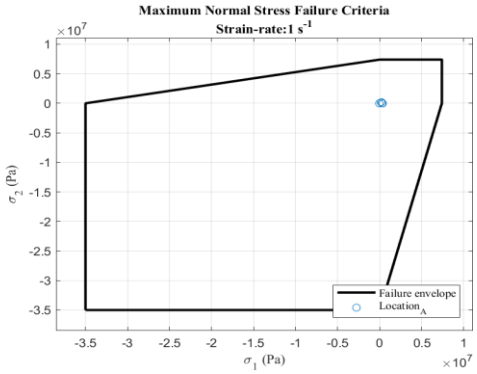
Figure A.95. Principal stress at strain rate, concrete (20 lb), test date: 27 Aug 19



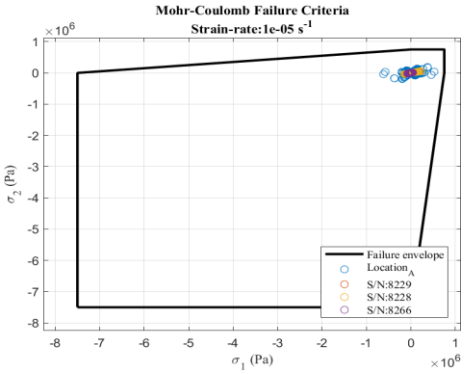
a.



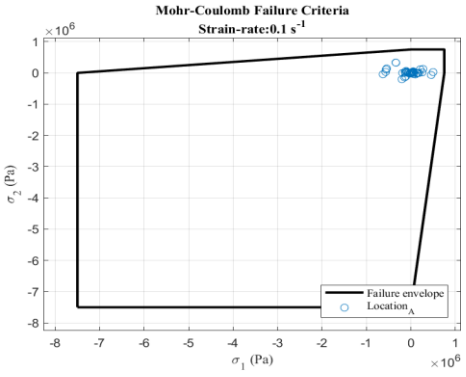
b.



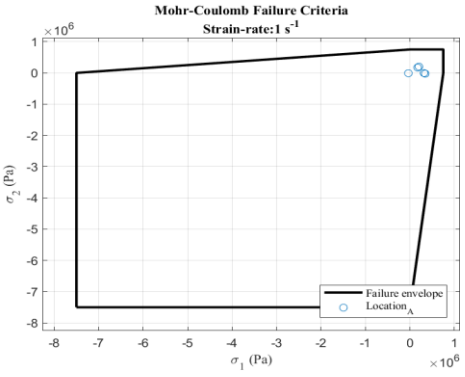
c.



d.

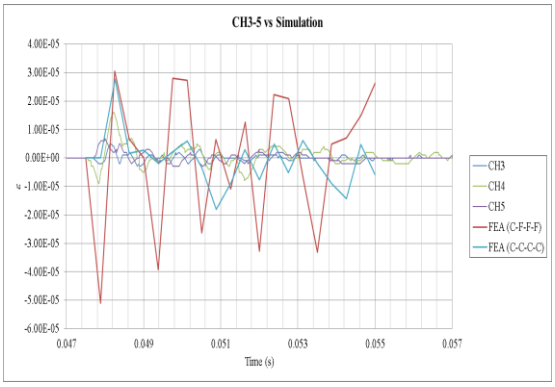


e.

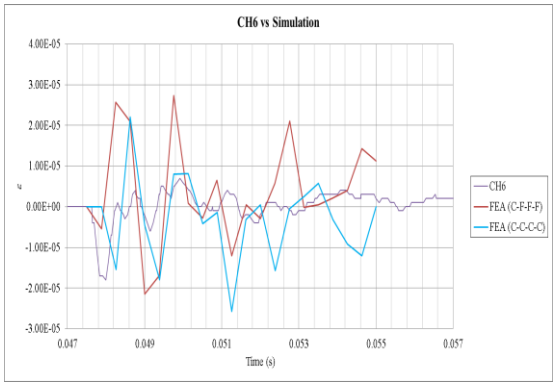


f.

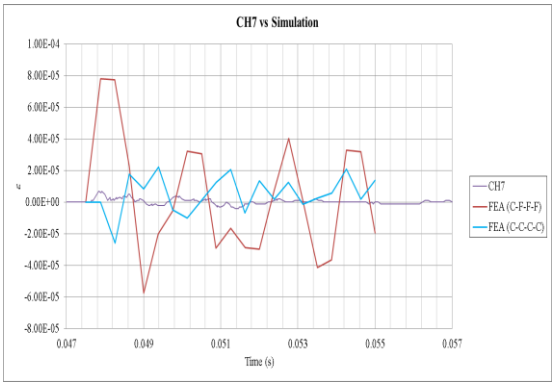
Figure A.96. Failure criterion envelopes, concrete (20 lb), test date: 27 Aug 19



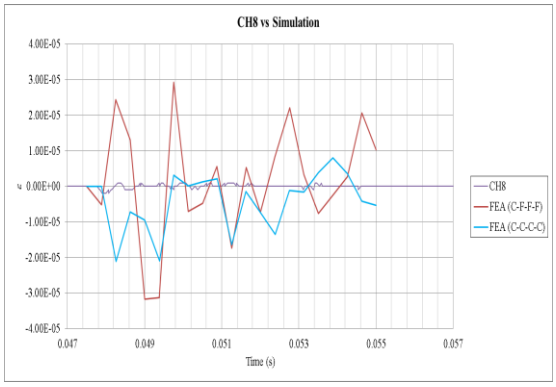
a.



b.



c.



d.

Figure A.97. Strain response and FEA comparison, concrete (20 lb), test date: 27 Aug 19

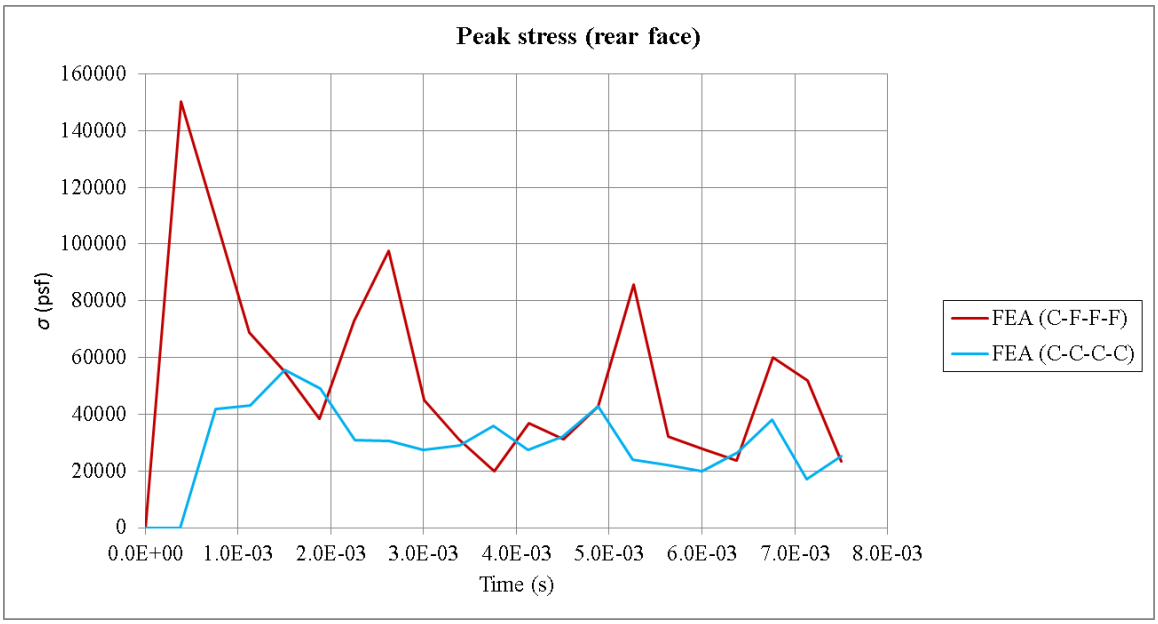
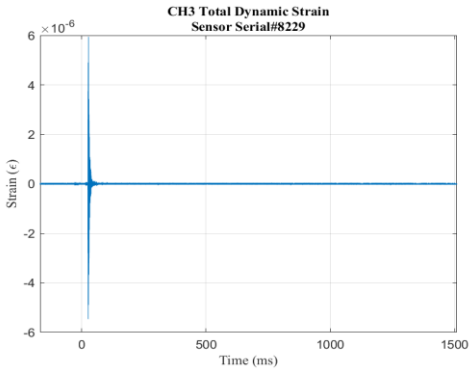


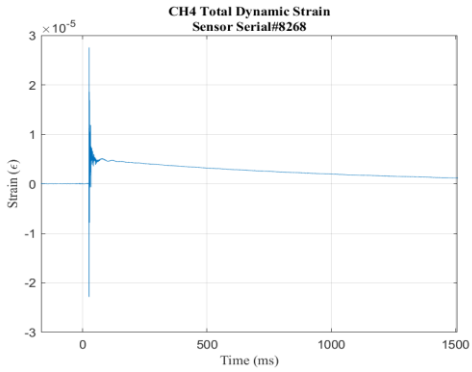
Figure A.98. FEA peak rear face stress, concrete (20 lb), test date: 27 Aug 19

I, Concrete (30 lb), 27 August 2019

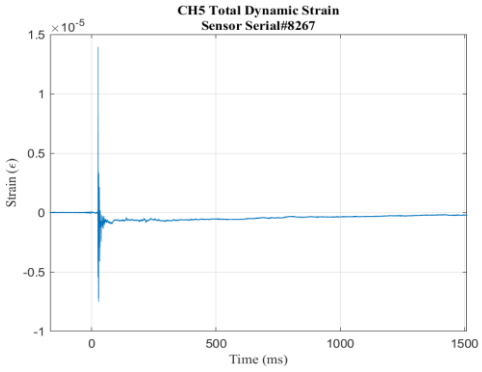
Figure A.99. Concrete (30 lb)



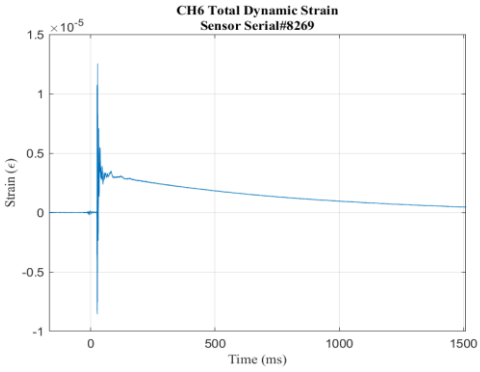
a.



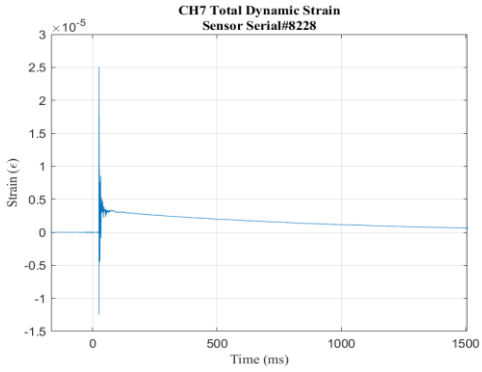
b.



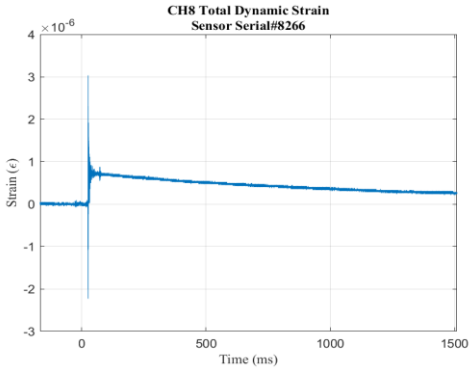
c.



d.



e.



f.

Figure A.100. Strain response, concrete (30 lb), test date: 27 Aug 19

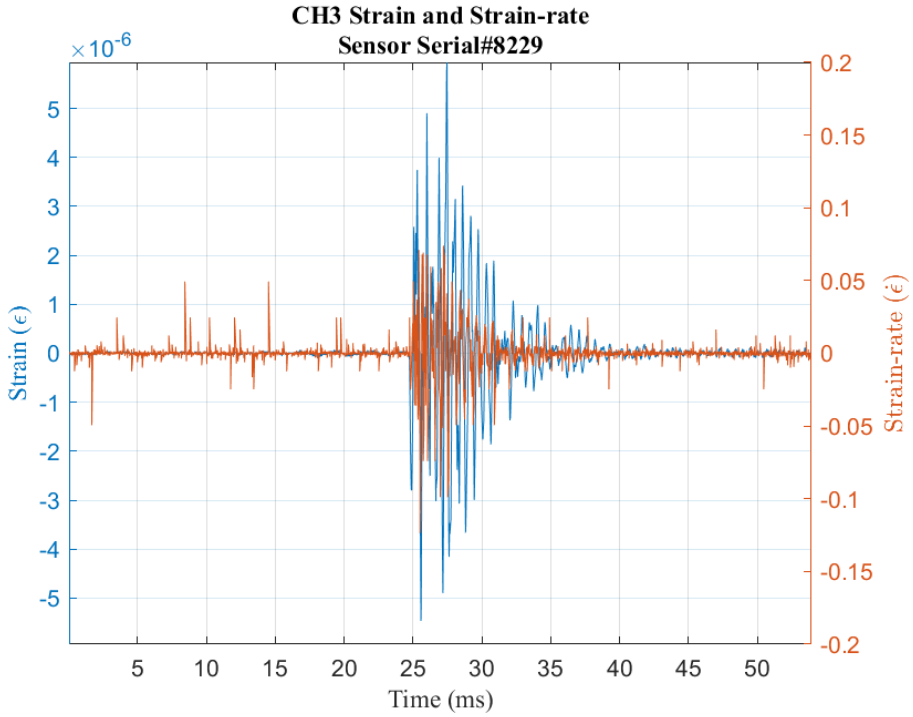


Figure A.101. Strain and strain rate, concrete (30 lb), CH3

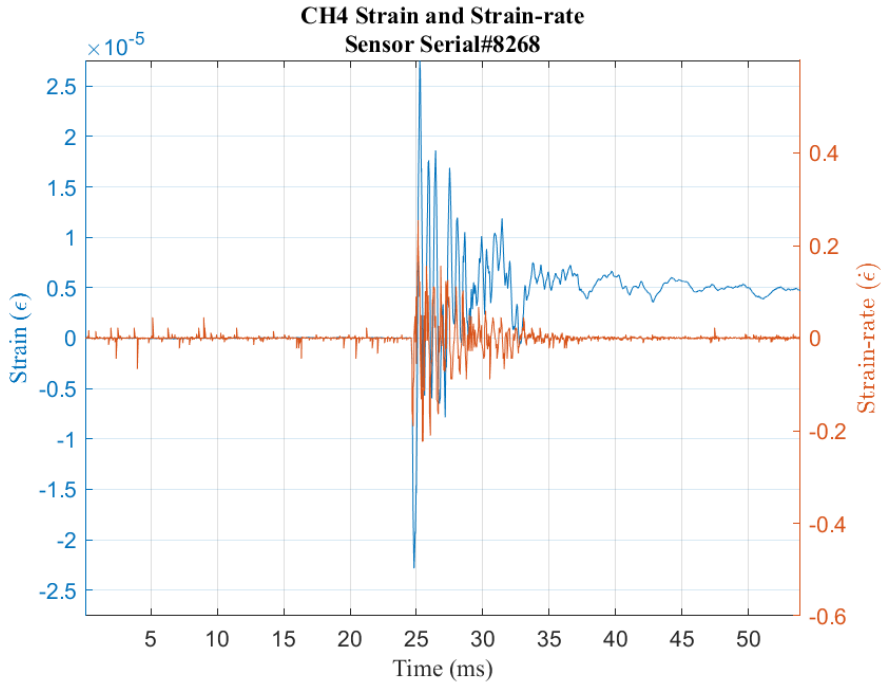


Figure A.102. Strain and strain rate, concrete (30 lb), CH4

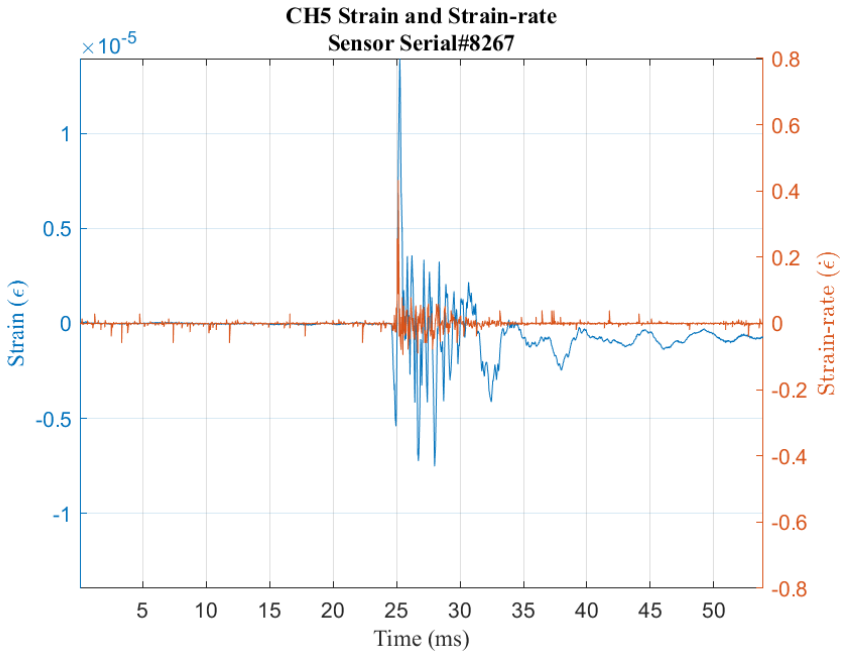


Figure A.103. Strain and strain rate, concrete (30 lb), CH5

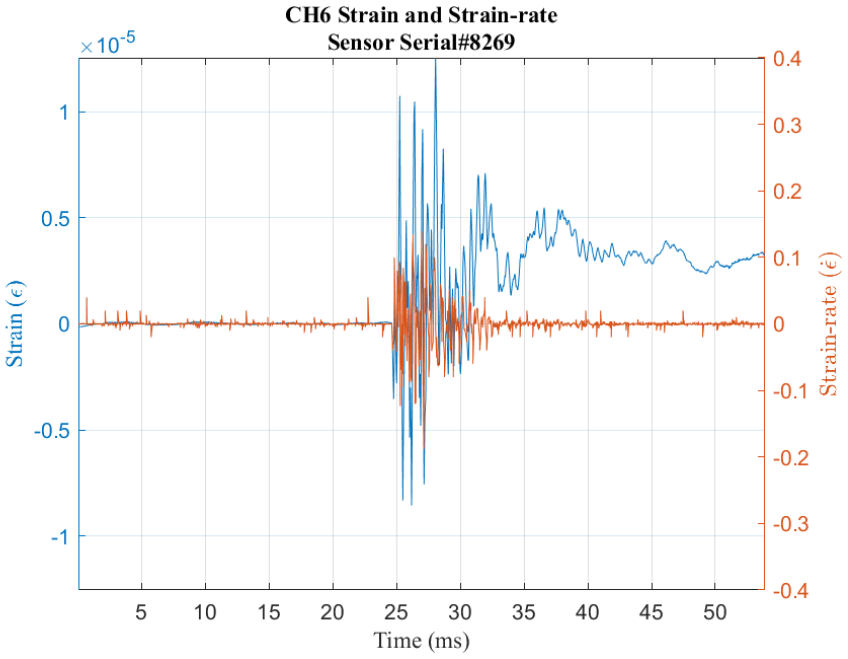


Figure A.104. Strain and strain rate, concrete (30 lb), CH6

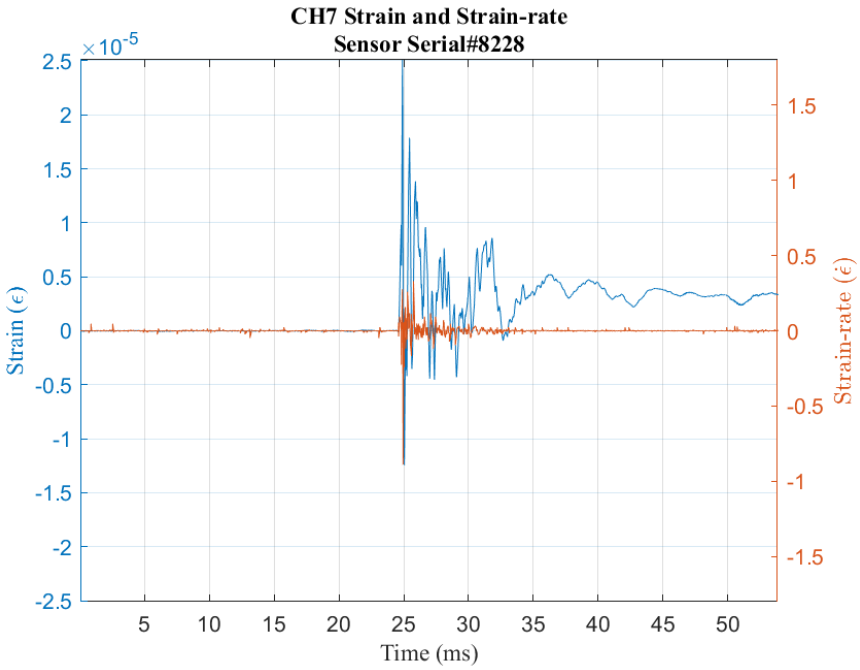


Figure A.105. Strain and strain rate, concrete (30 lb), CH7

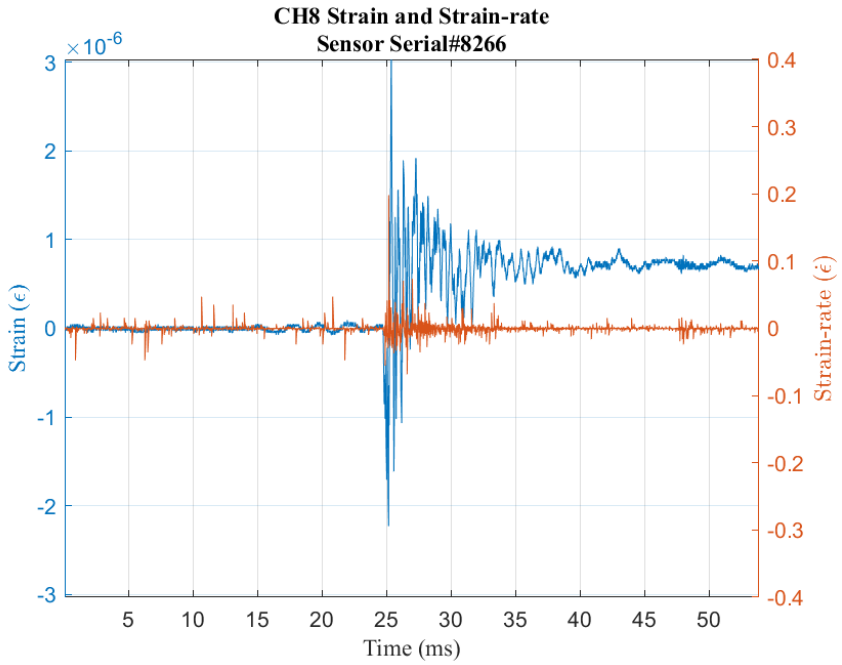
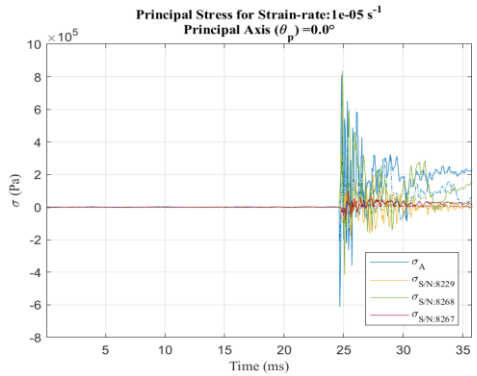
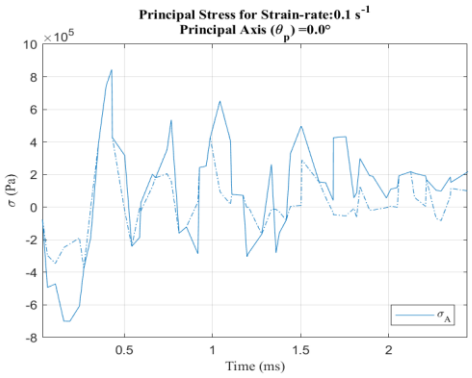


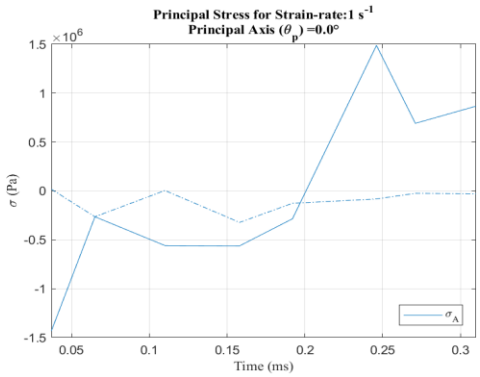
Figure A.106. Strain and strain rate, concrete (30 lb), CH8



a.

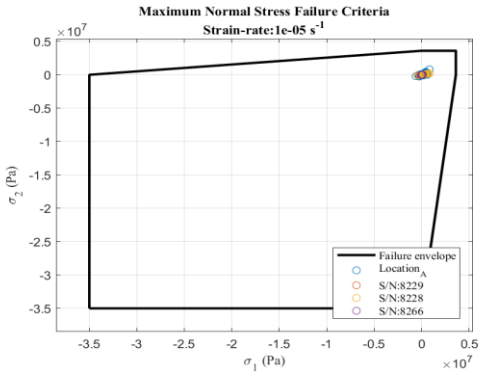


b.

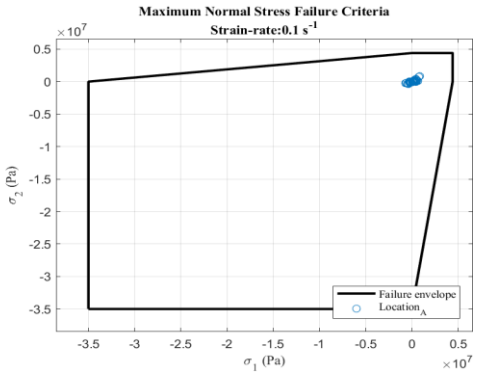


c.

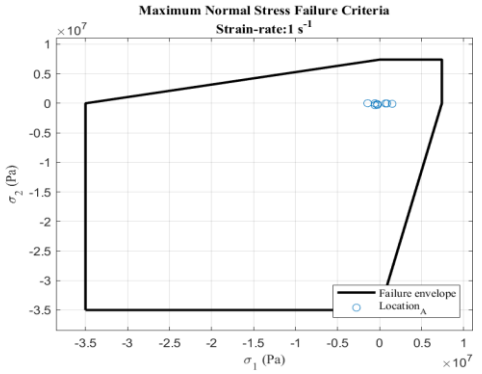
Figure A.107. Principal stress at strain rate, concrete (30 lb), test date: 27 Aug 19



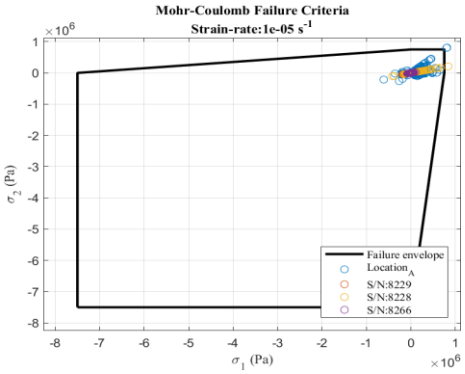
a.



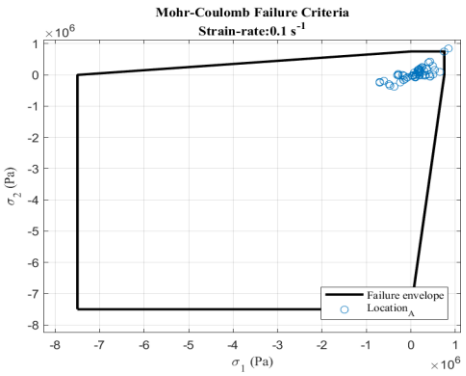
b.



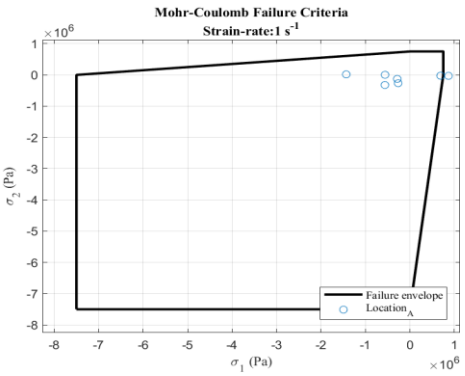
c.



d.

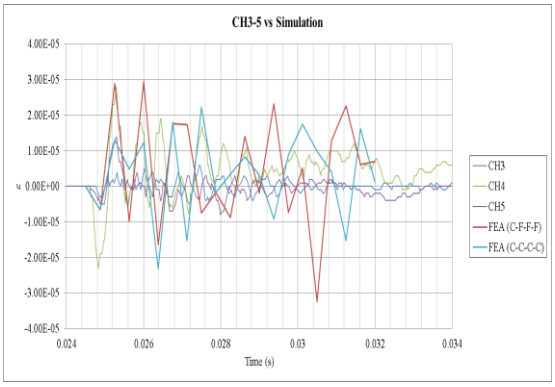


e

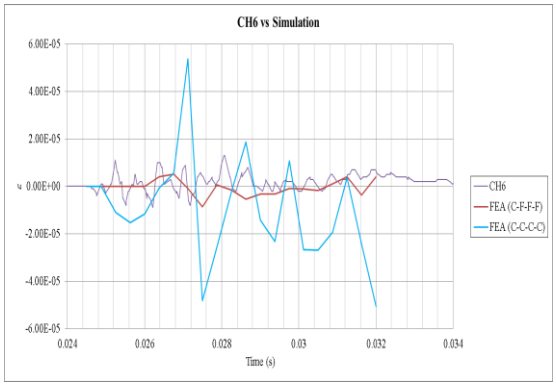


f

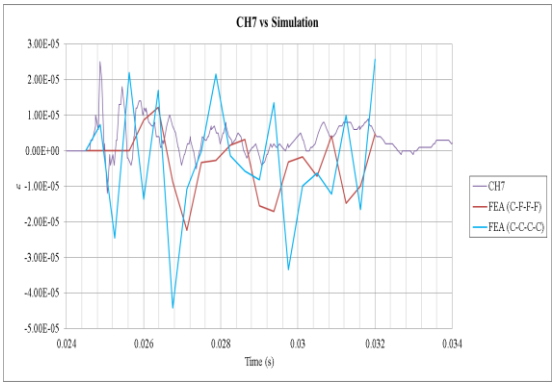
Figure A.108. Failure criterion envelopes, concrete (30 lb), test date: 27 Aug 19



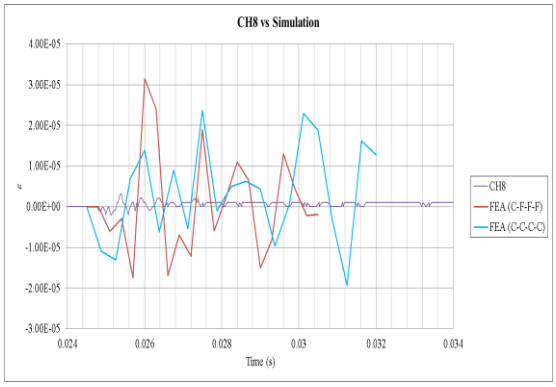
a.



b.



c.



d.

Figure A.109. Strain response and FEA comparison, concrete (30 lb), test date: 27 Aug

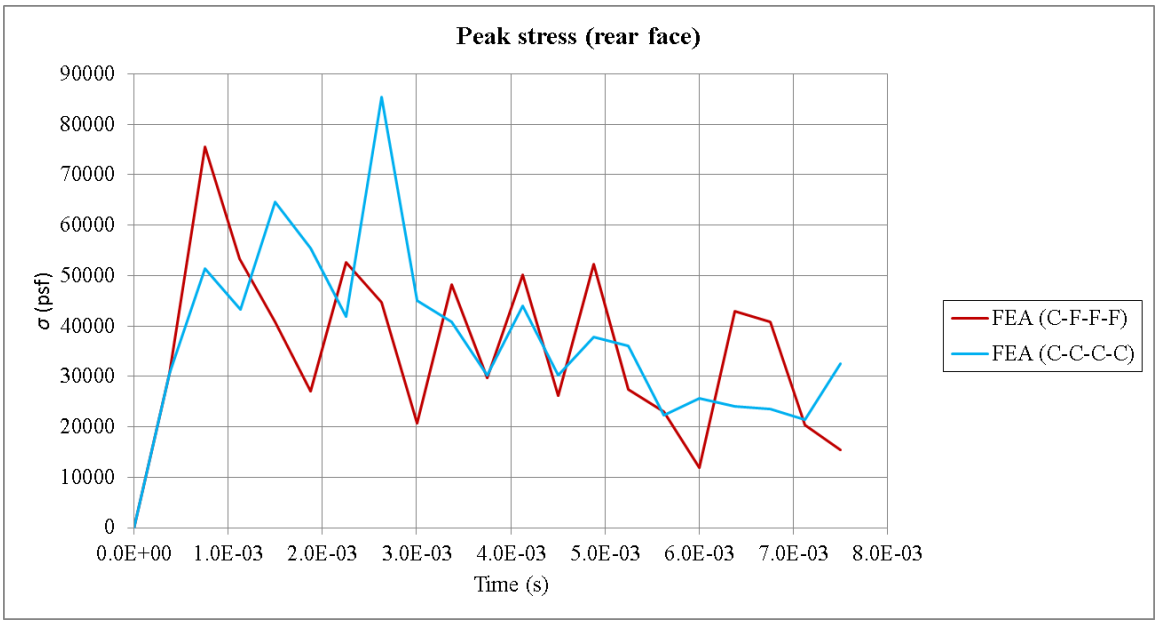
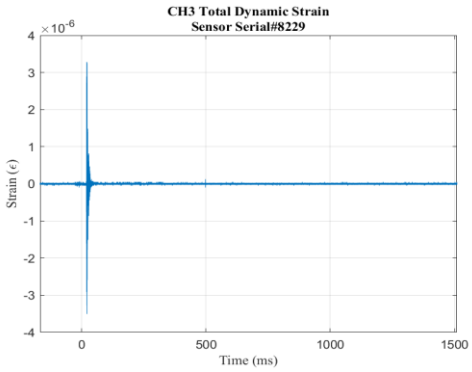


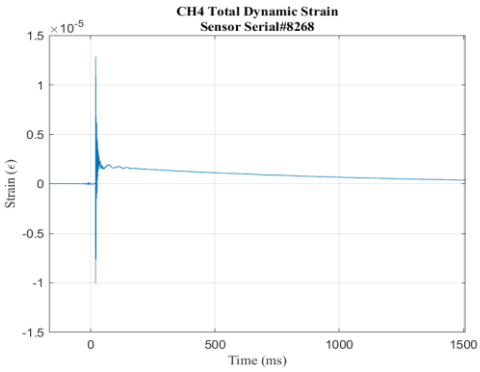
Figure A.110. FEA peak rear face stress, concrete (30 lb), test date: 27 Aug 19

J, Steel penetrator (10 lb), 27 August 2019

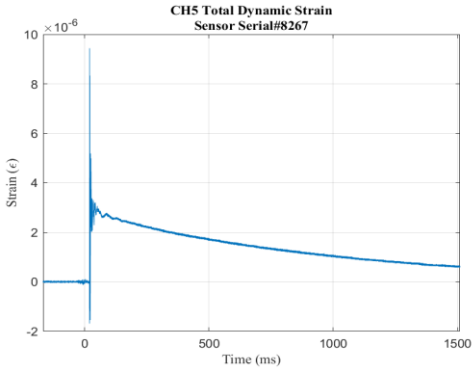
Figure A.111. Steel penetrator (10 lb)



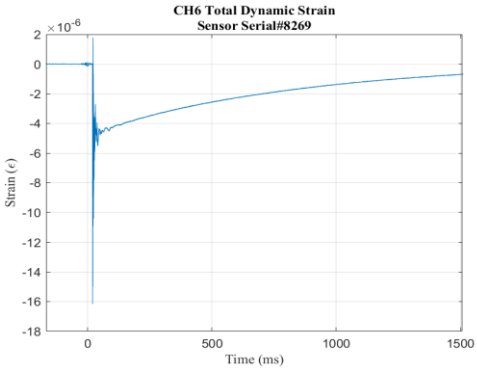
a.



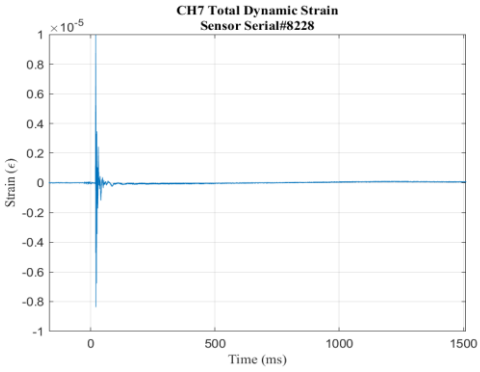
b.



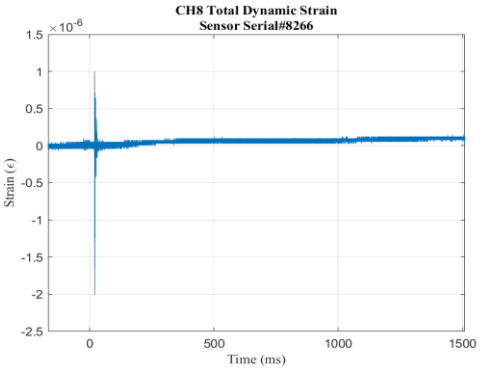
c.



d.



e.



f.

Figure A.112. Strain response, steel penetrator (10 lb), test date: 27 Aug 19

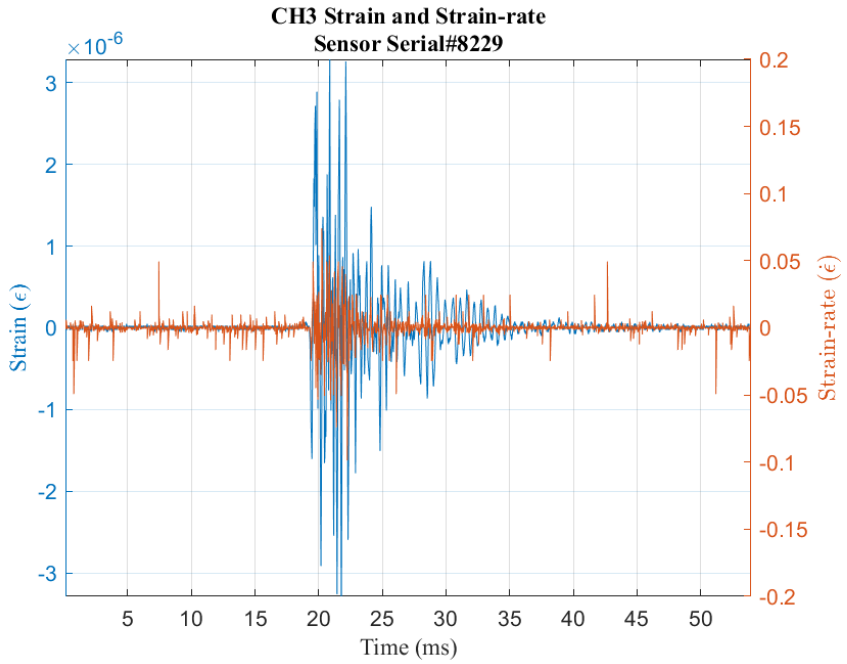


Figure A.113. Strain and strain rate, steel penetrator (10 lb), CH3

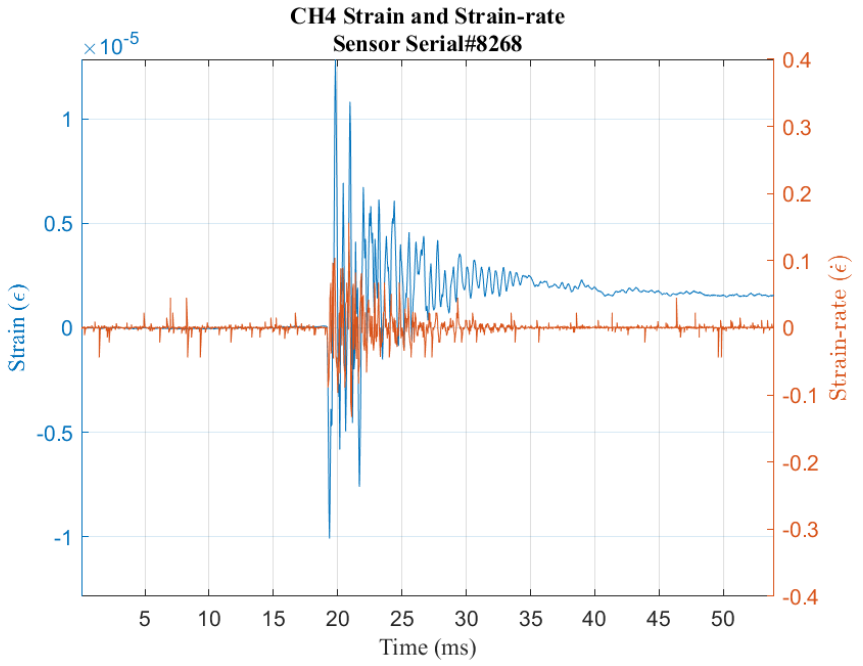


Figure A.114. Strain and strain rate, steel penetrator (10 lb), CH4

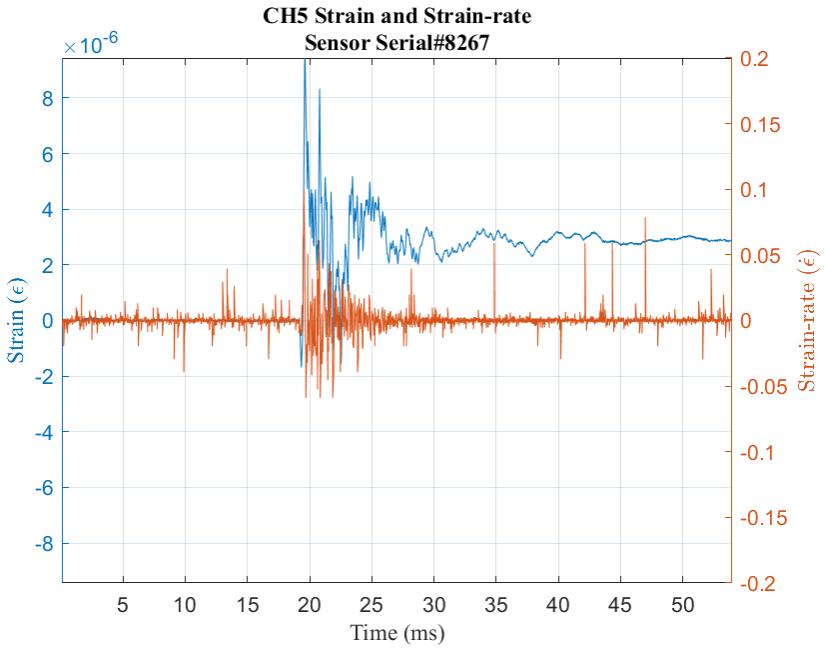


Figure A.115. Strain and strain rate, steel penetrator (10 lb), CH5

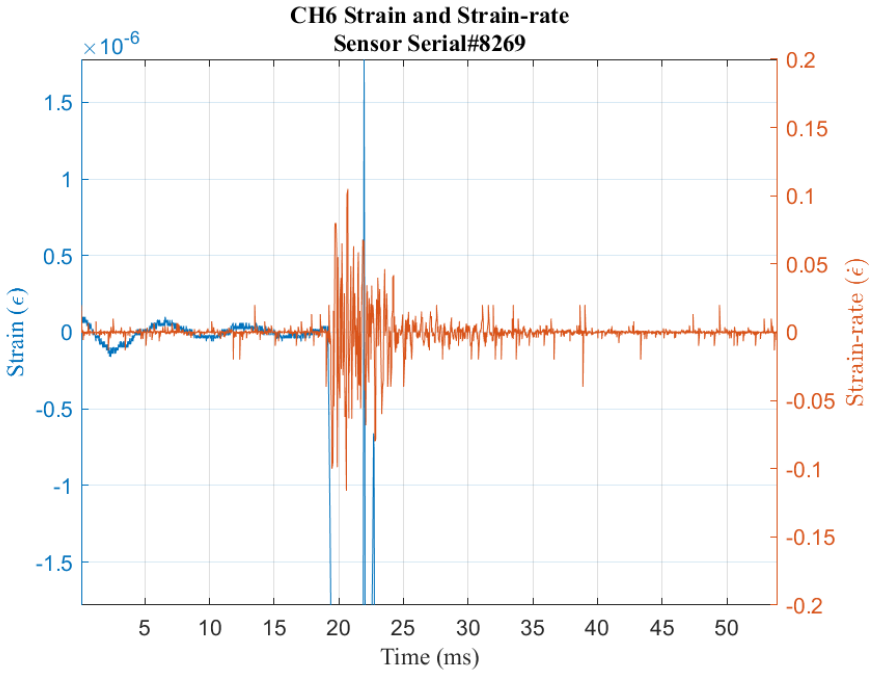


Figure A.116. Strain and strain rate, steel penetrator (10 lb), CH6

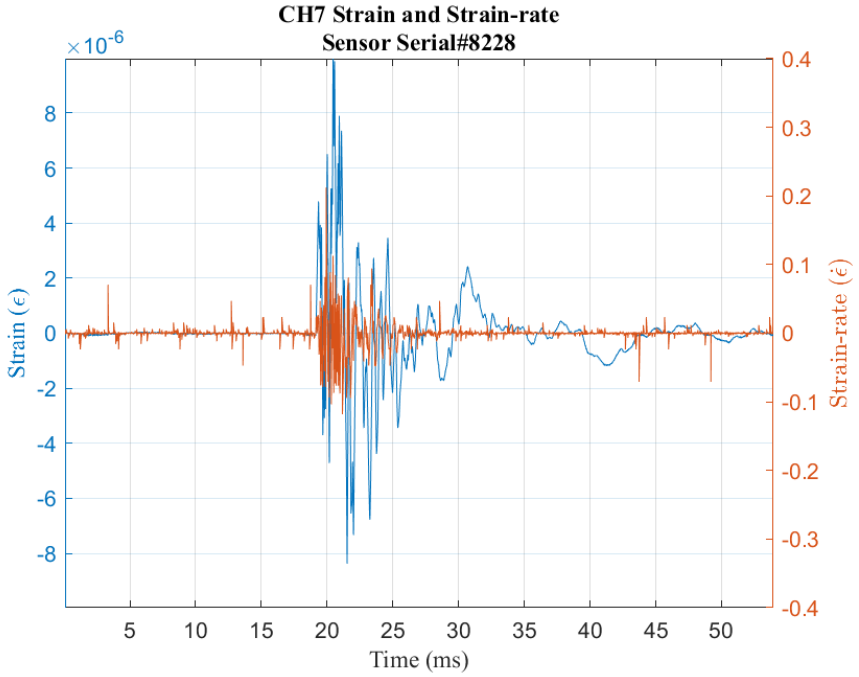


Figure A.117. Strain and strain rate, steel penetrator (10 lb), CH7

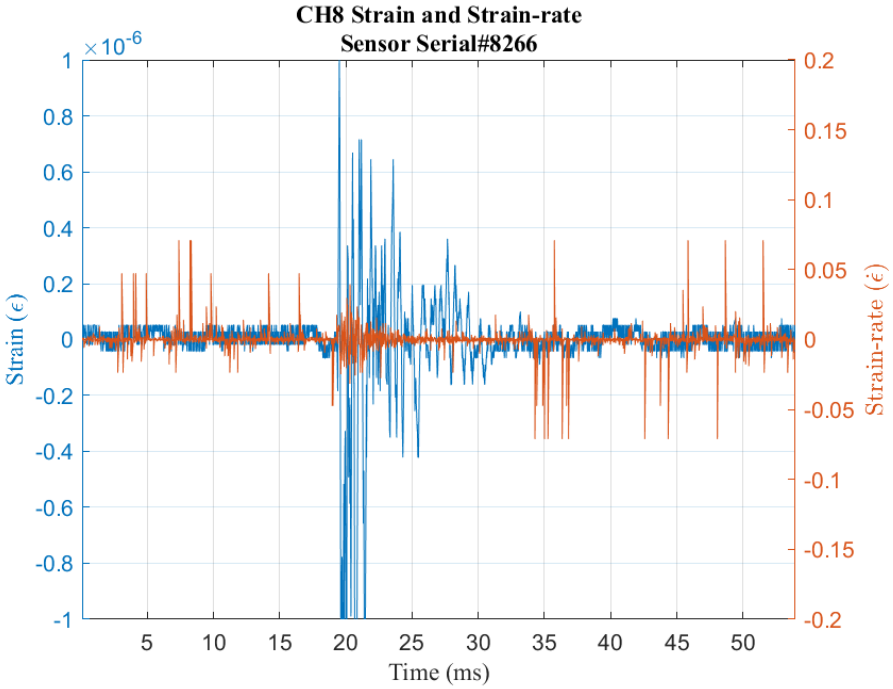
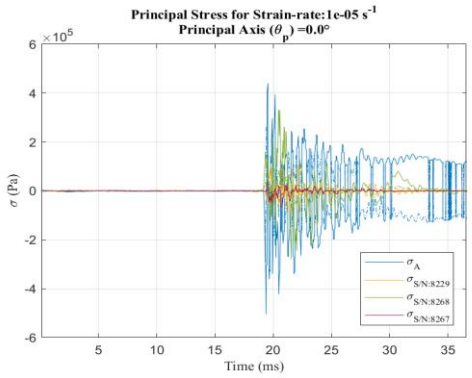
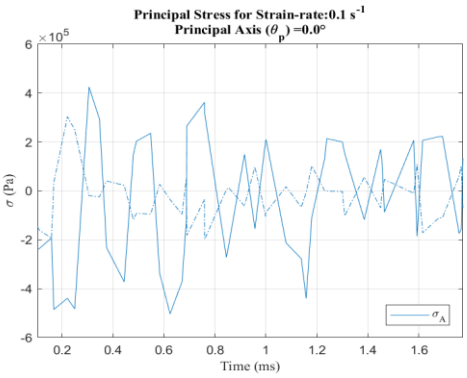


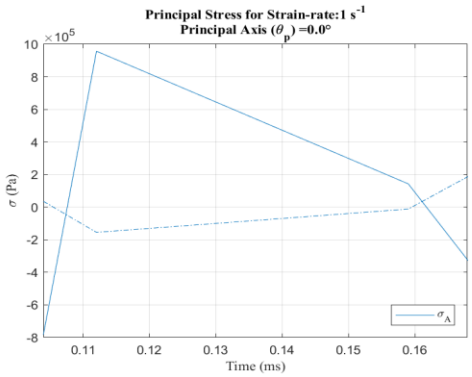
Figure A.118. Strain and strain rate, steel penetrator (10 lb), CH8



a.

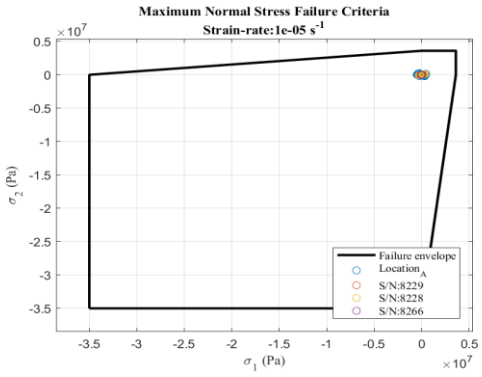


b.

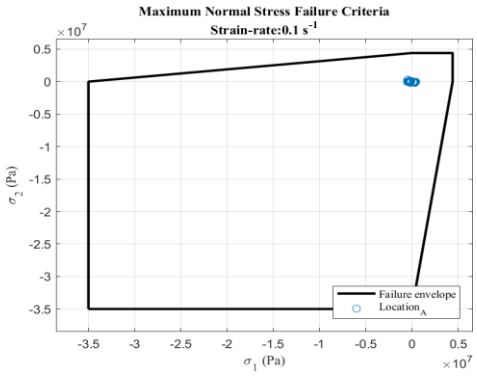


c.

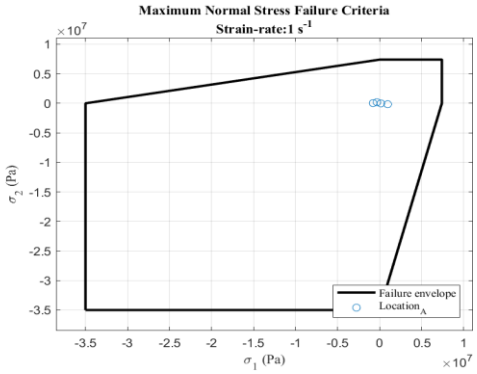
Figure A.119. Principal stress at strain rate, steel penetrator (10 lb), test date: 27 Aug 19



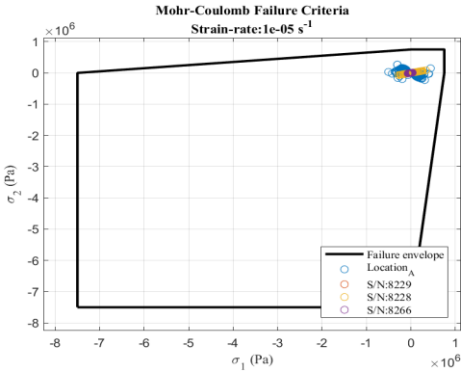
a.



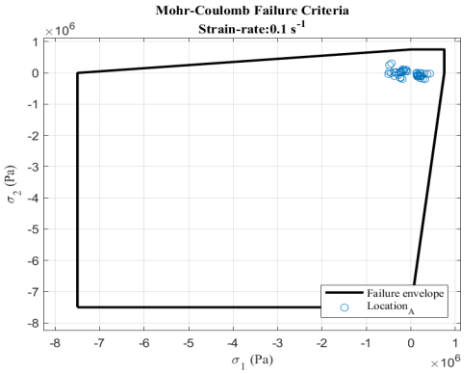
b.



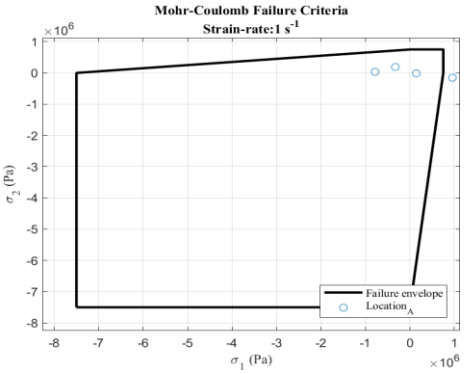
c.



d.

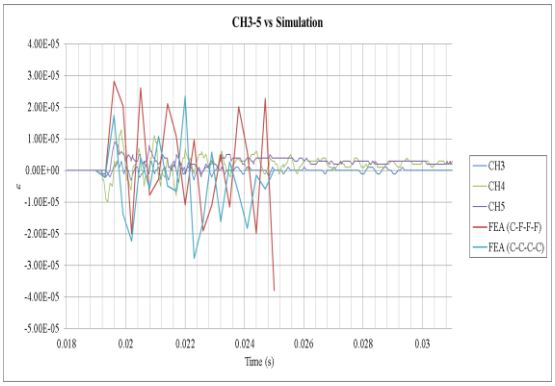


e.

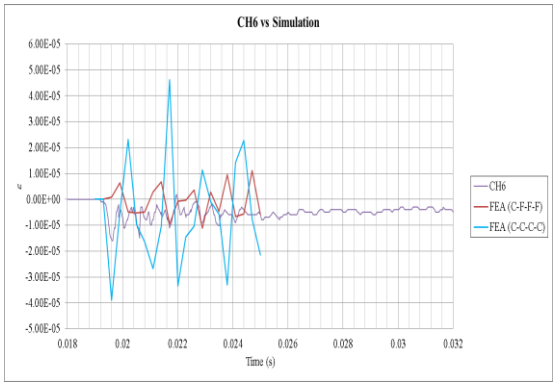


f.

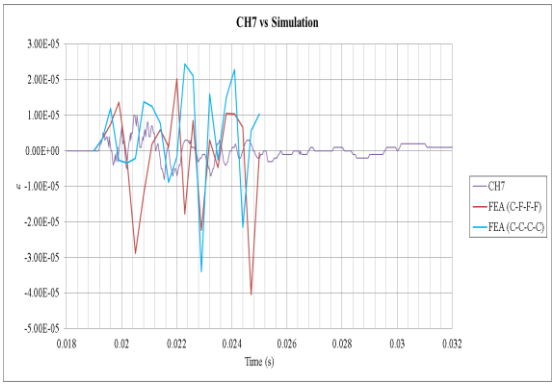
Figure A.120. Failure criterion envelopes, steel penetrator (10 lb), test date: 27 Aug 19



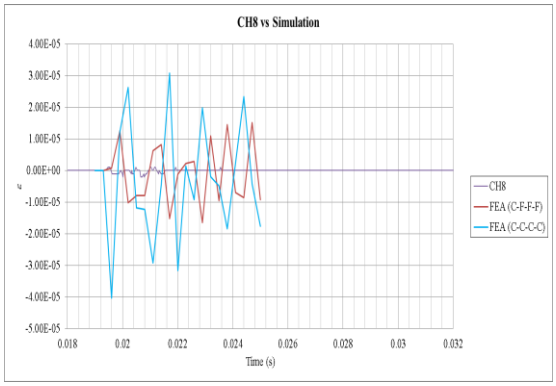
a.



b.



c.



d.

Figure A.121. Strain response and FEA comparison, steel penetrator (10 lb), test date: 27 Aug 19

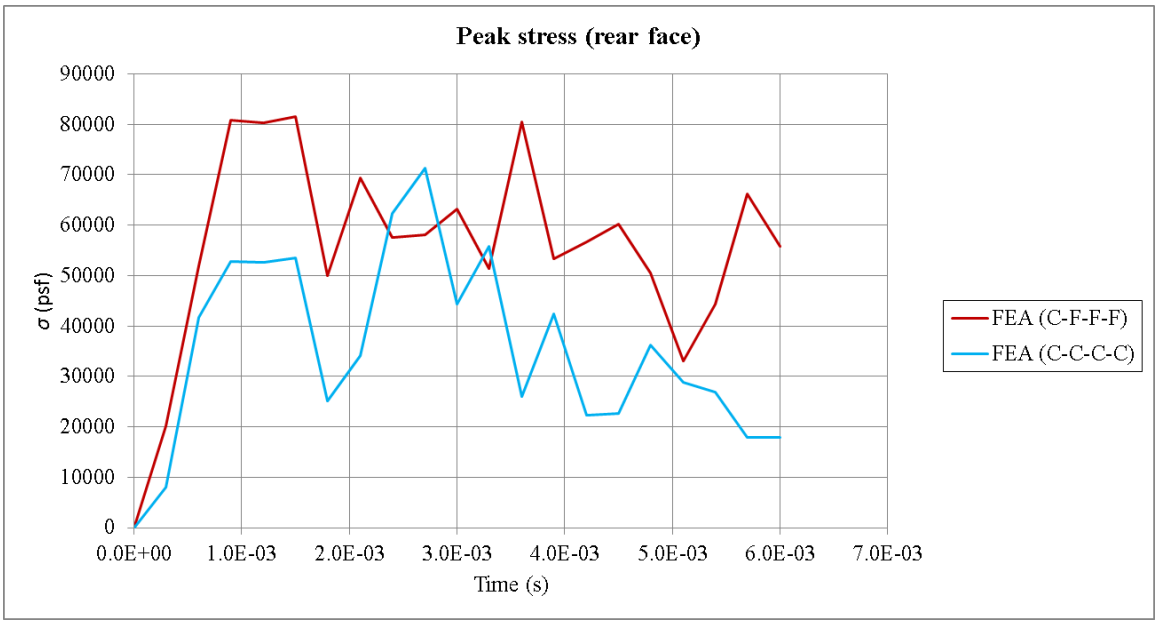
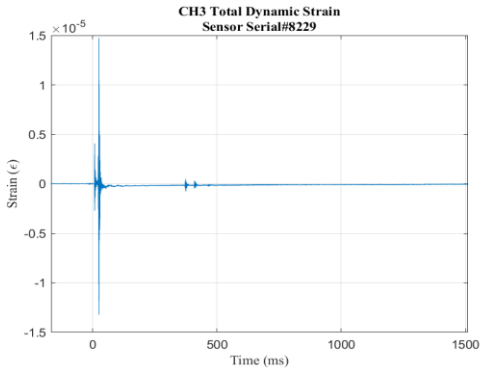


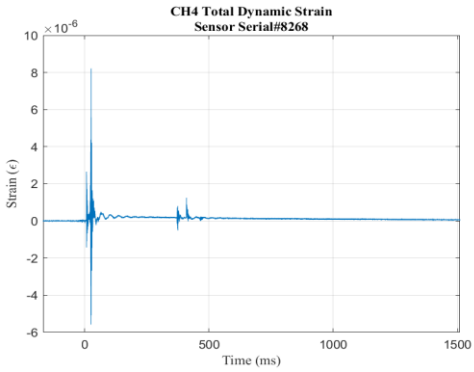
Figure A.122. FEA peak rear face stress, steel penetrator (10 lb), test date: 27 Aug 19

K, Steel rail (25 lb), 6 September 2019

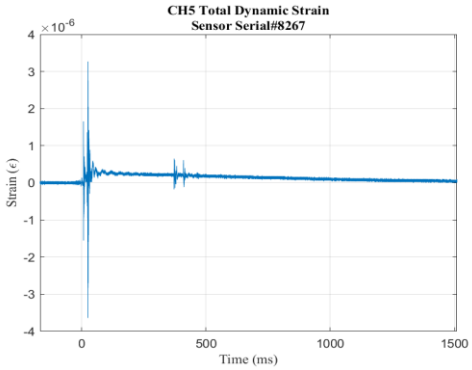
Figure A.123. Steel rail (25 lb)



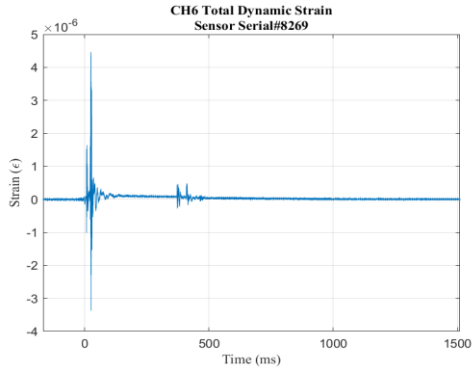
a.



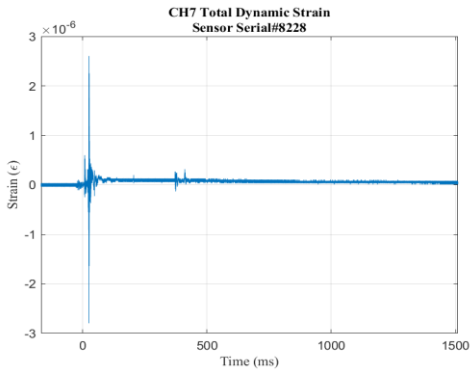
b.



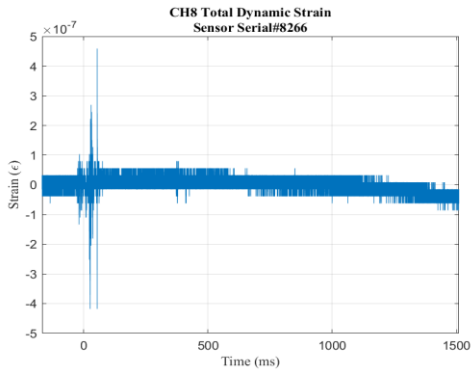
c.



d.



e.



f.

Figure A.124. Strain response, steel rail (25 lb), test date: 6 Sep 19

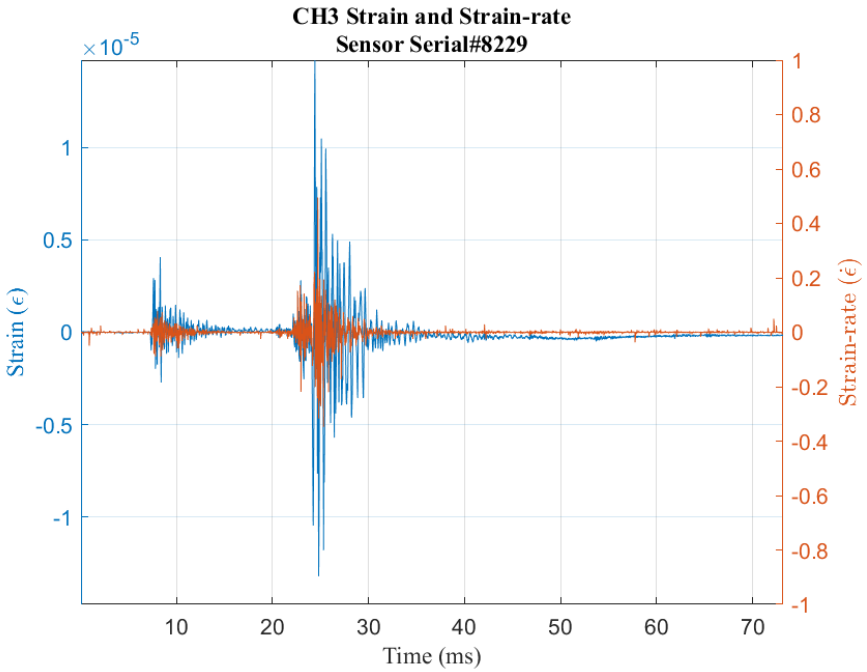


Figure A.125. Strain and strain rate, steel rail (25 lb), CH3

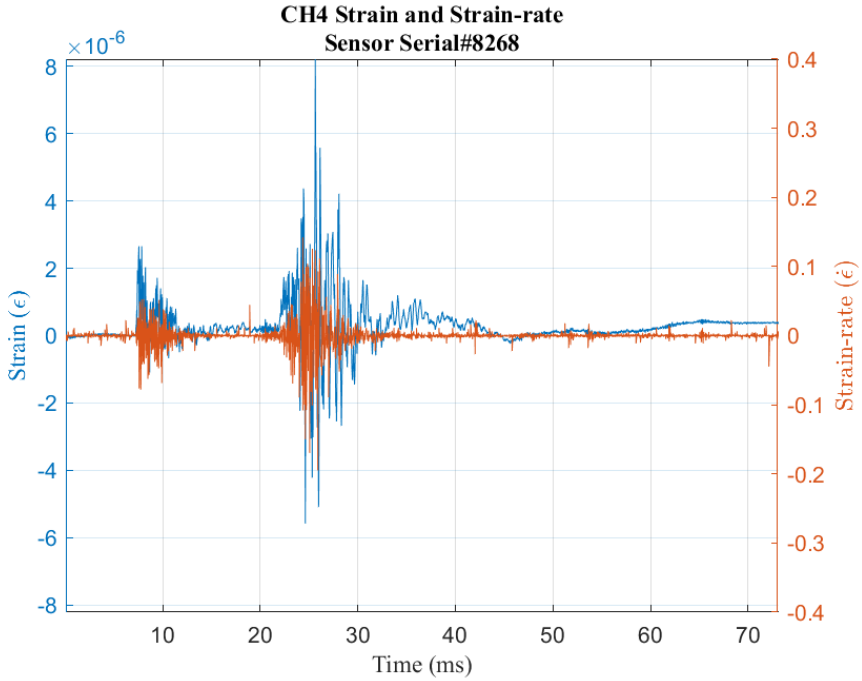


Figure A.126. Strain and strain rate, steel rail (25 lb), CH4

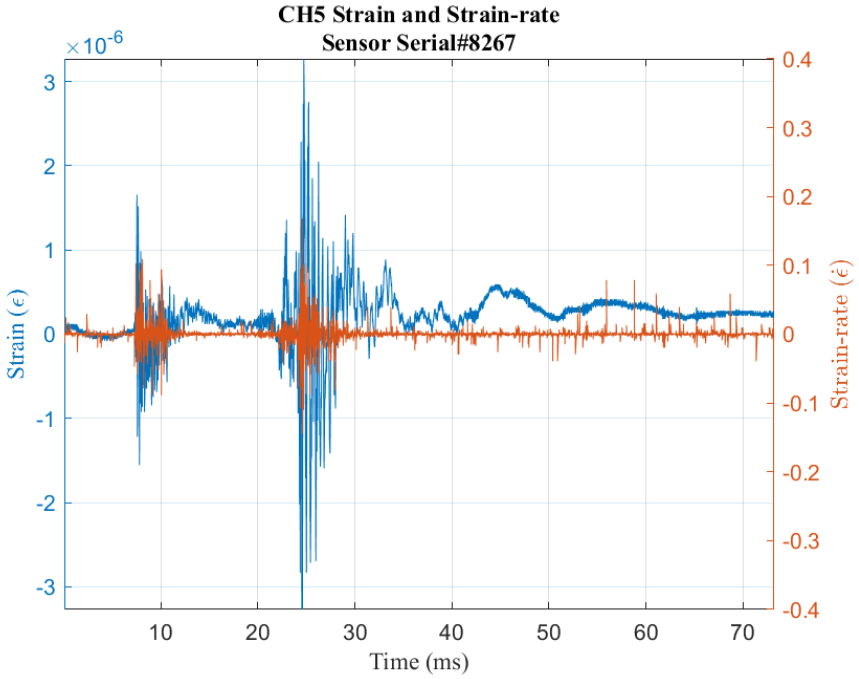


Figure A.127. Strain and strain rate, steel rail (25 lb), CH5

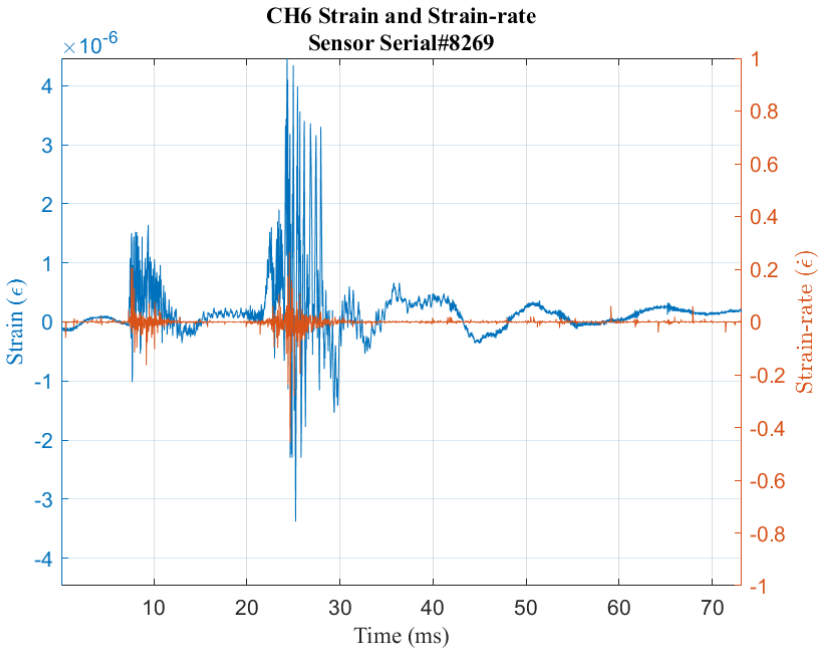


Figure A.128. Strain and strain rate, steel rail (25 lb), CH6

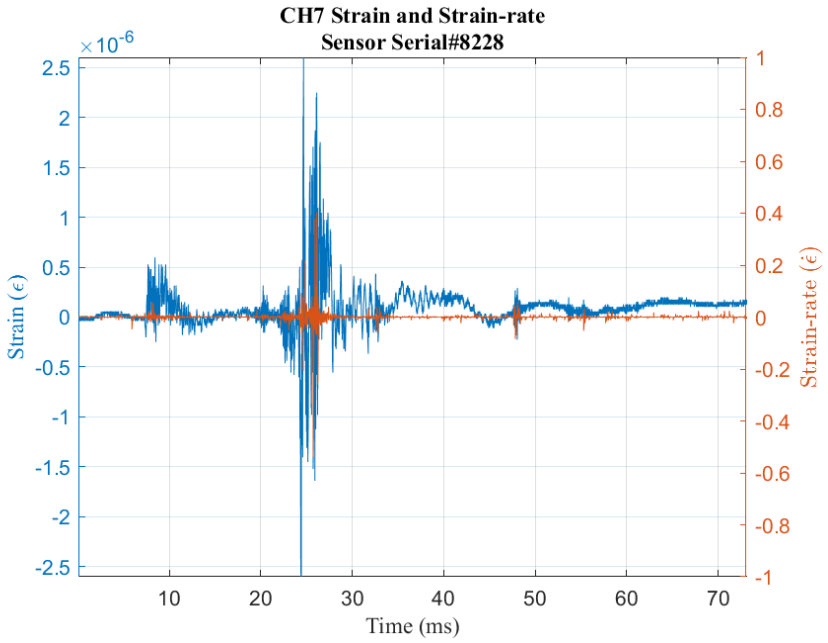


Figure A.129. Strain and strain rate, steel rail (25 lb), CH7

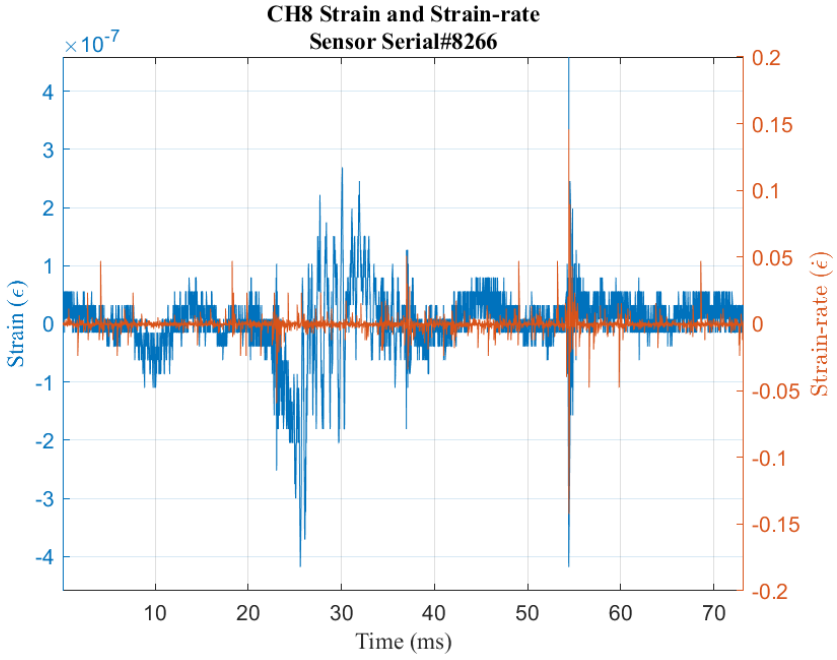
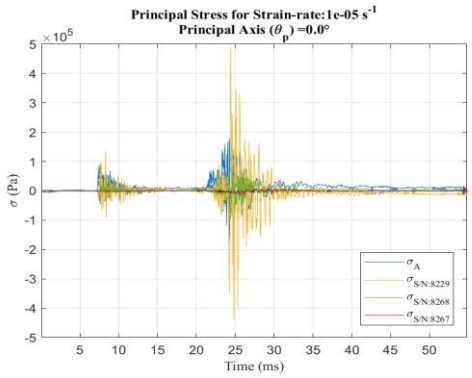
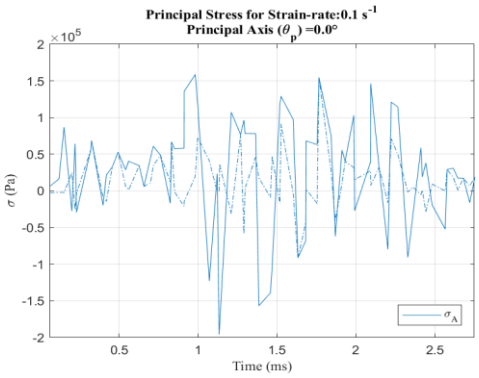


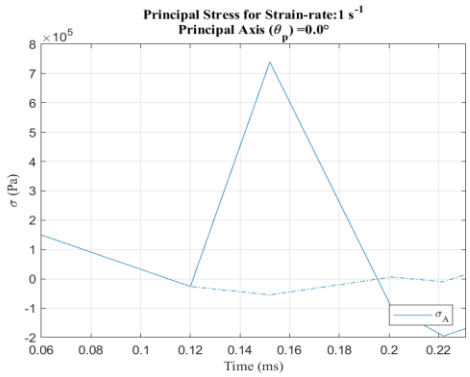
Figure A.130. Strain and strain rate, steel rail (25 lb), CH8



a.

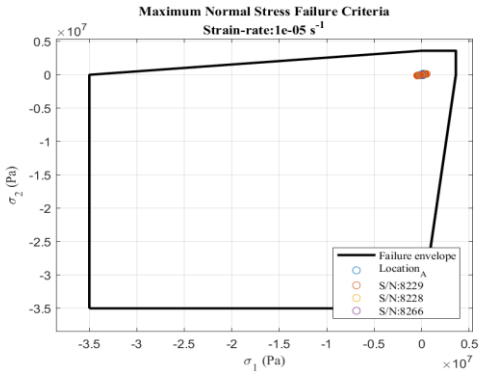


b.

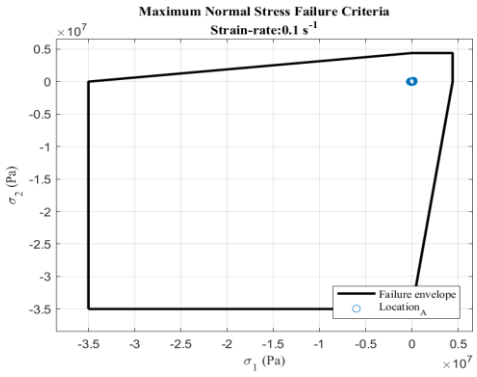


c.

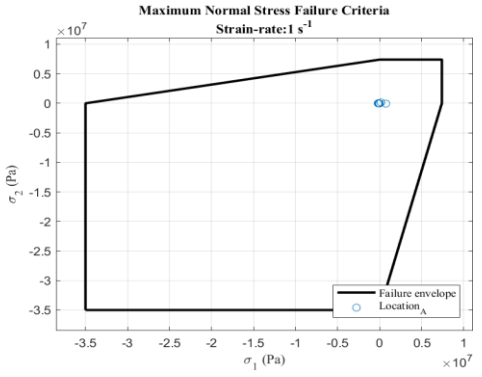
Figure A.131. Principal stress at strain-rate, steel rail (25 lb), test date: 6 Sep 19



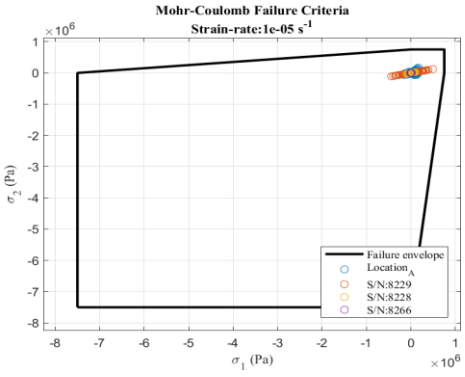
a.



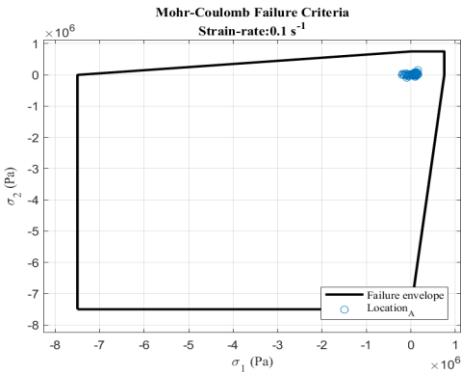
b.



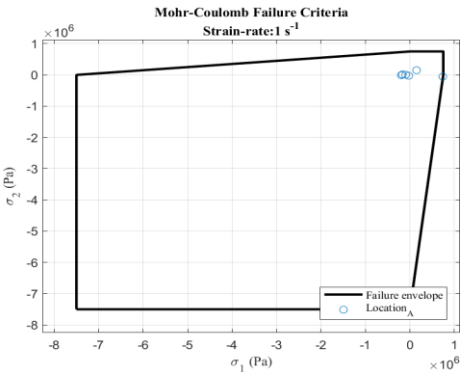
c.



d.

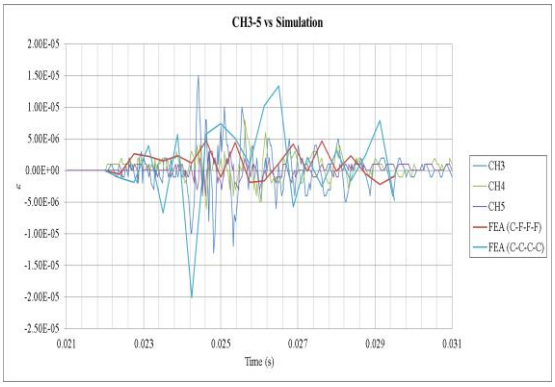


e.

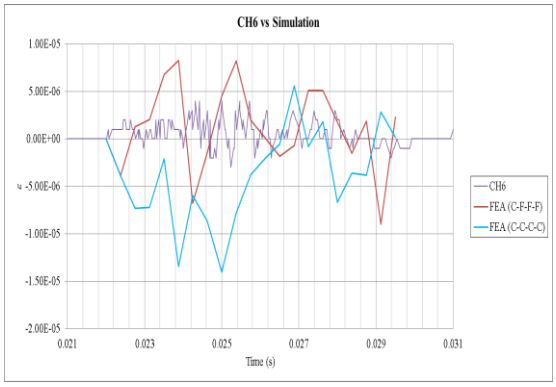


f.

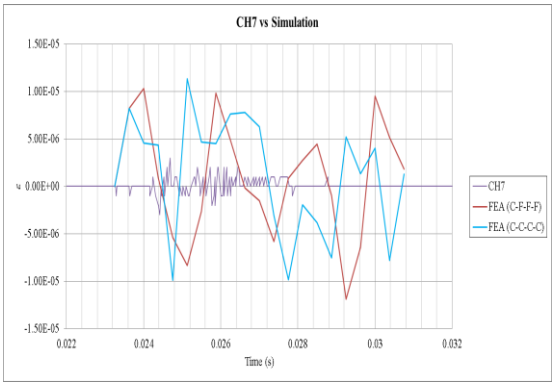
Figure A.132. Failure criterion envelopes, steel rail (25 lb), test date: 6 Sep 19



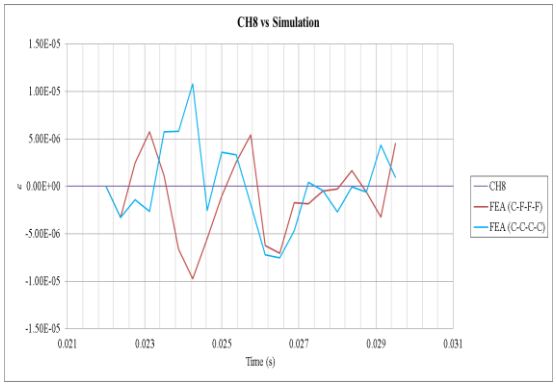
a.



b.



c.



d.

Figure A.133. Strain response and FEA comparison, steel rail (25 lb), test date: 6 Sep 19

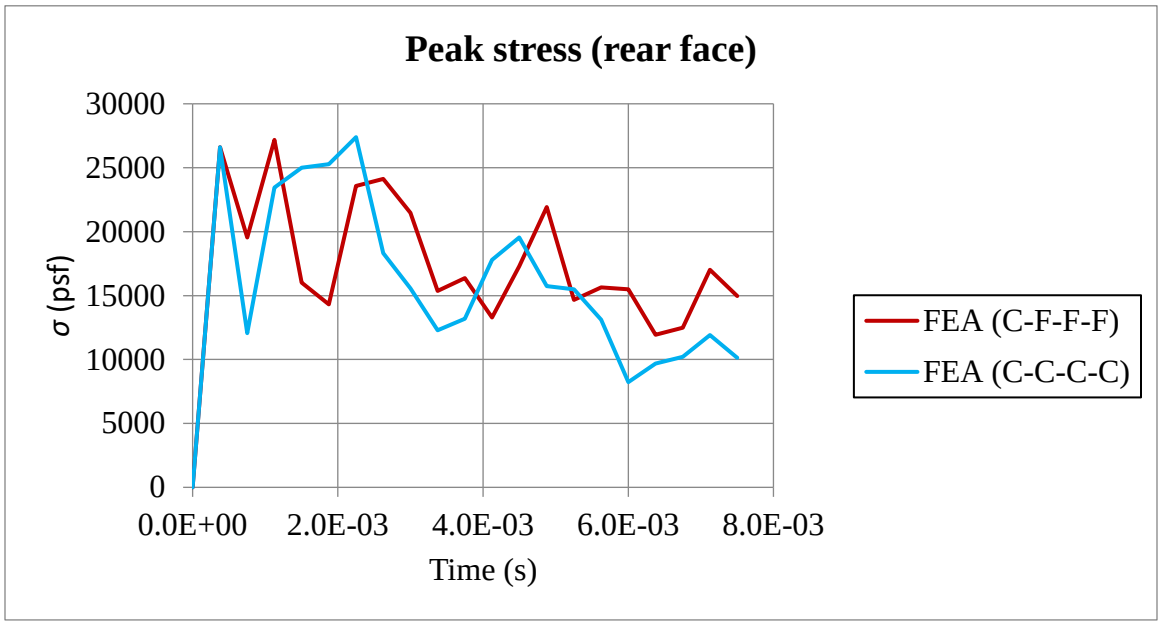
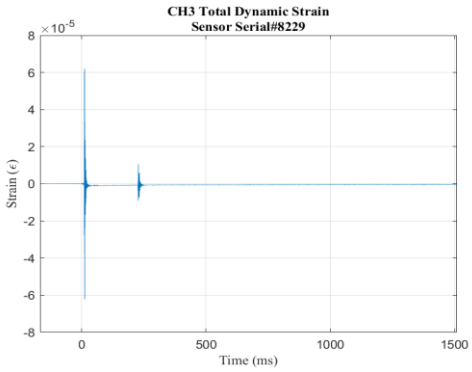


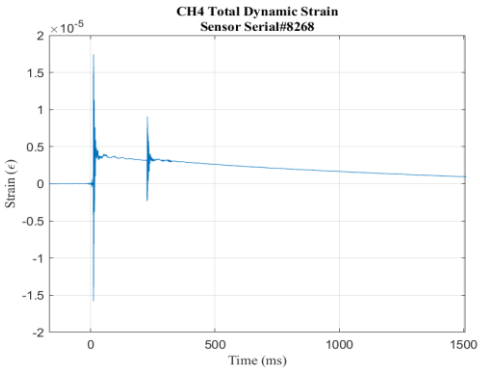
Figure A.134. FEA peak rear face stress, steel rail (25 lb), test date: 6 Sep 19

L, Steel penetrator (20 lb), 6 September 2019

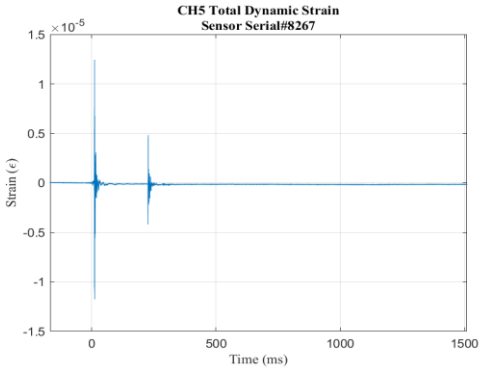
Figure A.135. Steel penetrator (20 lb), test date: 6 Sep 19



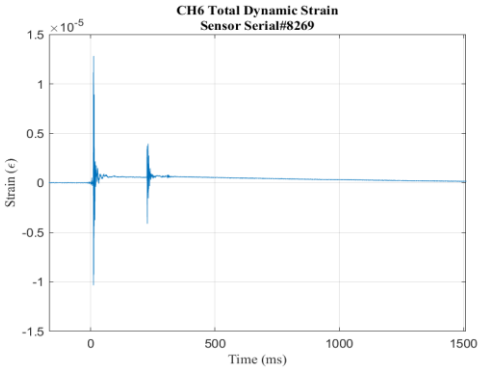
a.



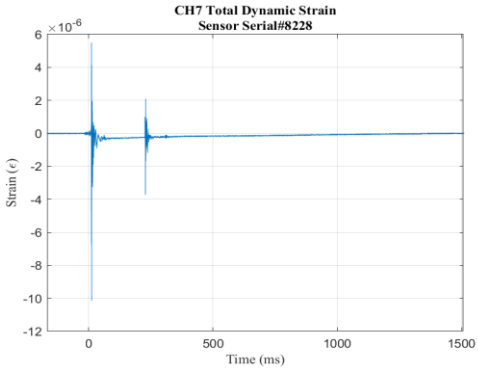
b.



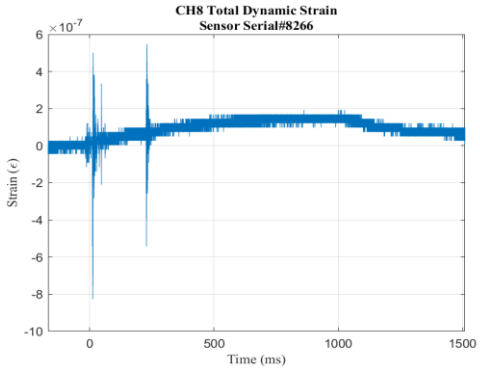
c.



d.



e.



f.

Figure A.136. Strain response, steel penetrator (20 lb), test date: 6 Sep 19

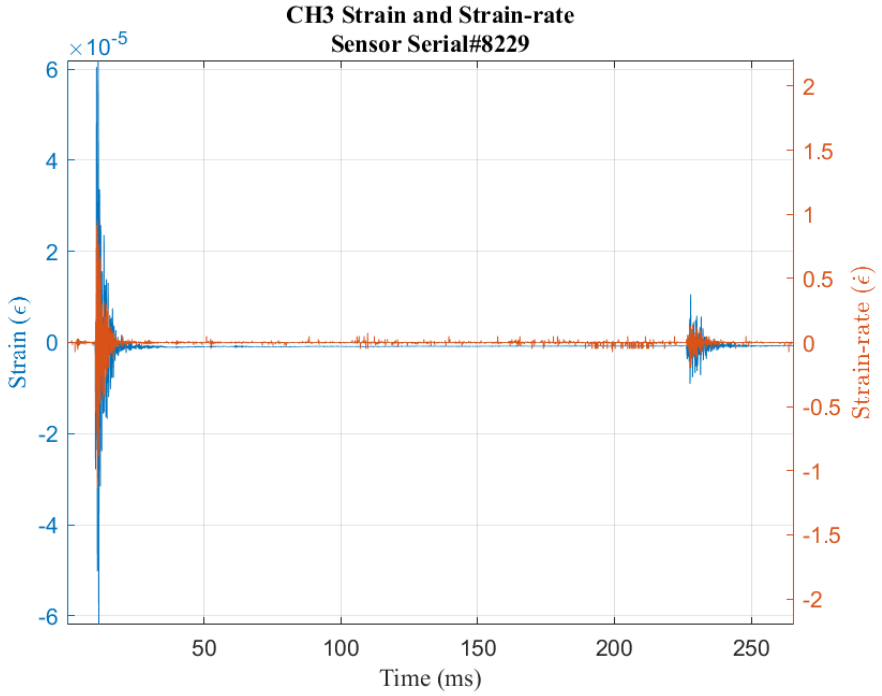


Figure A.137. Strain and strain rate, steel penetrator (20 lb), 6 Sep 19, CH3

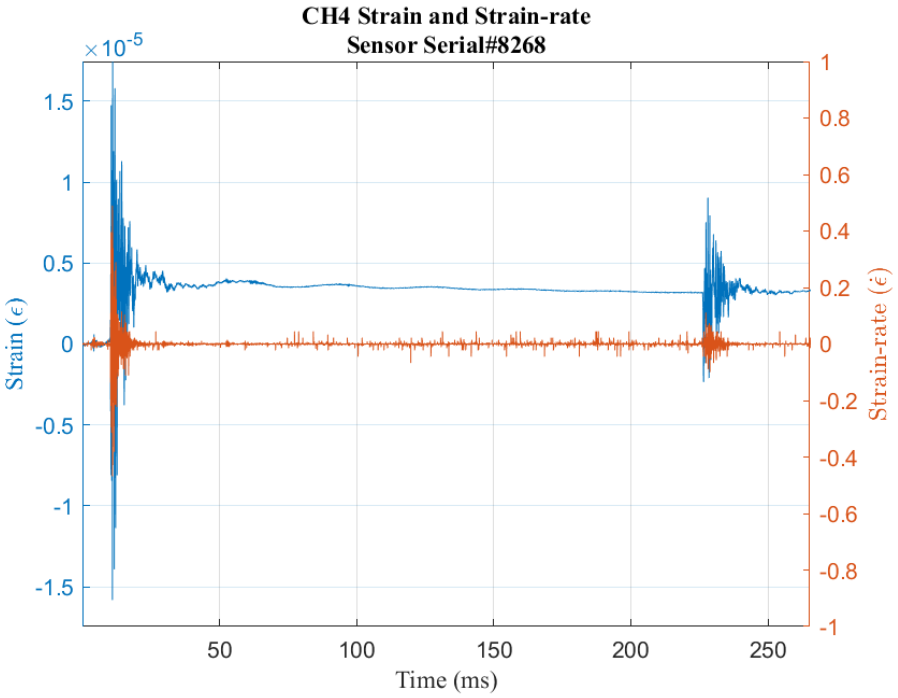


Figure A.138. Strain and strain rate, steel penetrator (20 lb), 6 Sep 19, CH4

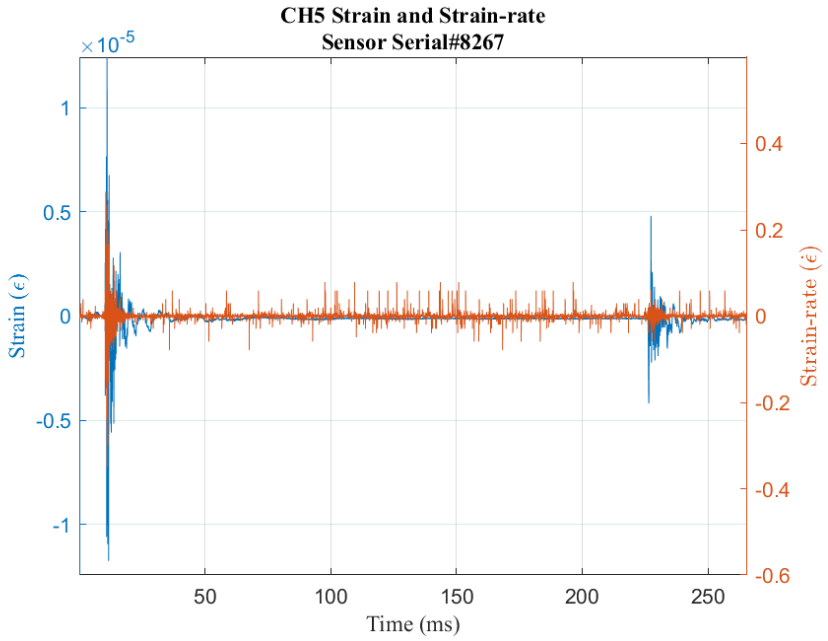


Figure A.139. Strain and strain rate, steel penetrator (20 lb), 6 Sep 19, CH5

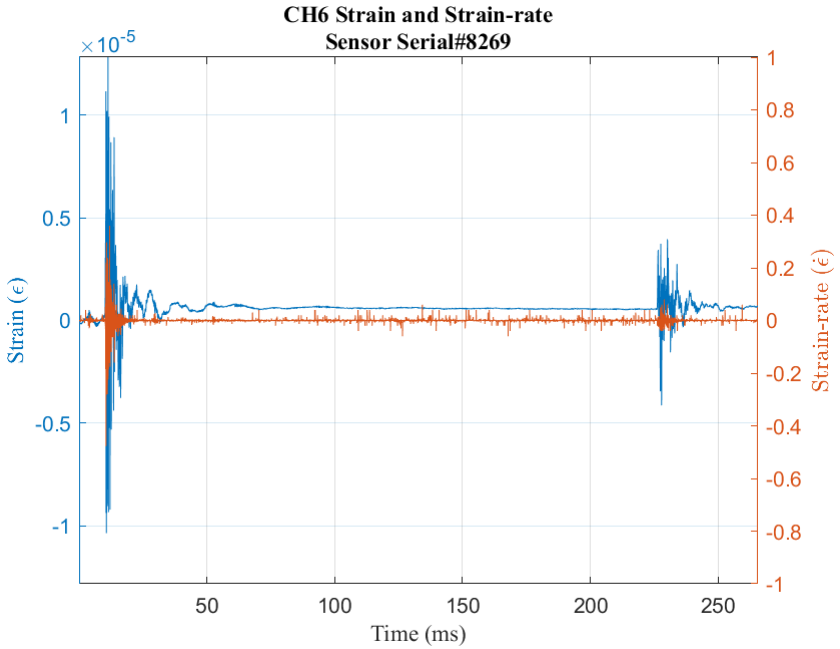


Figure A.140. Strain and strain rate, steel penetrator (20 lb), 6 Sep 19, CH6

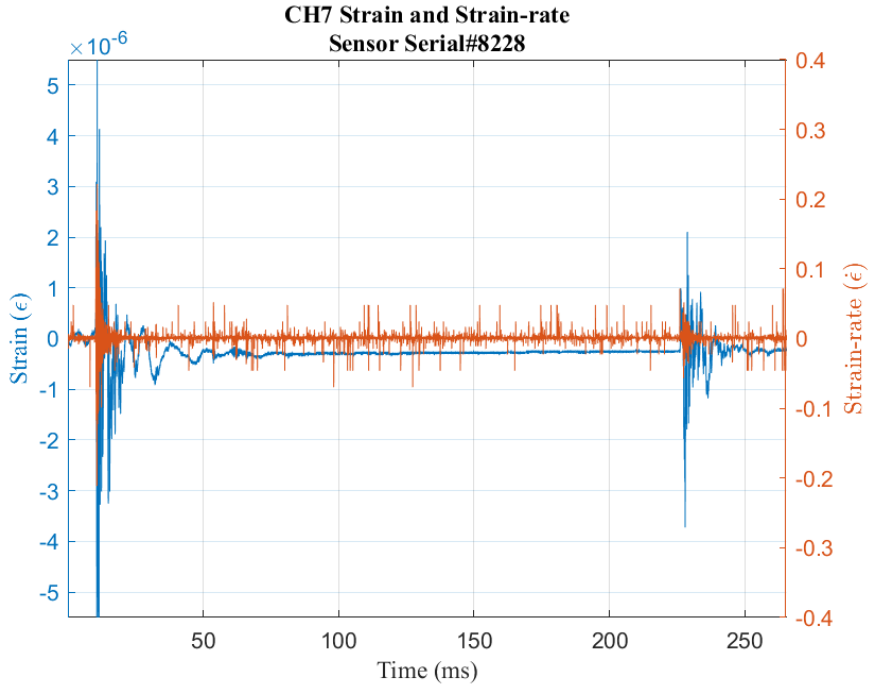


Figure A.141. Strain and strain rate, steel penetrator (20 lb), 6 Sep 19, CH7

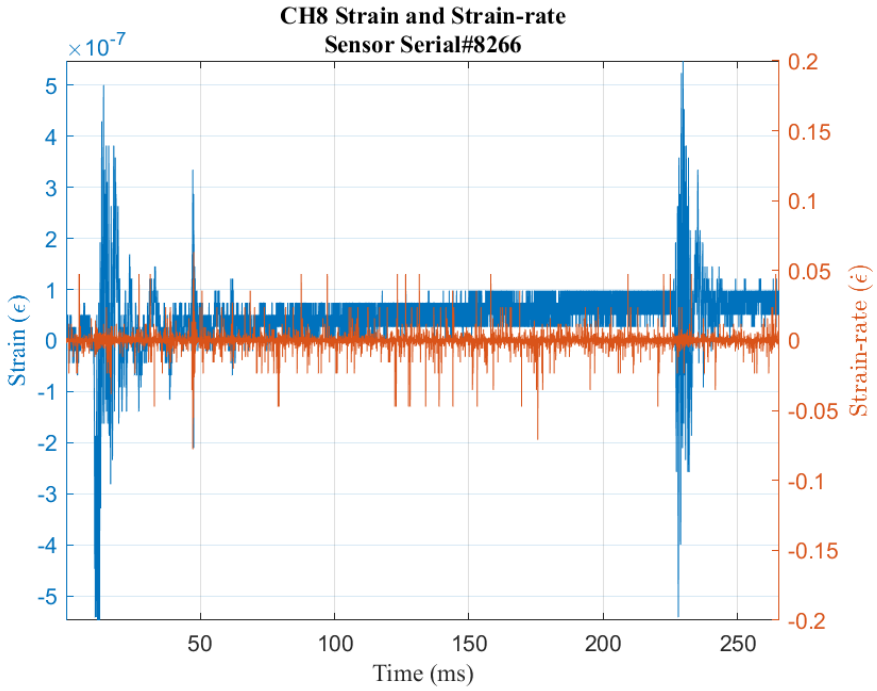
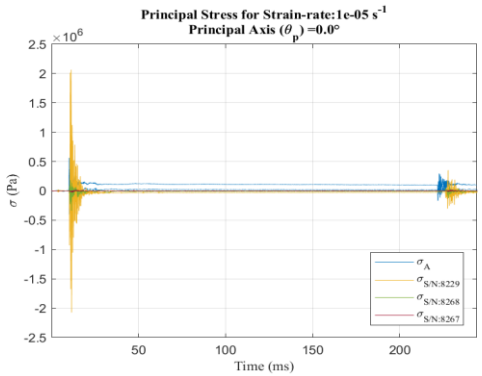
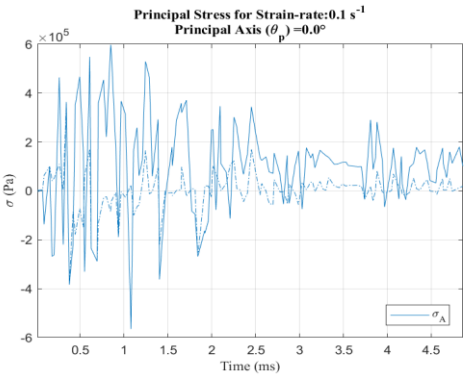


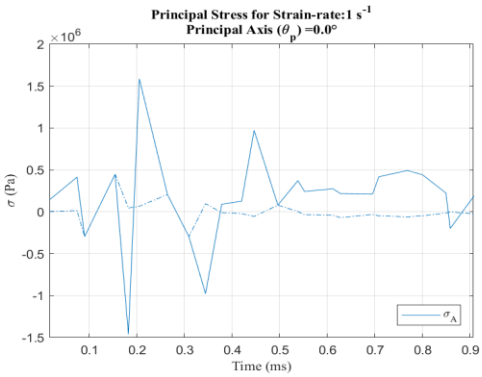
Figure A.142. Strain and strain rate, steel penetrator (20 lb), 6 Sep 19, CH8



a.



b.



c.

Figure A.143. Principal stress at strain rate, steel penetrator (20 lb), test date: 6 Sep 19

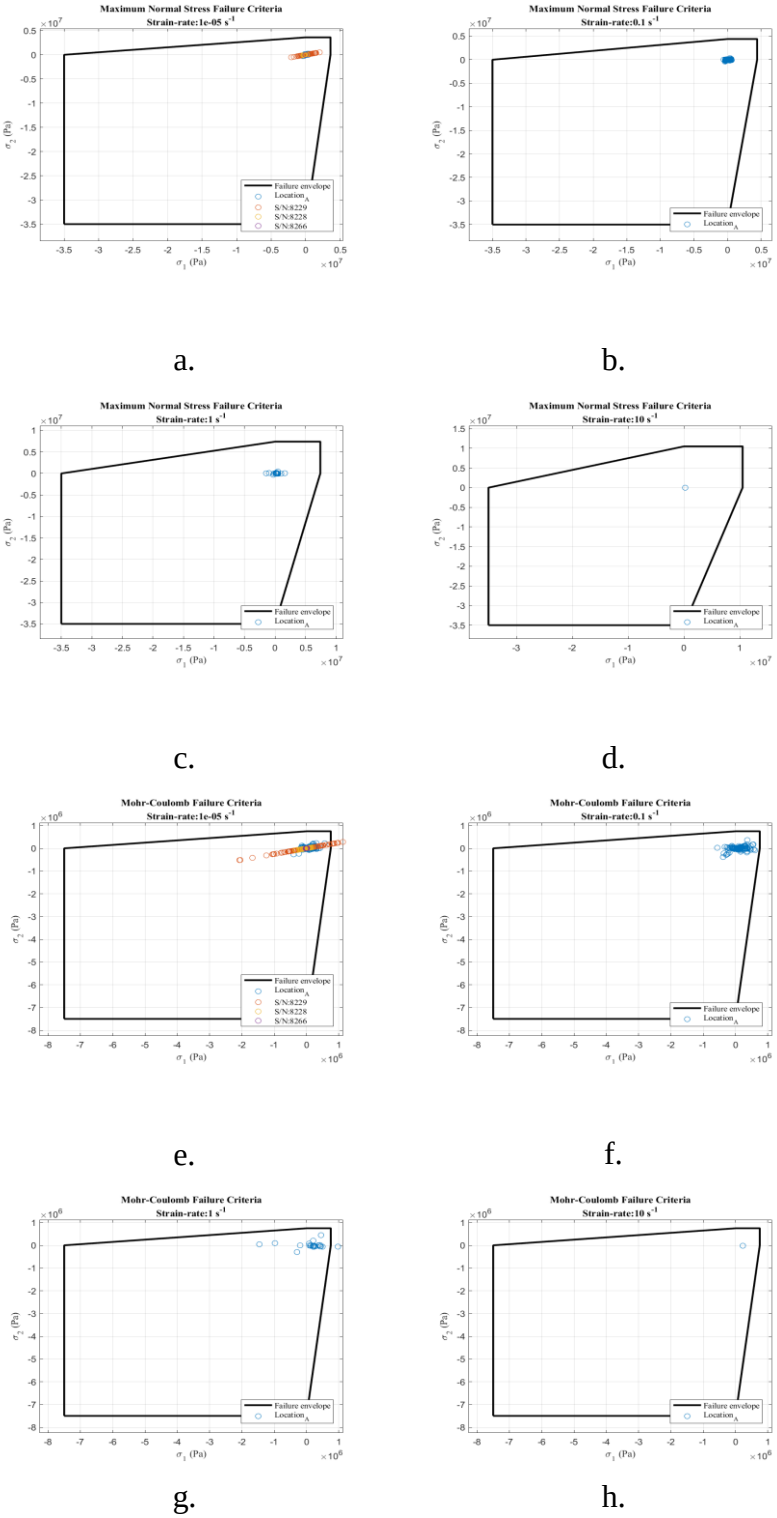
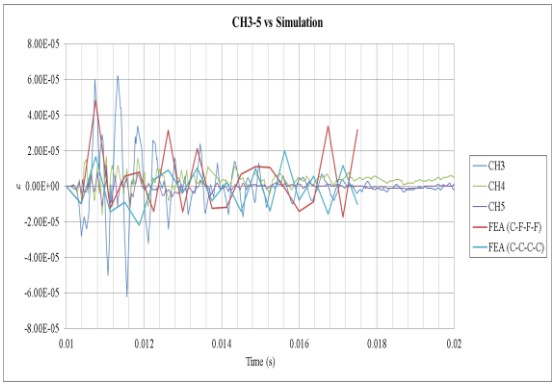
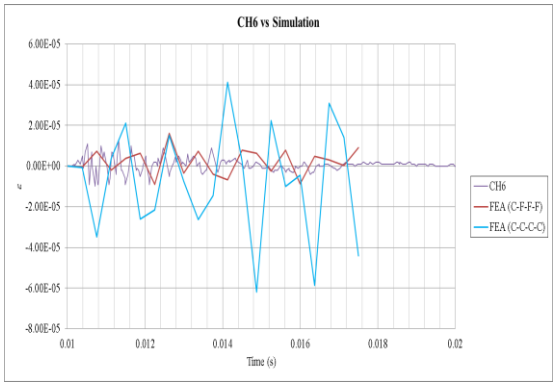


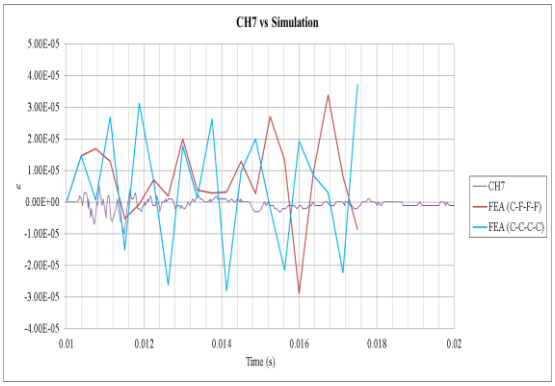
Figure A.144. Failure criterion envelopes, steel penetrator (20 lb), test date: 6 Sep 19



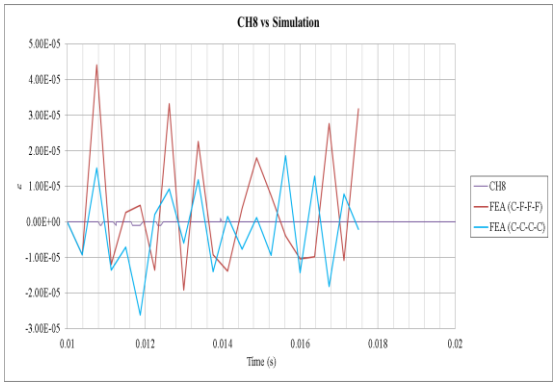
a.



b.



c.



d.

Figure A.145. Strain response and FEA comparison, steel penetrator (20 lb), test date: 6 Sep 19

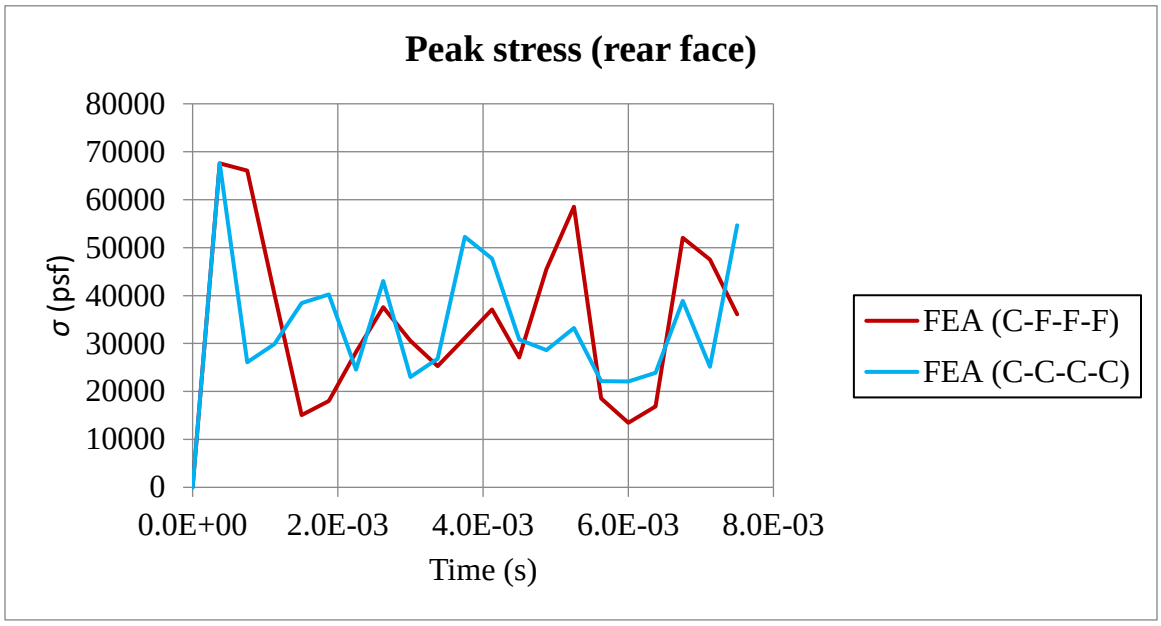
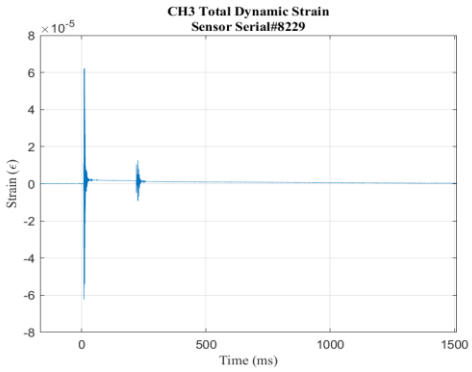


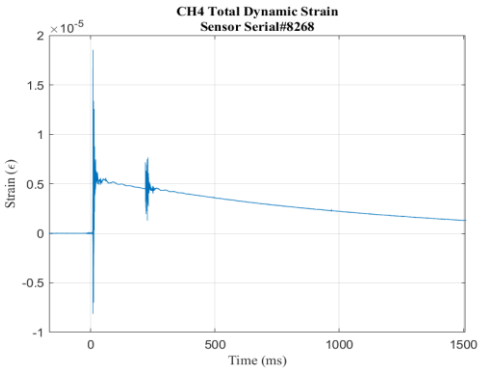
Figure A.146. FEA peak rear face stress, steel penetrator (20 lb), test date: 6 Sep 19

M, Steel penetrator (30 lb), 6 September 2019

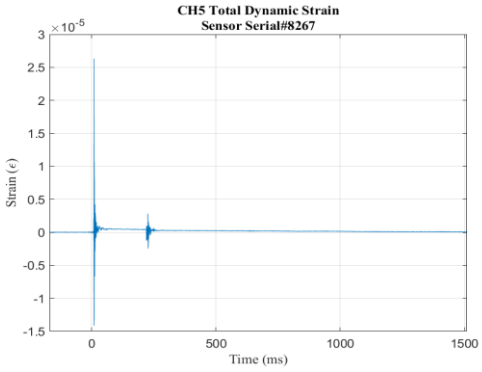
Figure A.147. Steel penetrator (30 lb), test date: 6 Sep 19



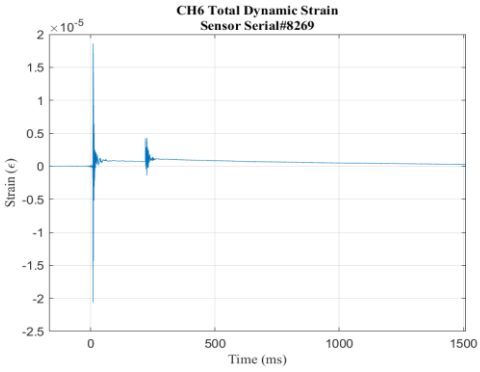
a.



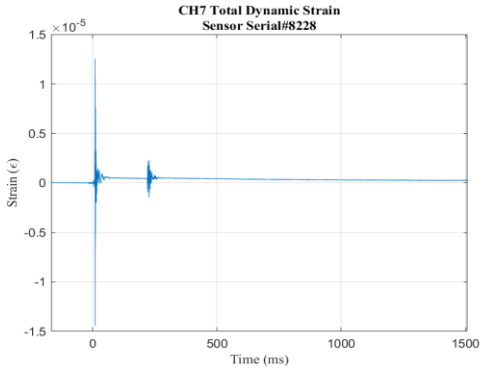
b.



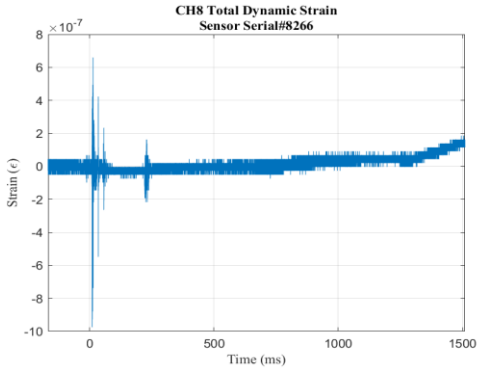
c.



d.



e.



f.

Figure A.148. Strain response, steel penetrator (30 lb), test date: 6 Sep 19

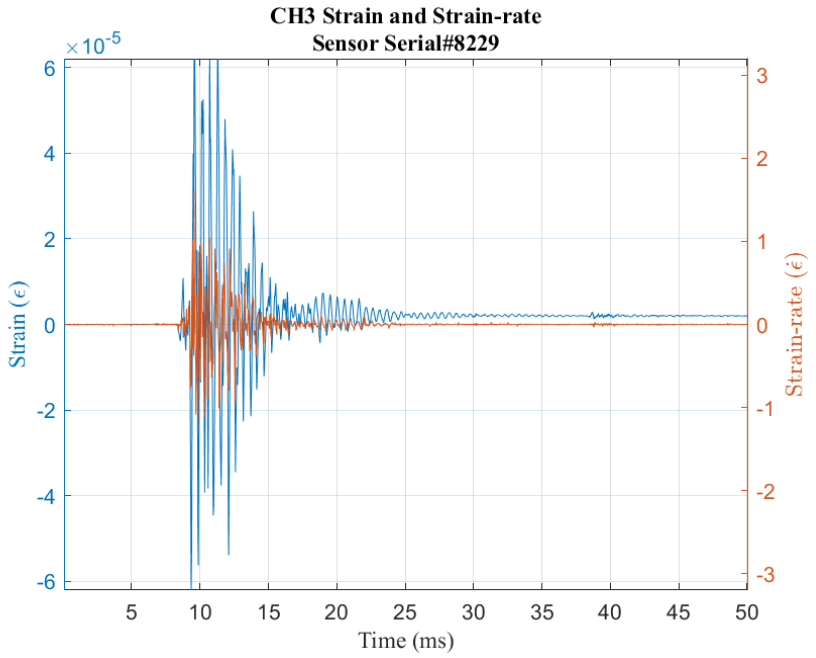


Figure A.149. Strain and strain rate, steel penetrator (30 lb), 6 Sep 19, CH3

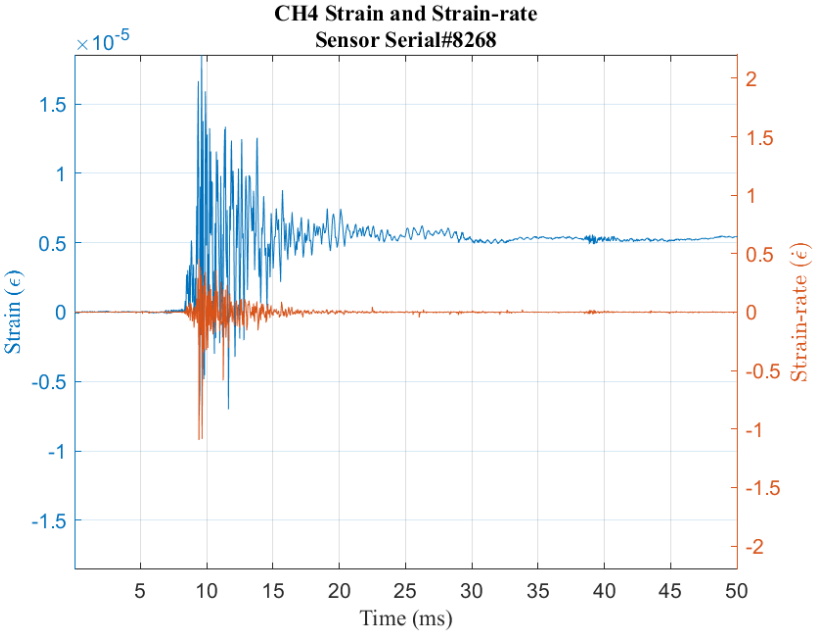


Figure A.150. Strain and strain rate, steel penetrator (30 lb), 6 Sep 19, CH4

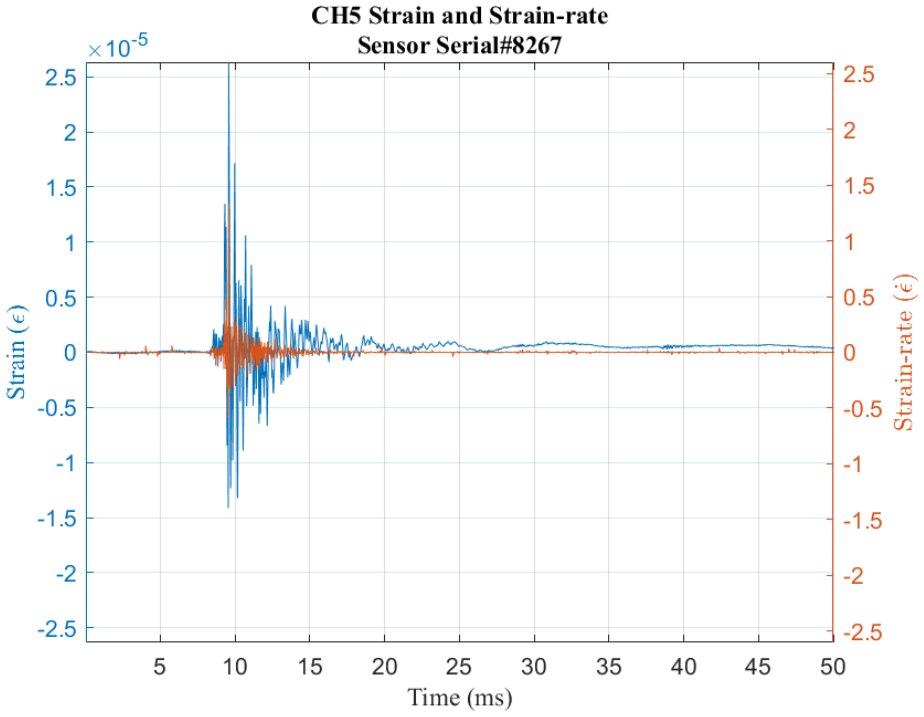


Figure A.151. Strain and strain rate, steel penetrator (30 lb), 6 Sep 19, CH5

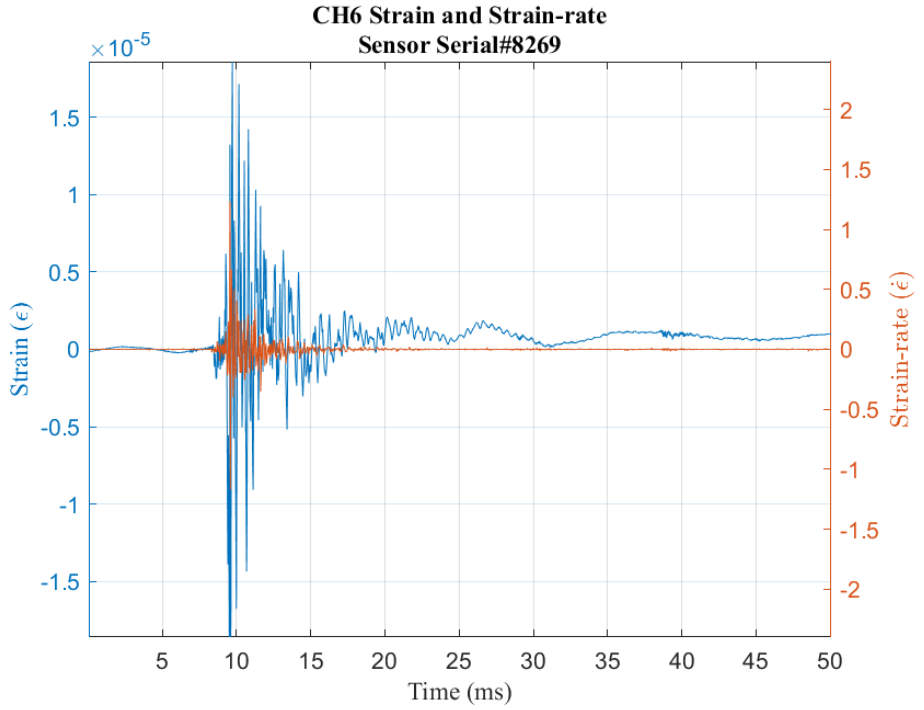


Figure A.152. Strain and strain rate, steel penetrator (30 lb), 6 Sep 19, CH6

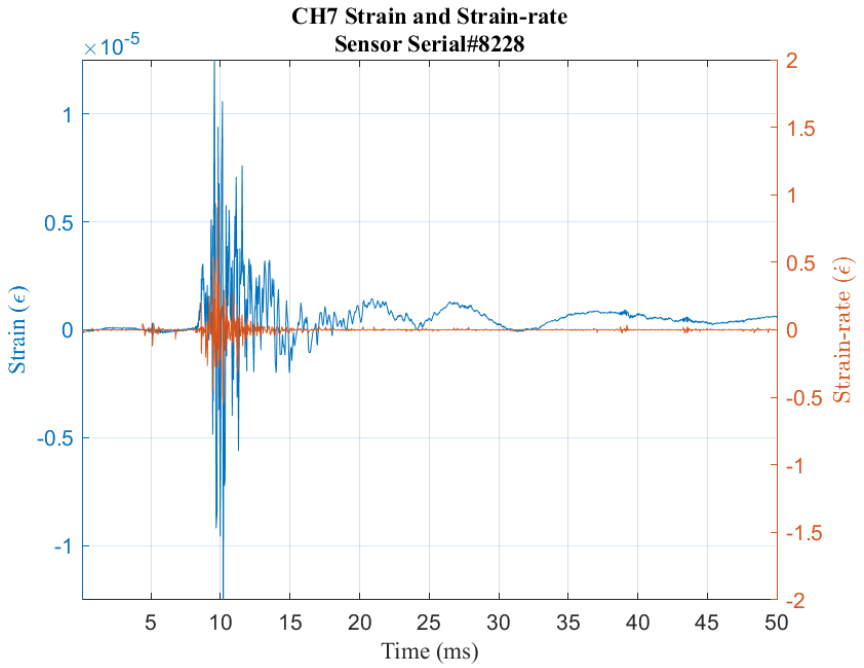


Figure A.153. Strain and strain rate, steel penetrator (30 lb), 6 Sep 19, CH7

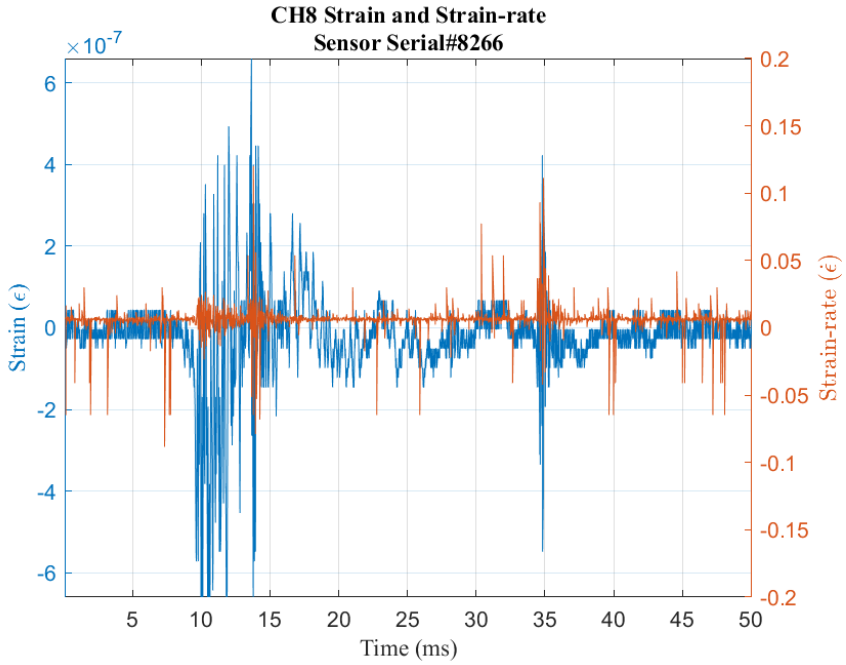
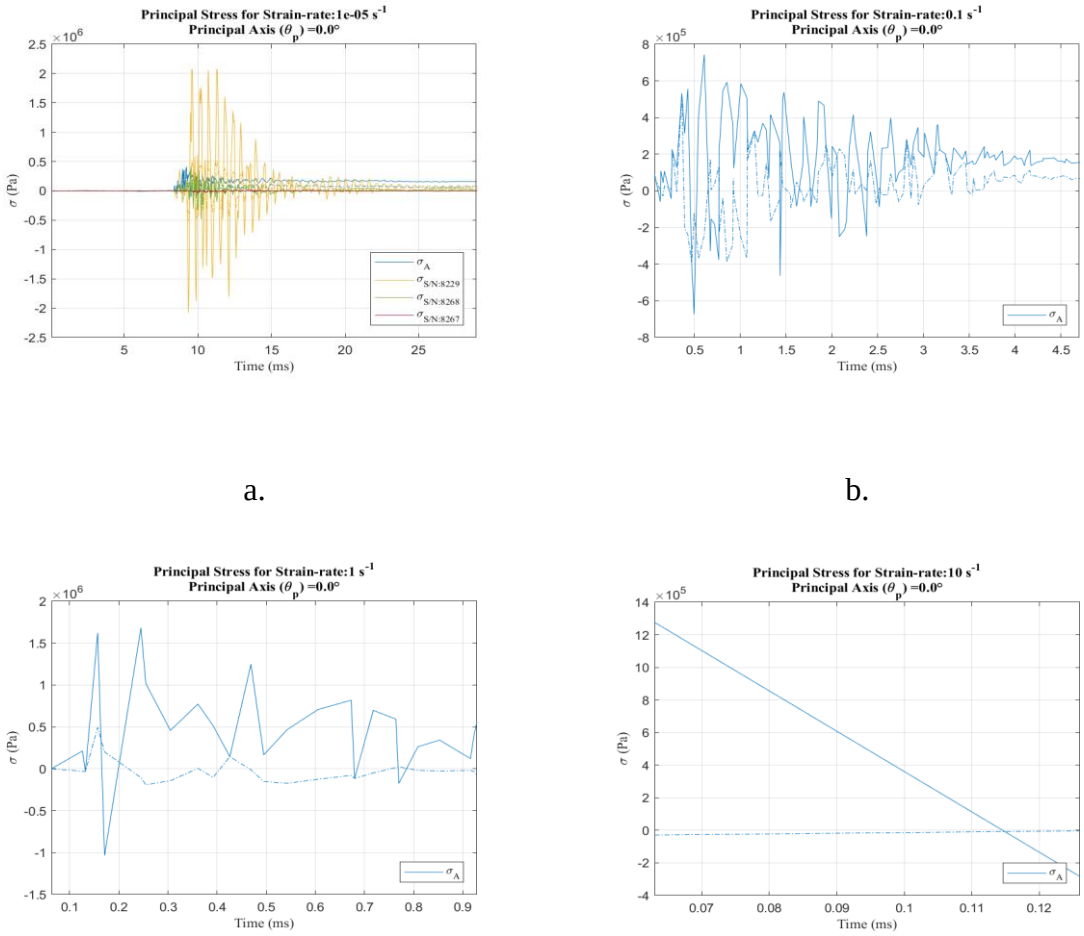


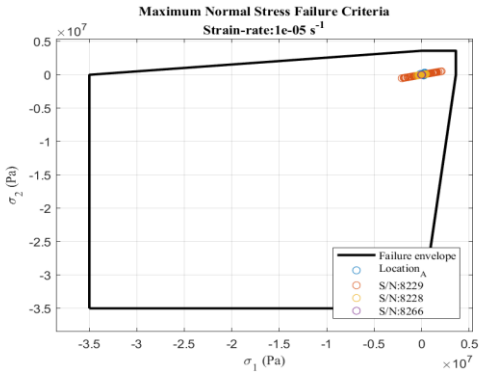
Figure A.154. Strain and strain rate, steel penetrator (30 lb), 6 Sep 19, CH8



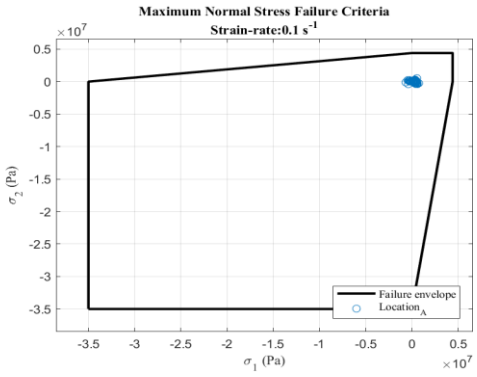
a. b.

c. d.

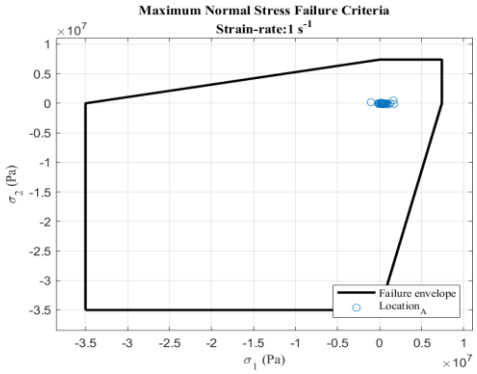
Figure A.155. Principal stress at strain rate, steel penetrator (30 lb), test date: 6 Sep 19



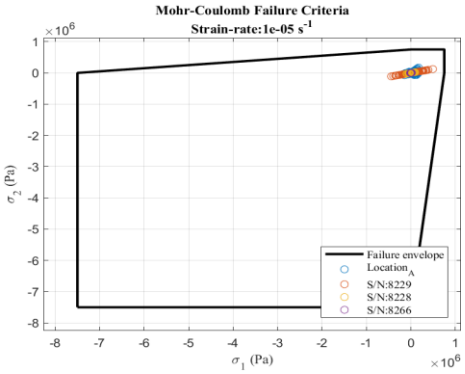
a.



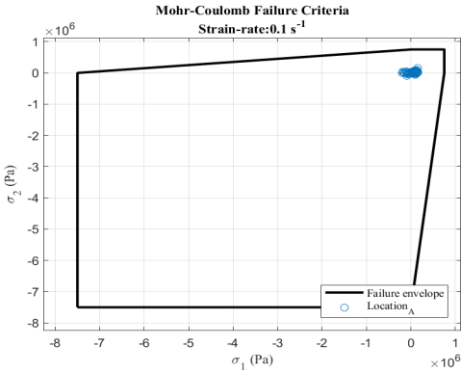
b.



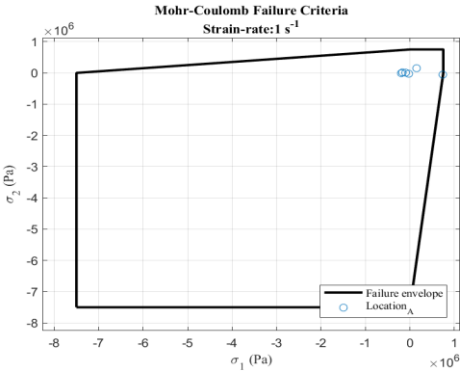
c.



d.

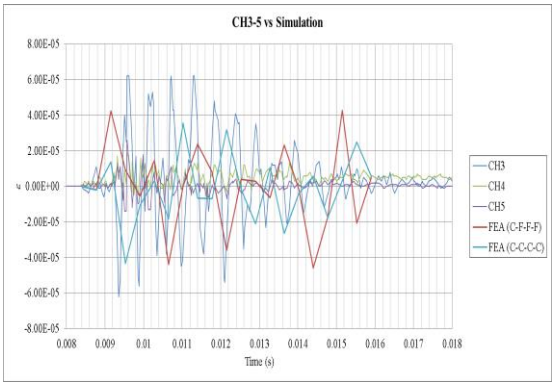


e.

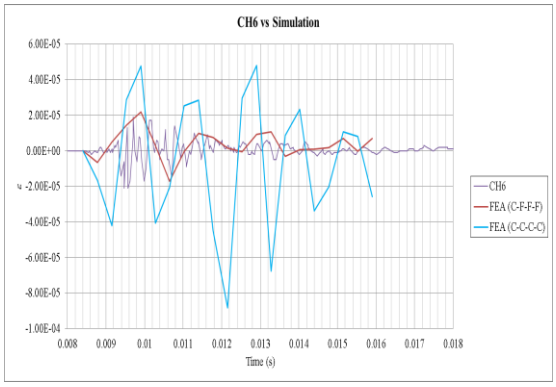


f.

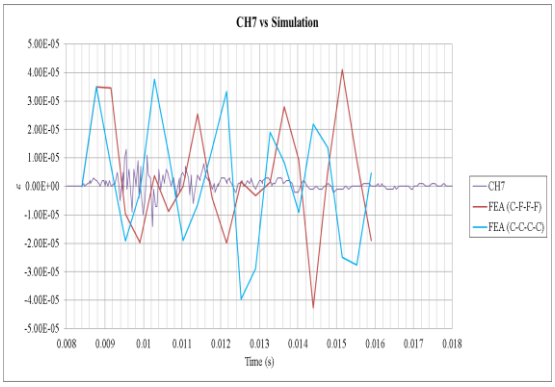
Figure A.156. Failure criterion envelopes, steel penetrator (30 lb), test date: 6 Sep 19



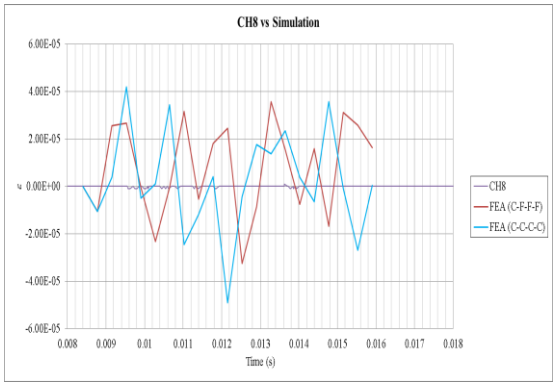
a.



b.



c.



d.

Figure A.157. Strain response and FEA comparison, steel penetrator (30 lb), test date: 6 Sep 19

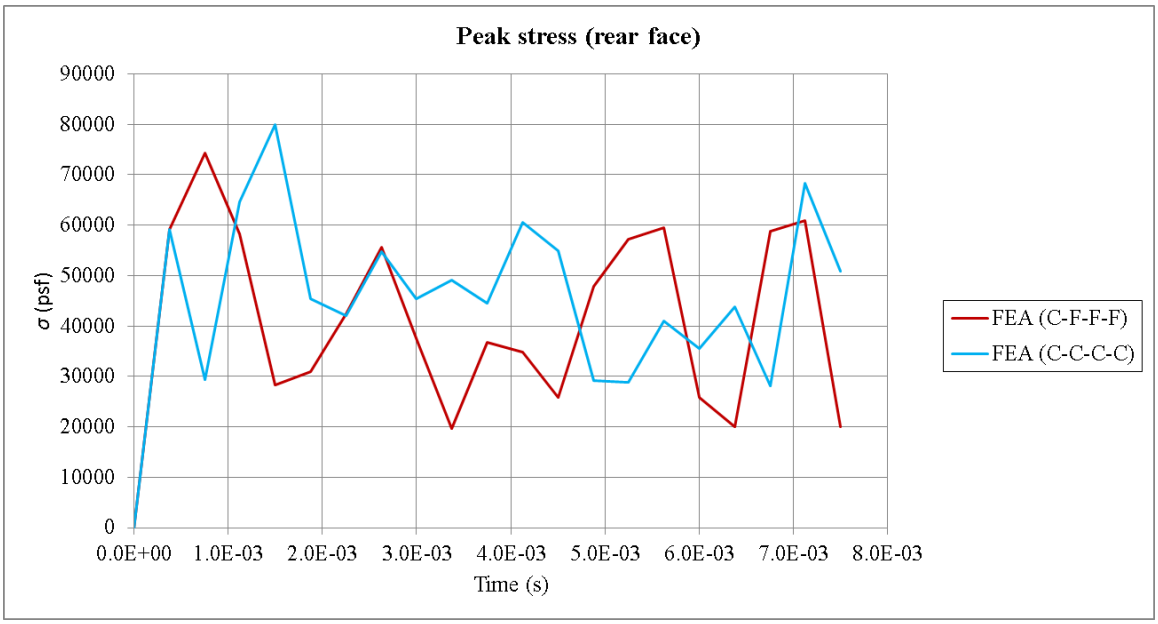
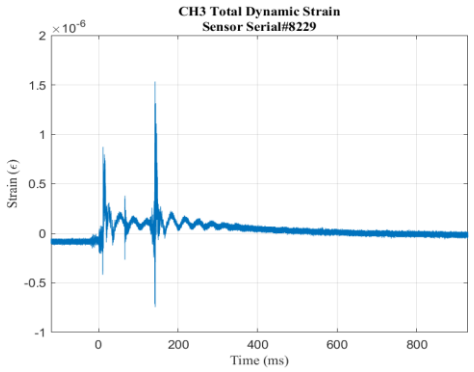


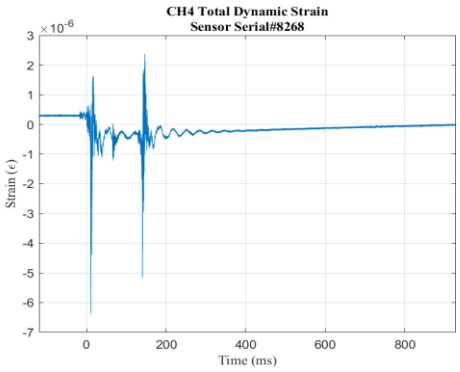
Figure A.158. FEA peak rear face stress, steel penetrator (30 lb), test date: 6 Sep 19

N, Steel rail (35 lb), 19 September 2019

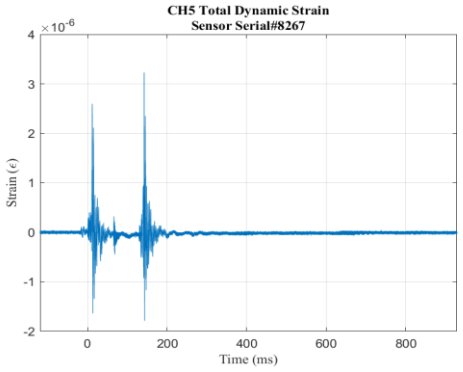
Figure A.159. Steel rail (35 lb)



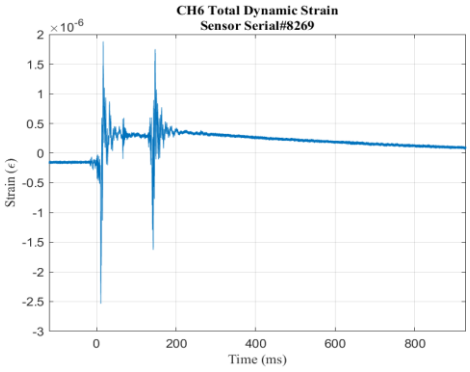
a.



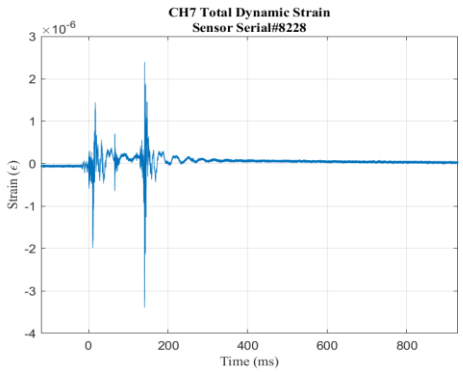
b.



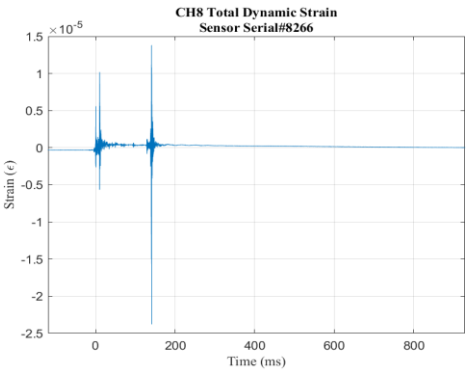
c.



d.



e.



f.

Figure A.160. Strain response, steel rail (35 lb), test date: 19 Sep 19

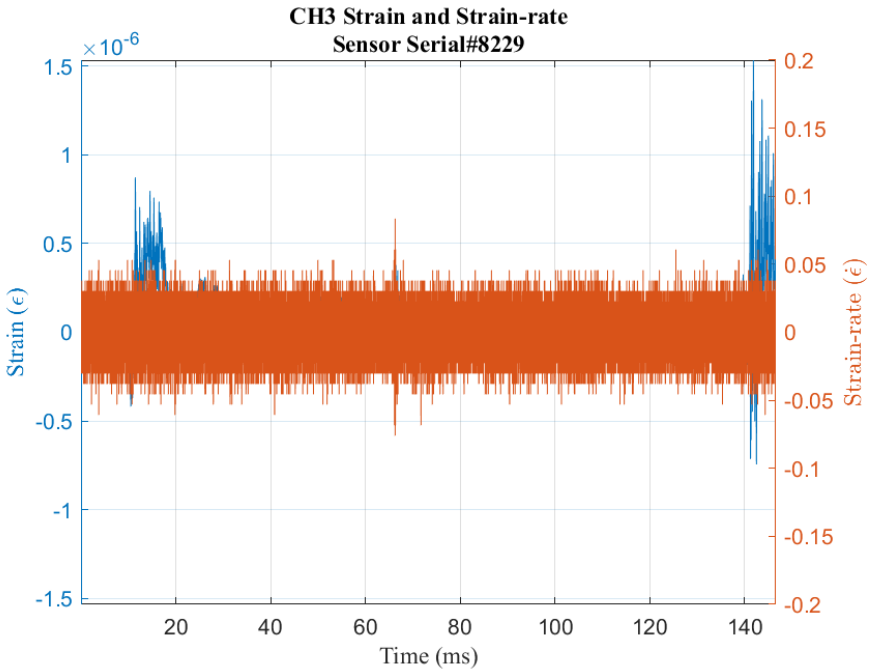


Figure A.161. Strain and strain rate, steel rail (35 lb), CH3

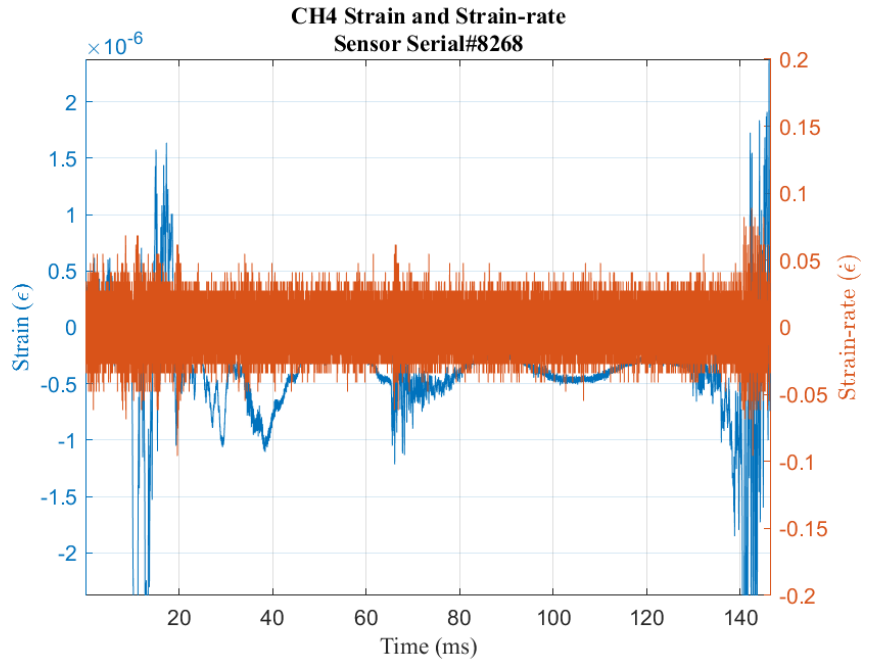


Figure A.162. Strain and strain rate, steel rail (35 lb), CH4

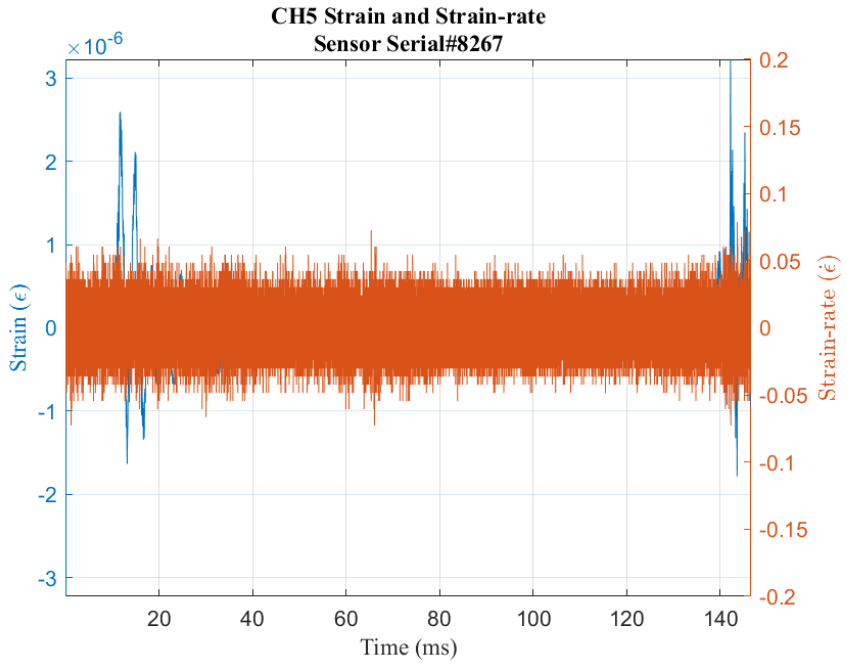


Figure A.163. Strain and strain rate, steel rail (35 lb), CH5

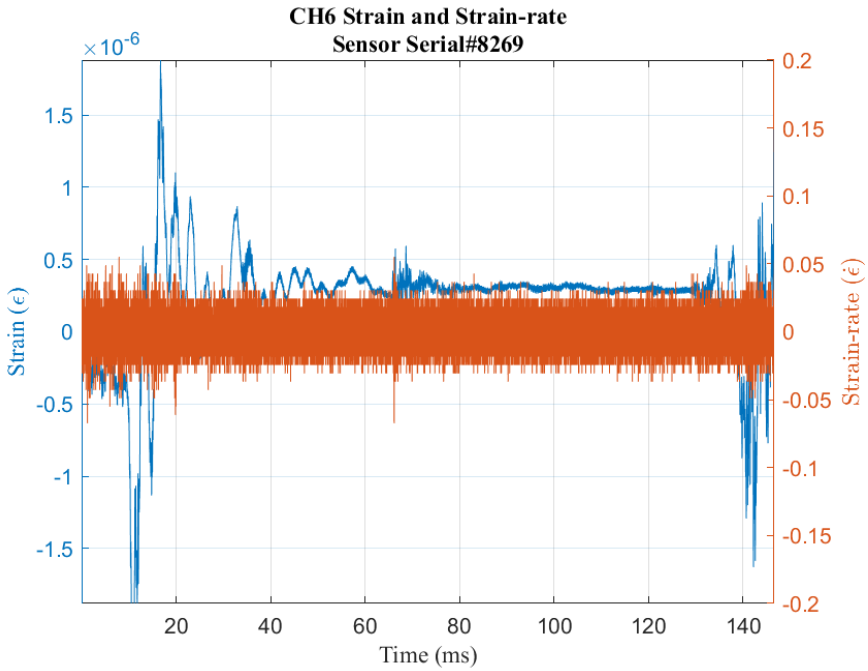


Figure A.164. Strain and strain rate, steel rail (35 lb), CH6

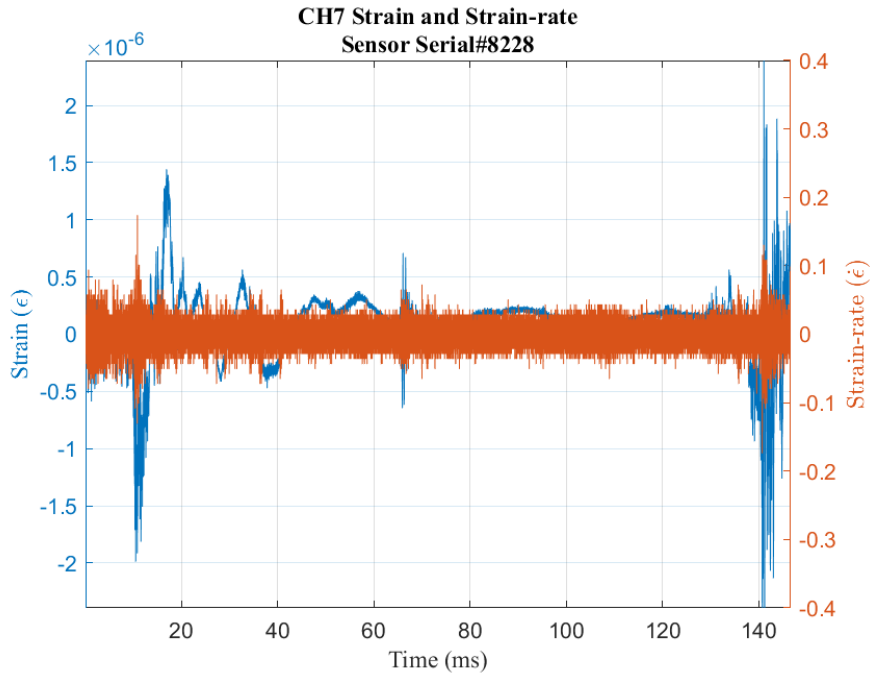


Figure A.165. Strain and strain rate, steel rail (35 lb), CH7

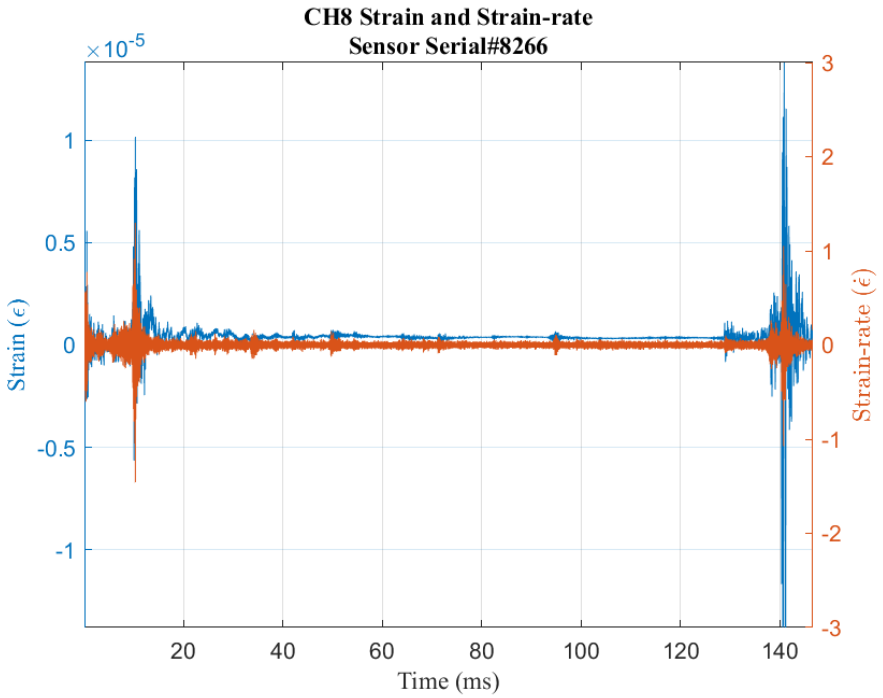
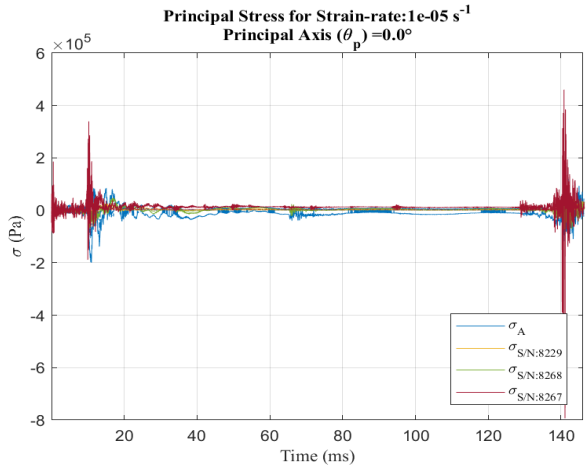
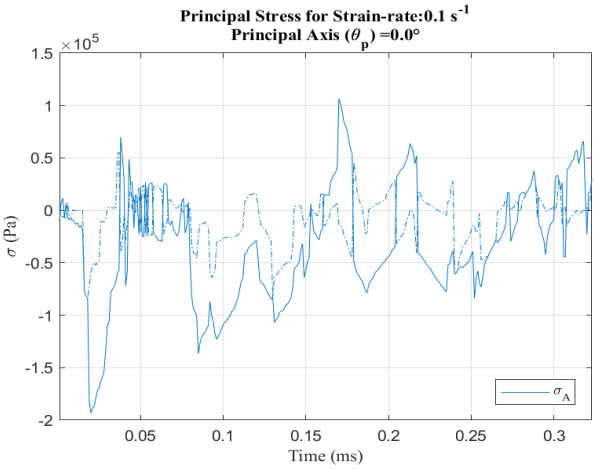


Figure A.166. Strain and strain rate, steel rail (35 lb), CH8

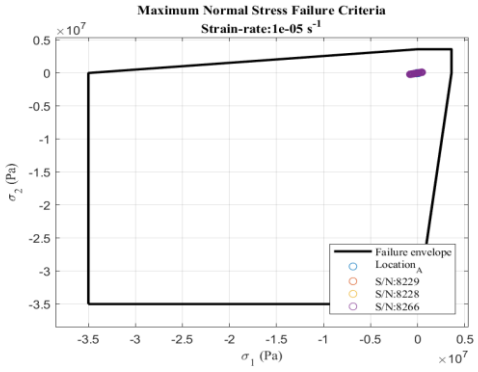


a.

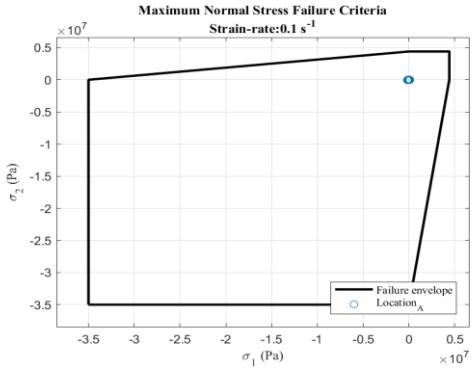


b.

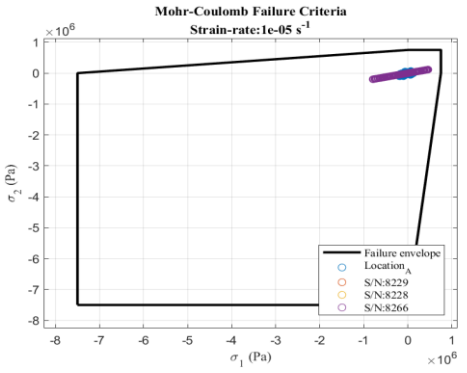
Figure A.167. Principal stress at strain rate, steel rail (35 lb), test date: 19 Sep 19



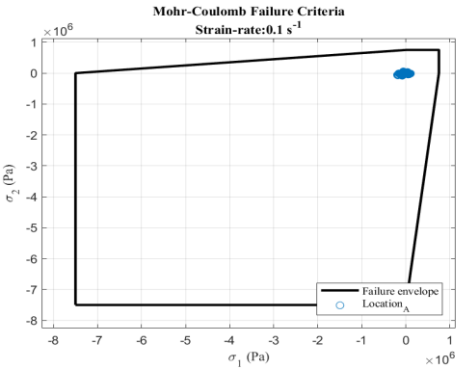
a.



b.

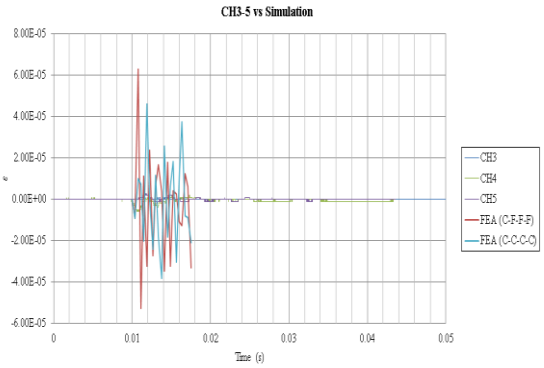


c.

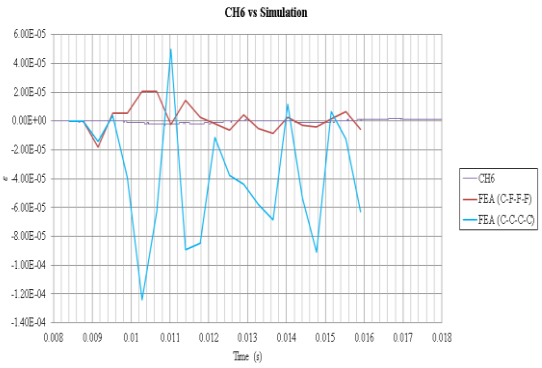


d.

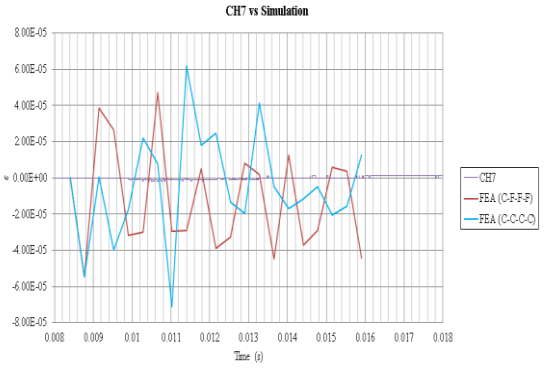
Figure A.168. Failure criterion envelopes, steel rail (35 lb), test date: 19 Sep 19



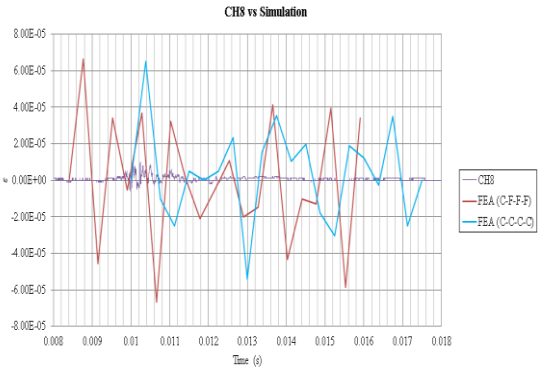
a.



b.



c.



d.

Figure A.169. Strain response and FEA comparison, steel rail (35 lb), test date: 19 Sep 19

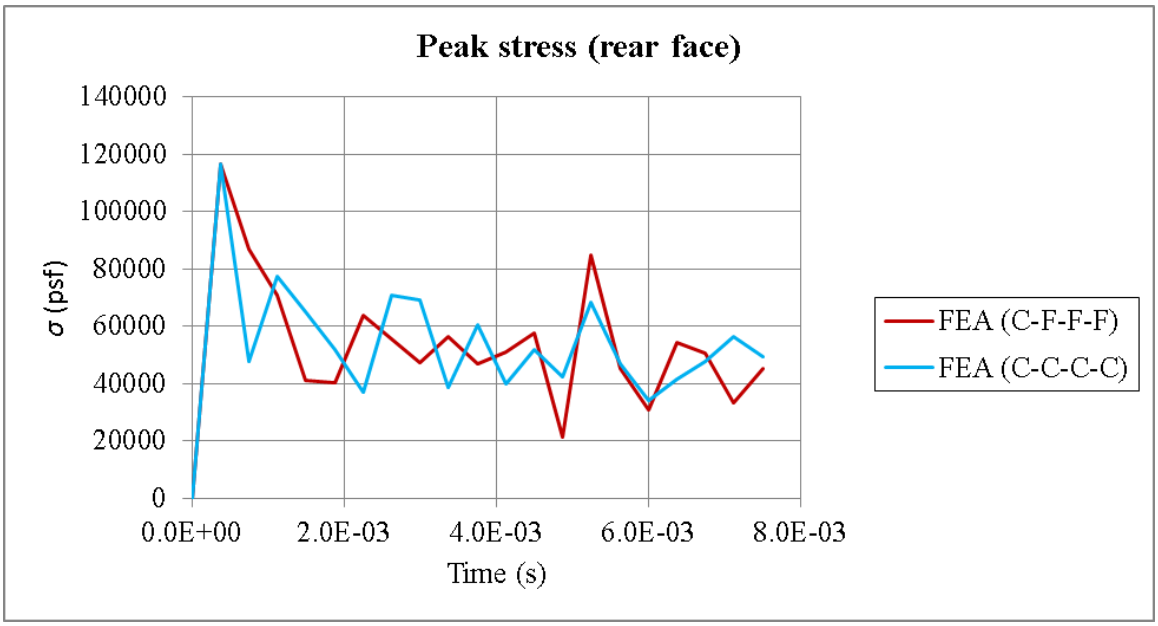
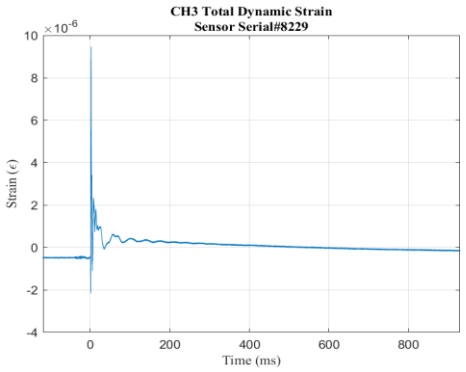


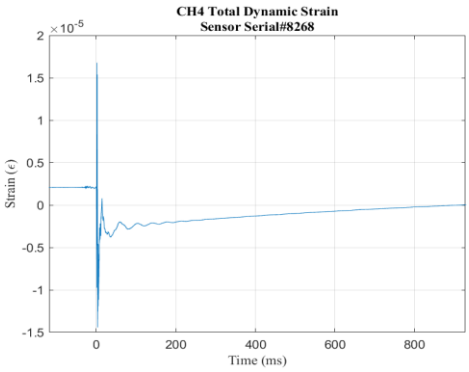
Figure A.170. FEA peak rear face stress, steel rail (35 lb), test date: 19 Sep 19

O, Steel penetrator (20 lb), 19 September 2019

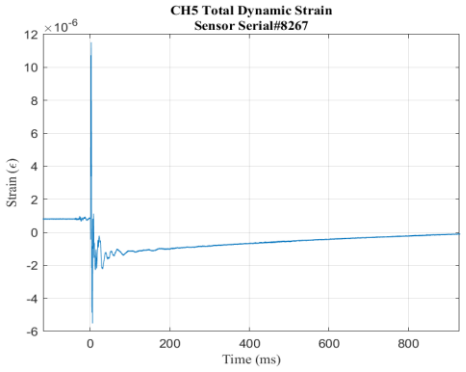
Figure A.171. Steel penetrator (20 lb), test date: 19 Sep 19



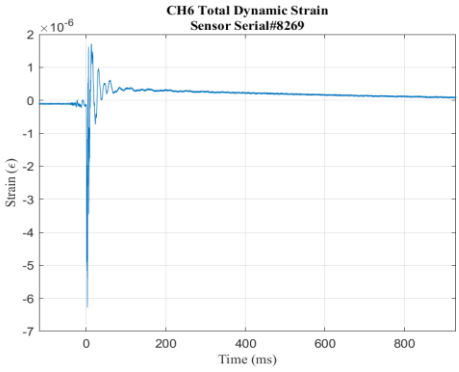
a.



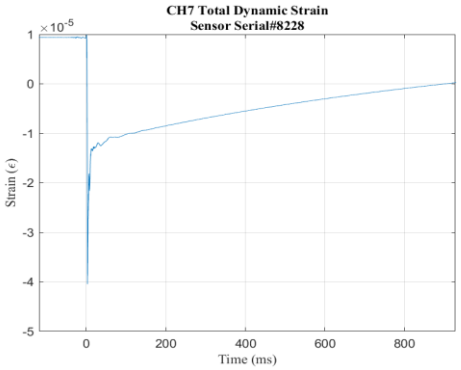
b.



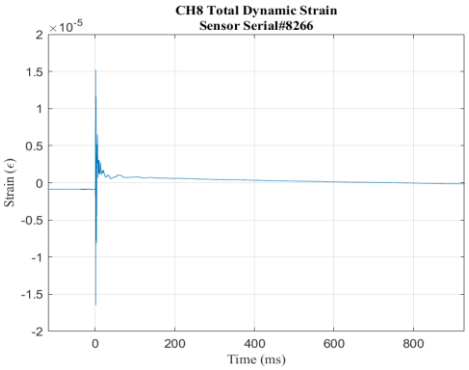
c.



d.



e.



f.

Figure A.172. Strain response, steel penetrator (20 lb), test date: 19 Sep 19

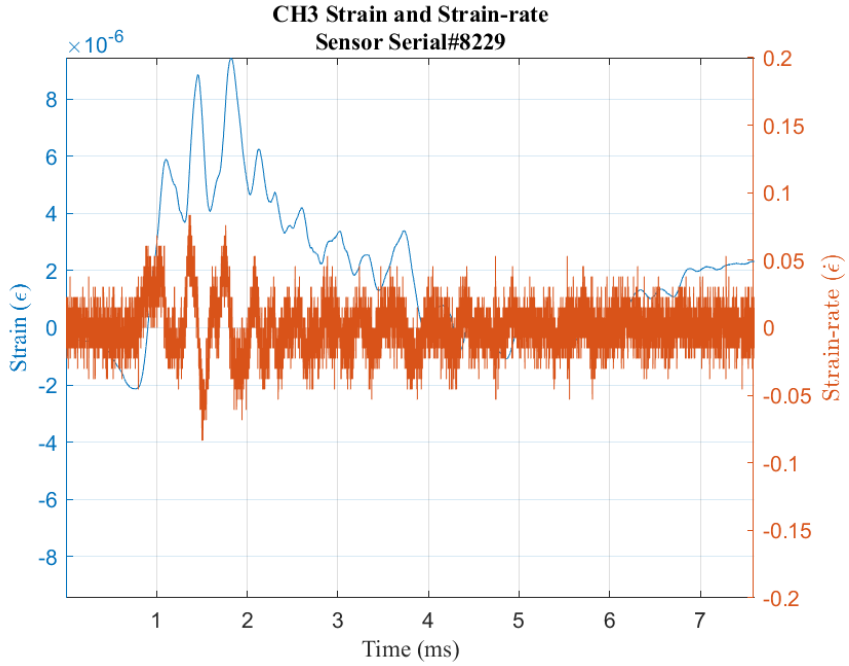


Figure A.173. Strain and strain rate, steel penetrator (20 lb), 19 Sep 19, CH3

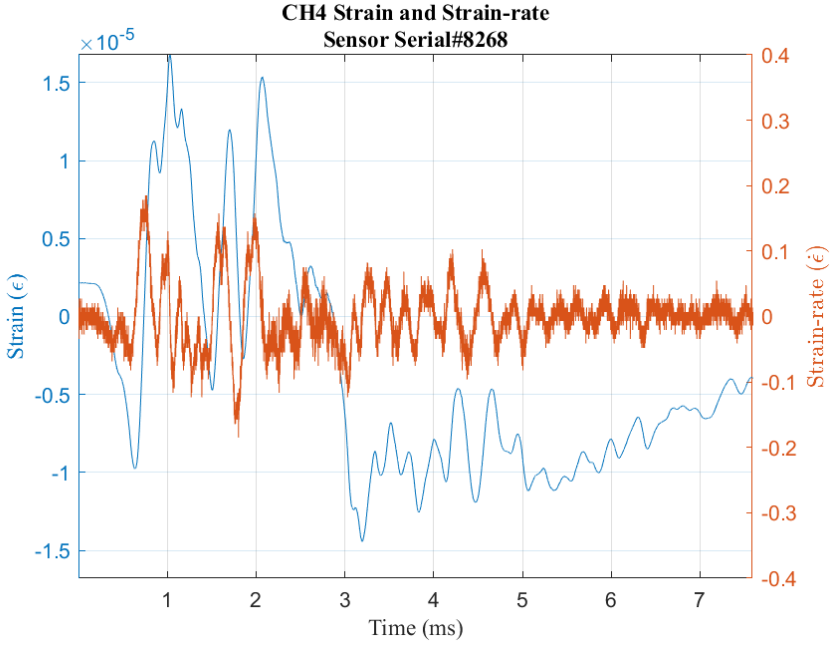


Figure A.174. Strain and strain rate, steel penetrator (20 lb), 19 Sep 19, CH4

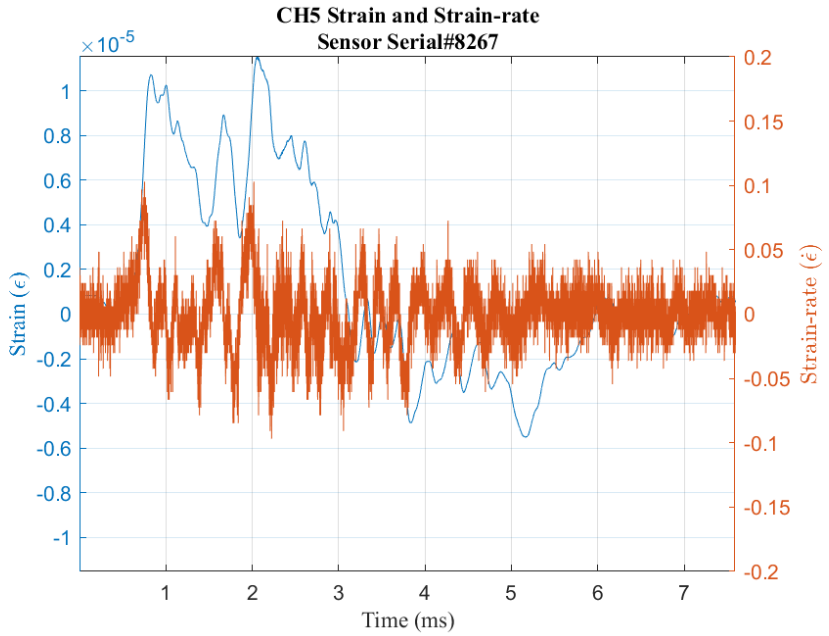


Figure A.175. Strain and strain rate, steel penetrator (20 lb), 19 Sep 19, CH5

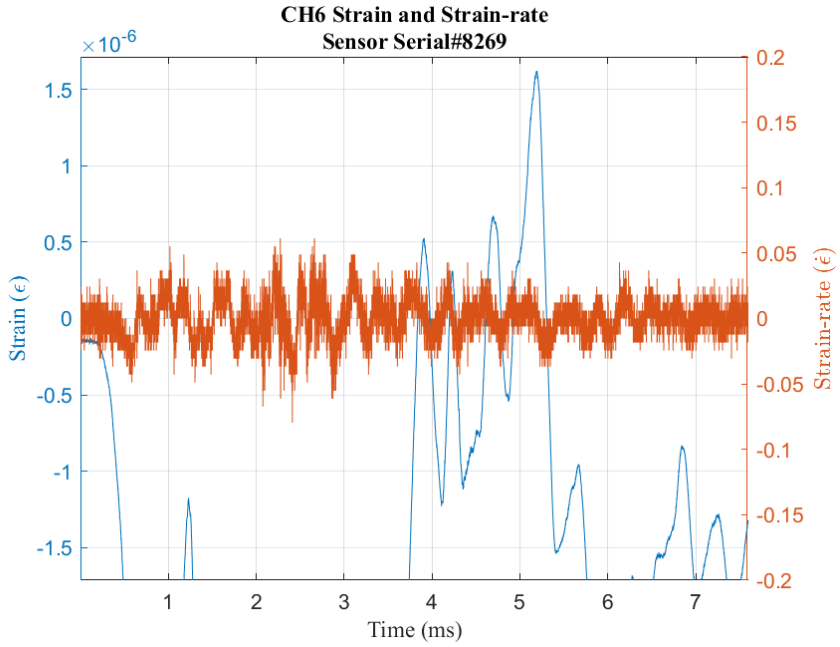


Figure A.176. Strain and strain rate, steel penetrator (20 lb), 19 Sep 19, CH6

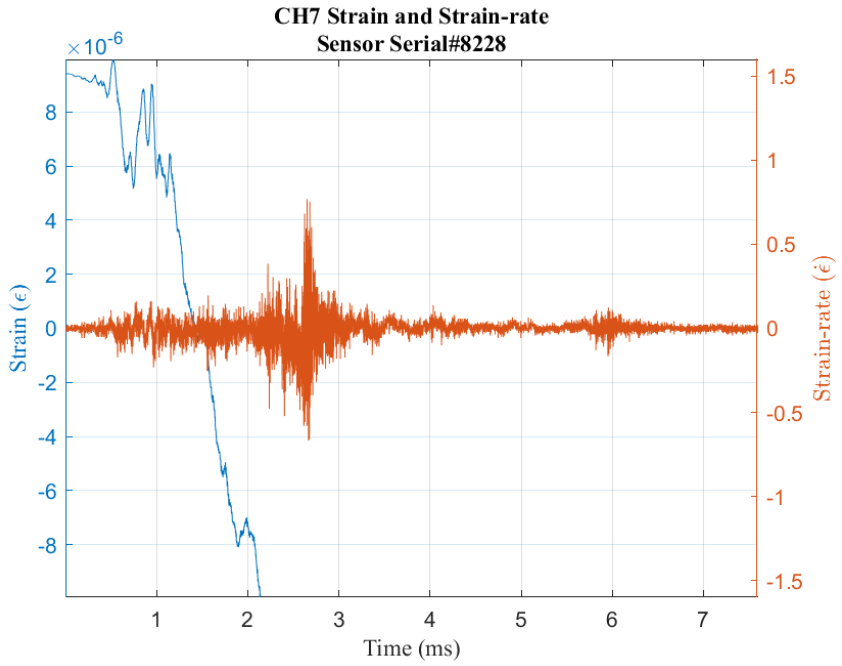


Figure A.177. Strain and strain rate, steel penetrator (20 lb), 19 Sep 19, CH7

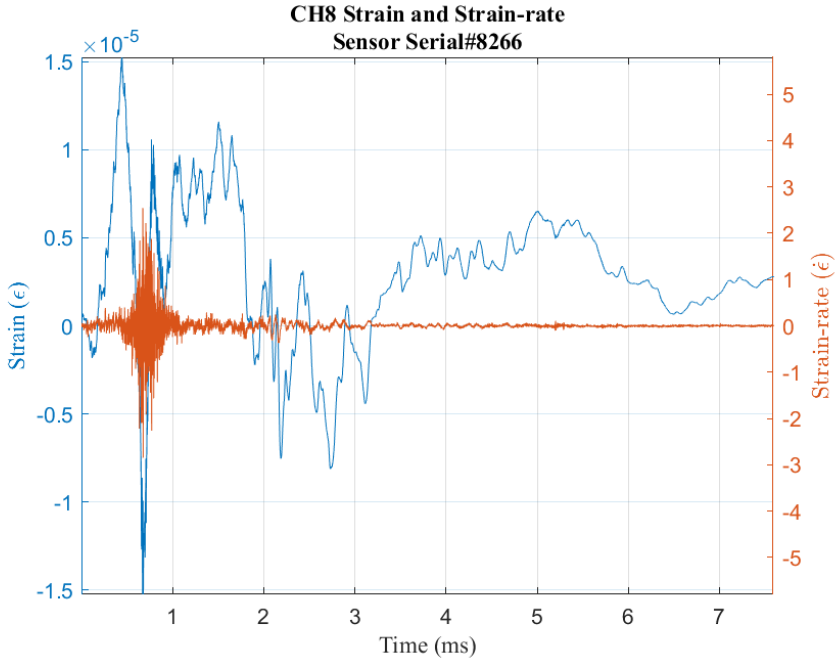
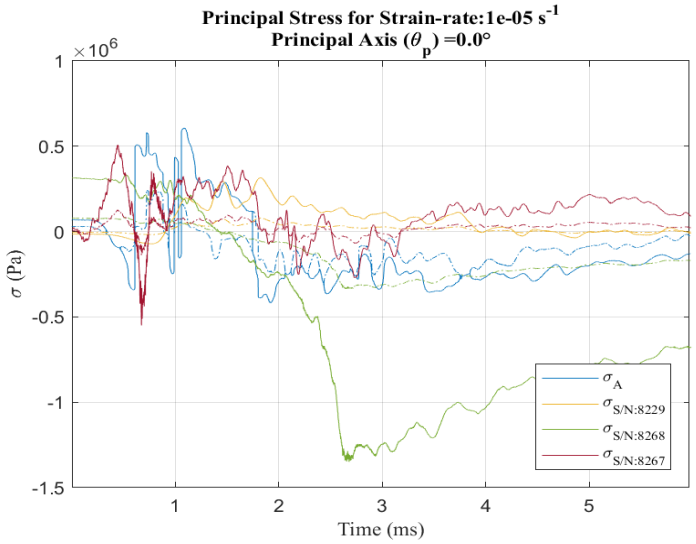
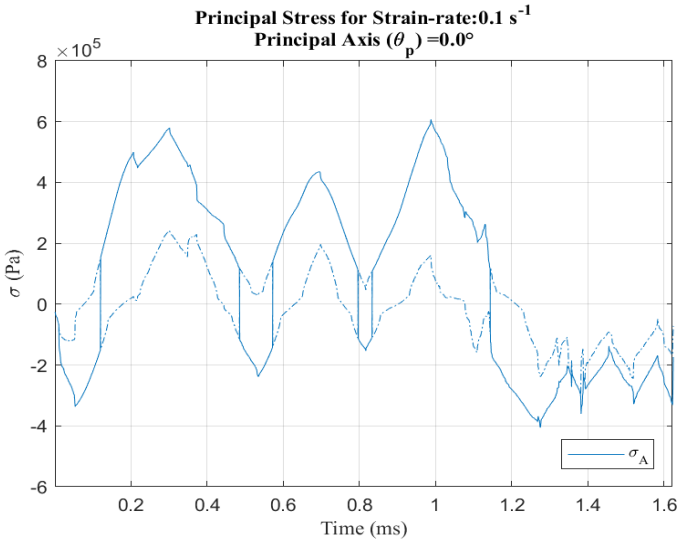


Figure A.178. Strain and strain rate, steel penetrator (20 lb), 19 Sep 19, CH8

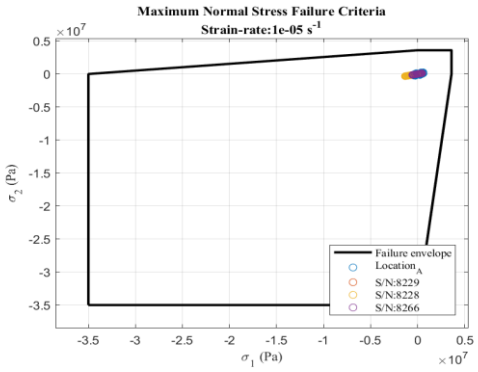


a.

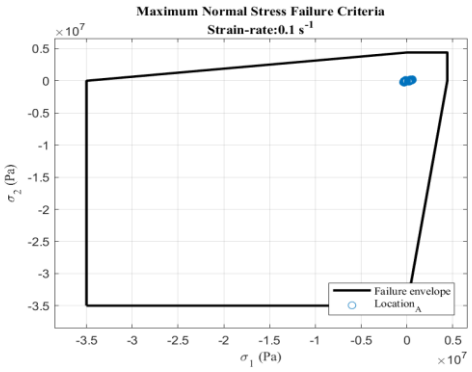


b.

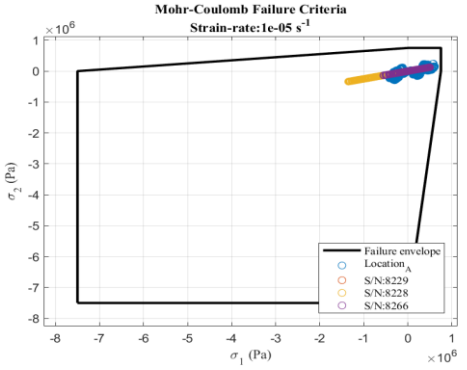
Figure A.179. Principal stress at strain rate, steel penetrator (20 lb), test date: 19 Sep 19



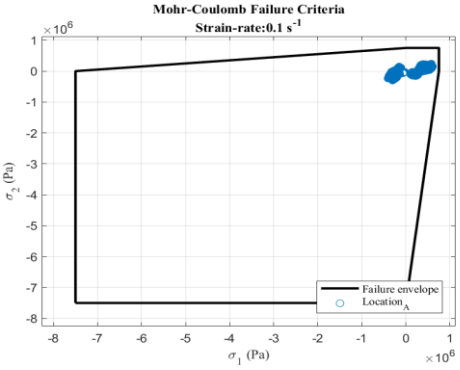
a.



b.

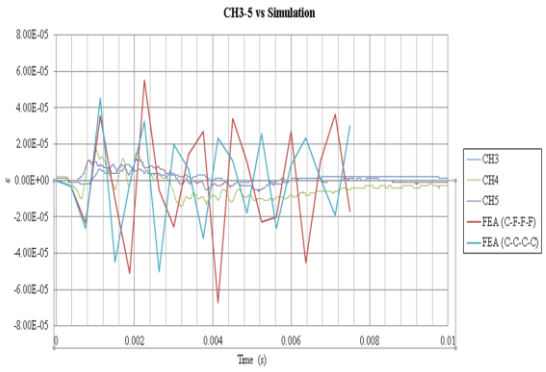


c.

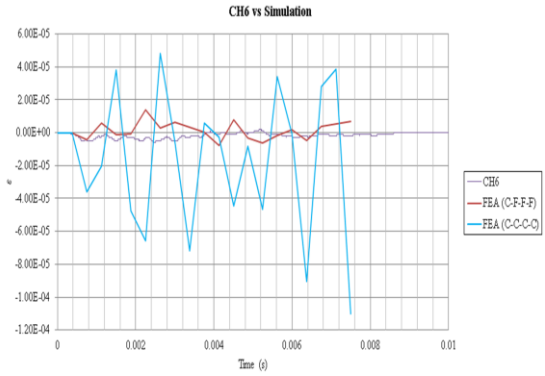


d.

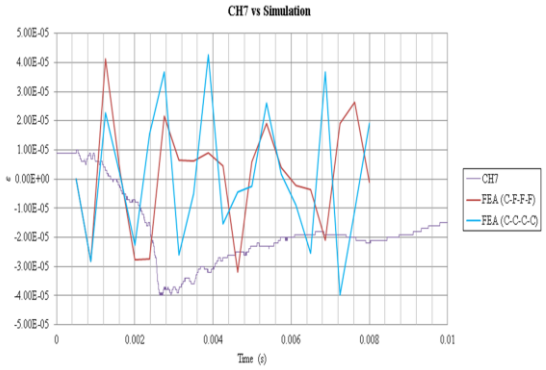
Figure A.180. Failure criterion envelopes, steel penetrator (20 lb), test date: 19 Sep 19



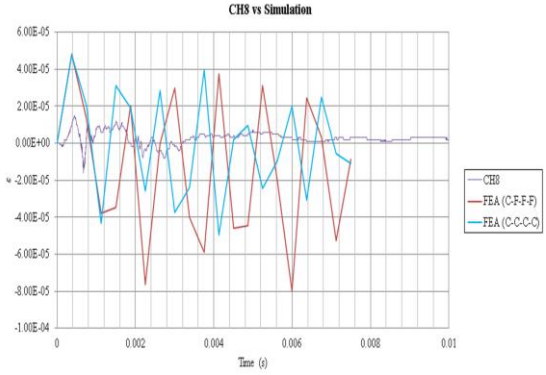
a.



b.



c.



d.

Figure A.181. Strain response and FEA comparison, steel penetrator (20 lb), test date: 19 Sep 19

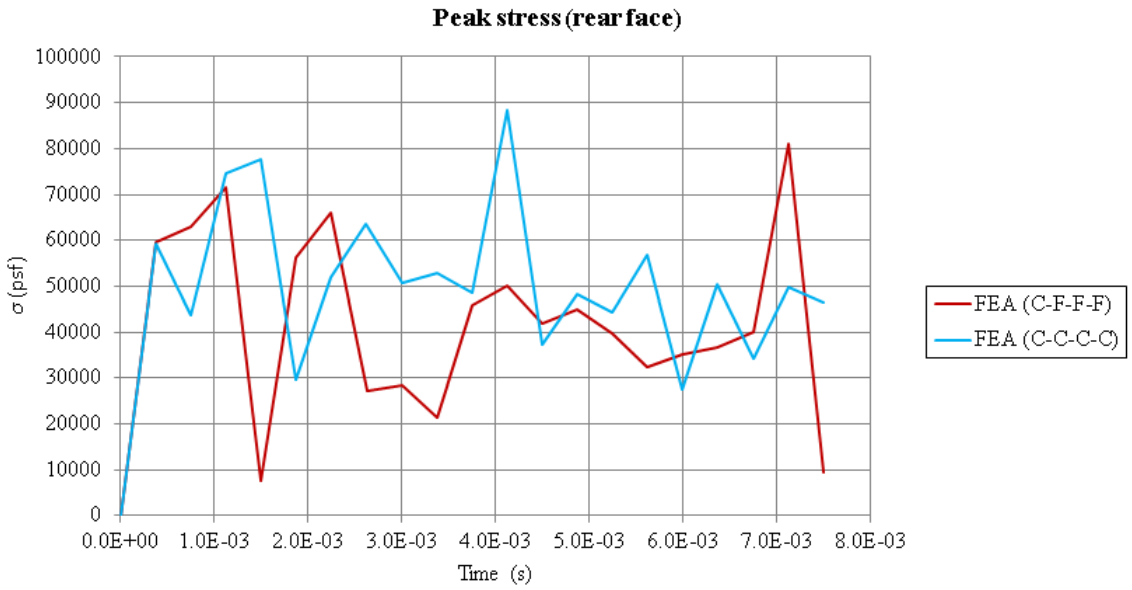
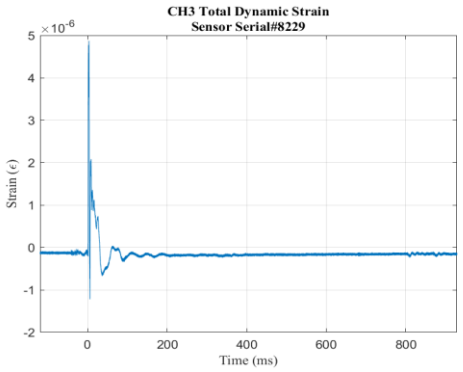


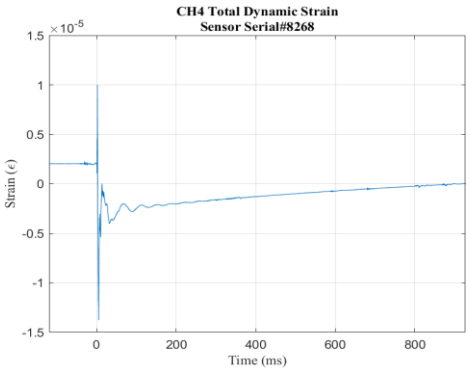
Figure A.182. FEA peak rear face stress, steel penetrator (20 lb), test date: 19 Sep 19

P, Steel penetrator (30 lb), 19 September 2019

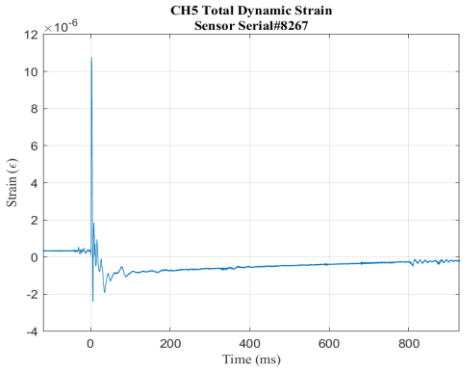
Figure A.183. Steel penetrator (30 lb), test date: 19 Sep 19



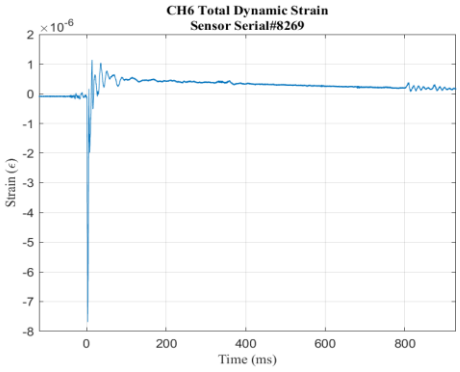
a.



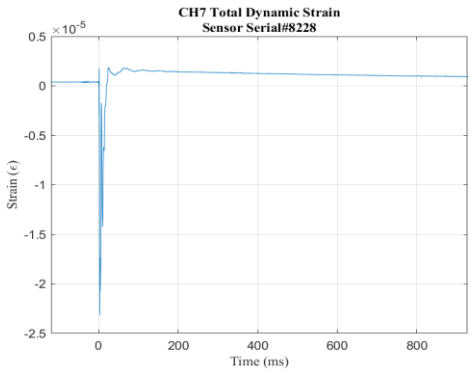
b.



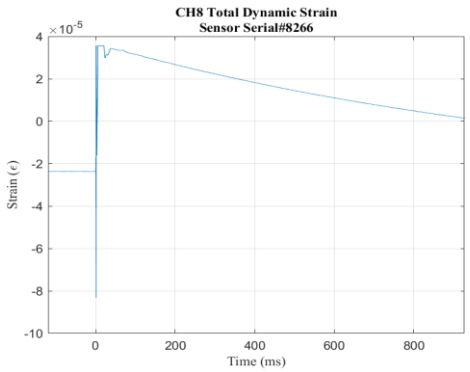
c.



d.



e.



f.

Figure A.184. Strain response, steel penetrator (30 lb), test date: 19 Sep 19

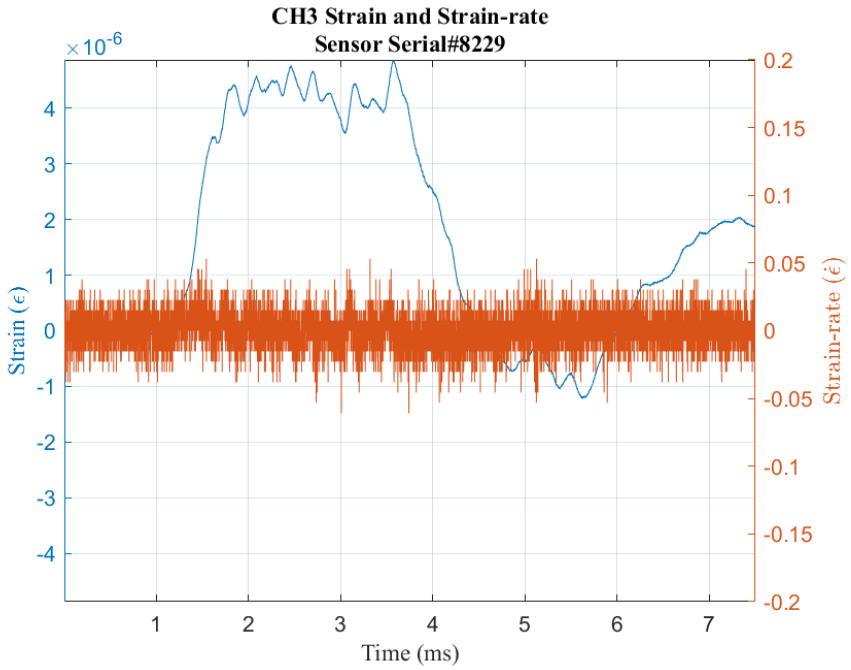


Figure A.185. Strain and strain rate, steel penetrator (30 lb), 19 Sep 19, CH3

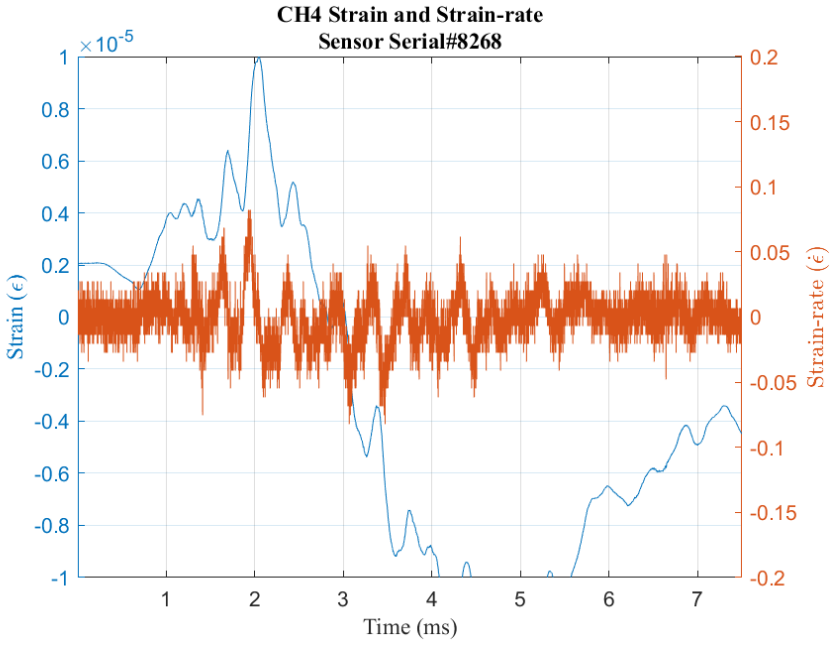


Figure A.186. Strain and strain rate, steel penetrator (30 lb), 19 Sep 19, CH4

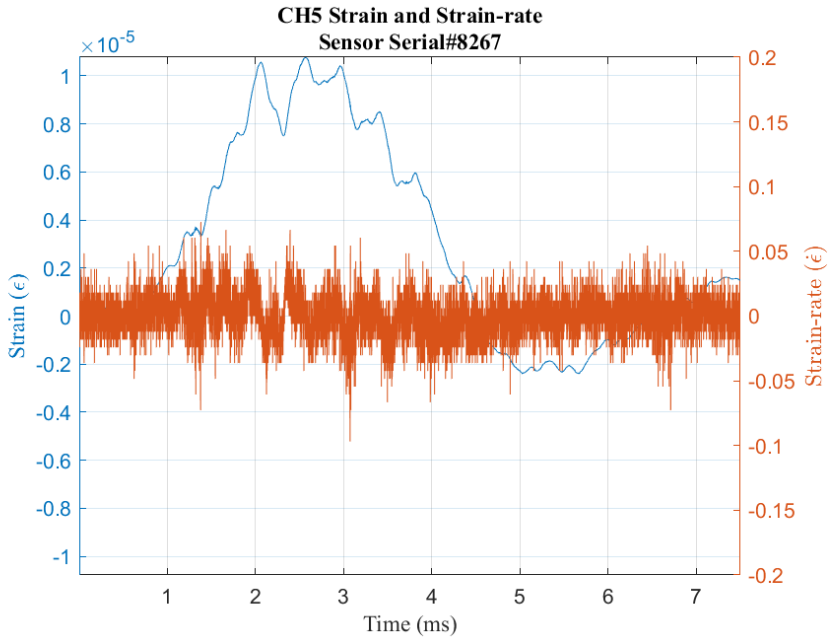


Figure A.187. Strain and strain rate, steel penetrator (30 lb), 19 Sep 19, CH5

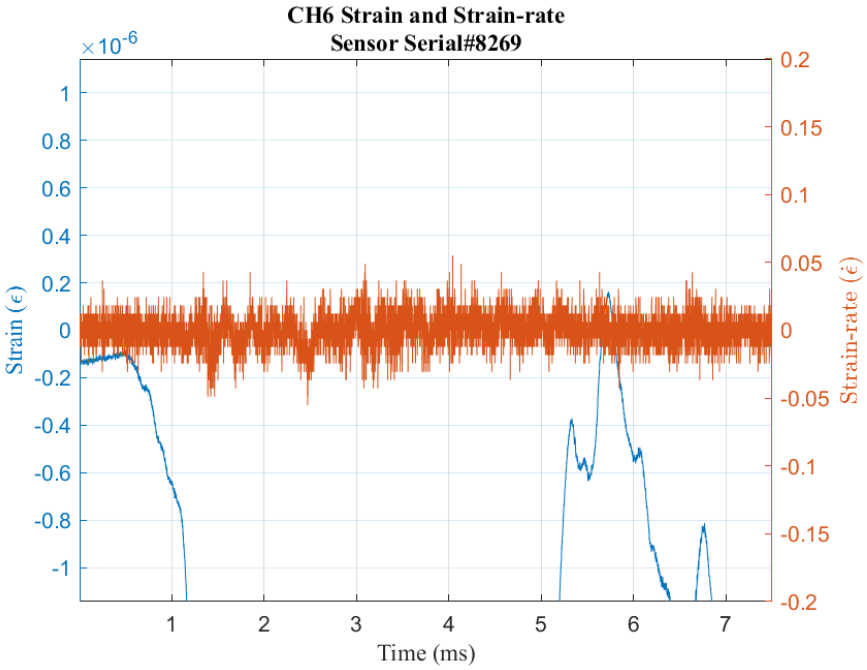


Figure A.188. Strain and strain rate, steel penetrator (30 lb), 19 Sep 19, CH6

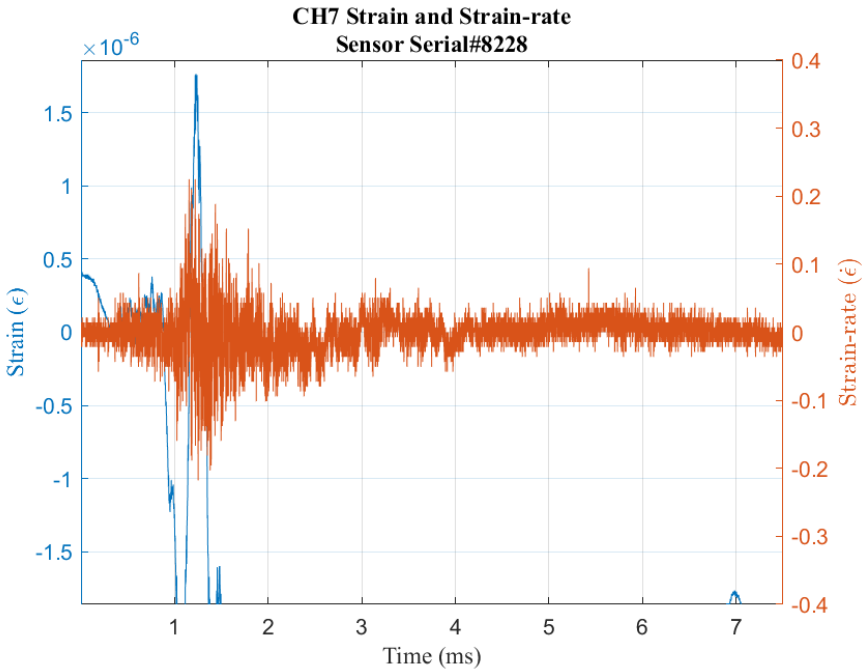


Figure A.189. Strain and strain rate, steel penetrator (30 lb), 19 Sep 19, CH7

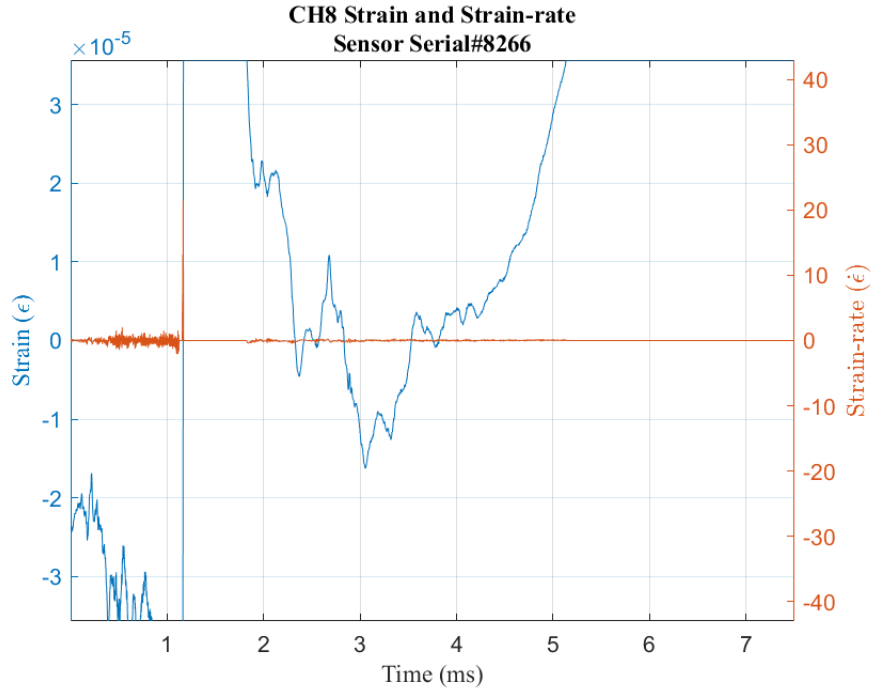
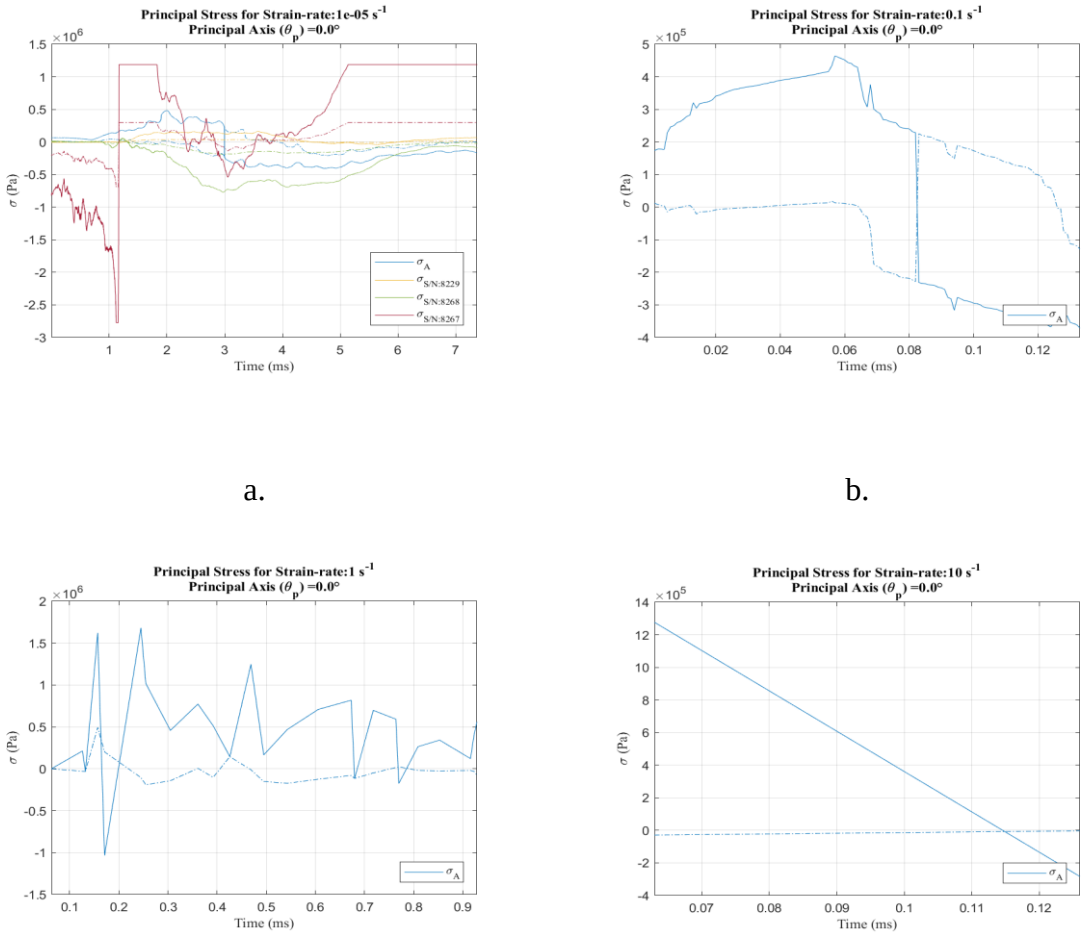


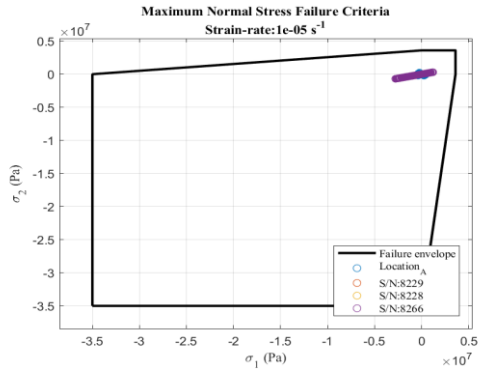
Figure A.190. Strain and strain rate, steel penetrator (30 lb), 19 Sep 19, CH8



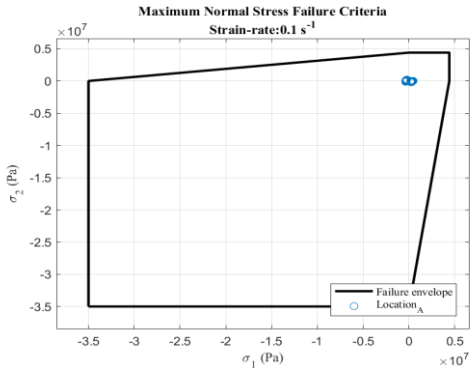
a. b.

c. d.

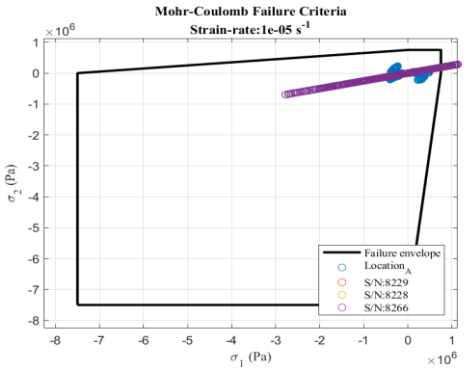
Figure A.191. Principal stress at strain rate, steel penetrator (30 lb), test date: 19 Sep 19



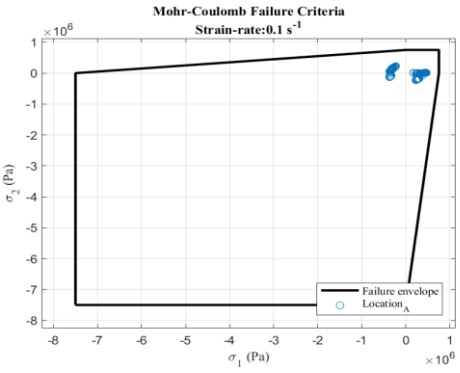
a.



b.



c.



d.

Figure A.192. Failure criterion envelopes, steel penetrator (30 lb), test date: 19 Sep 19

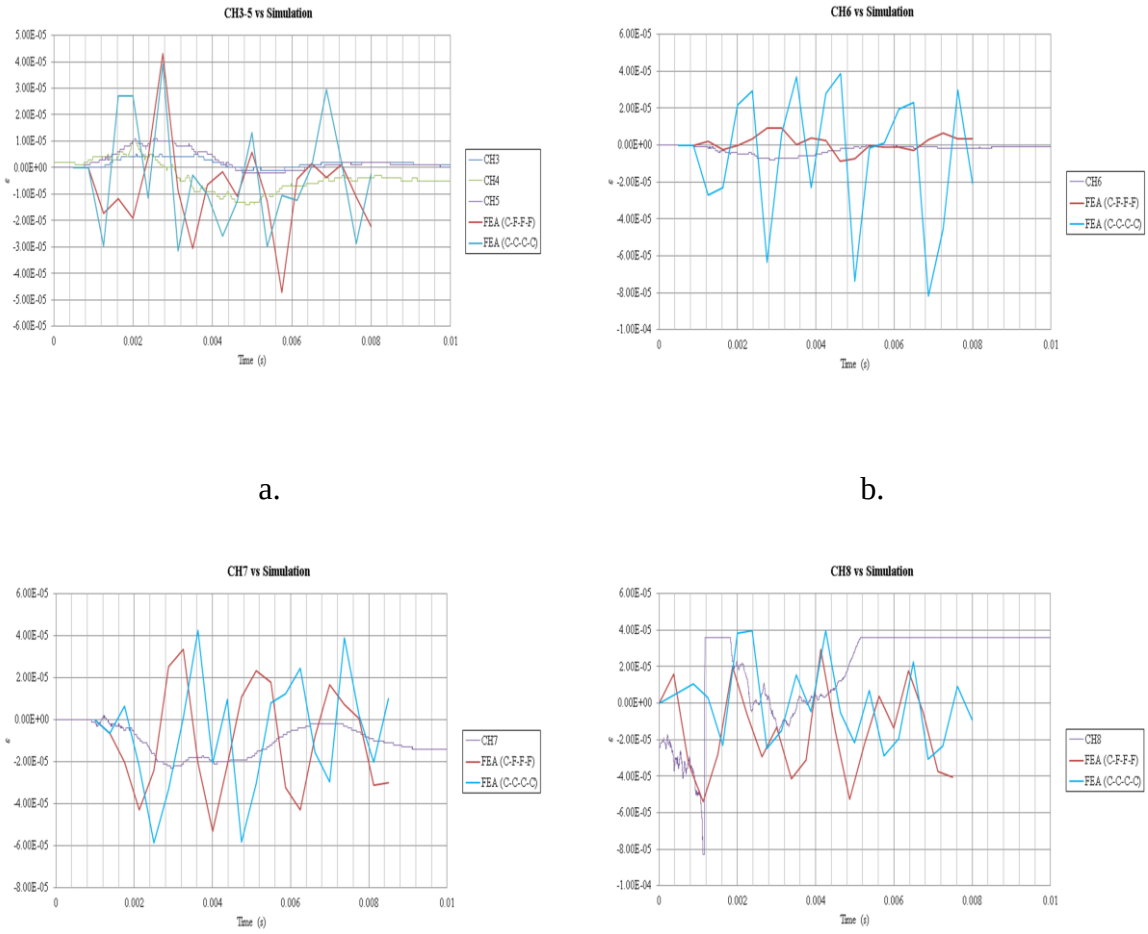


Figure A.193. Strain response and FEA comparison, steel penetrator (30 lb), test date: 6 Sep 19

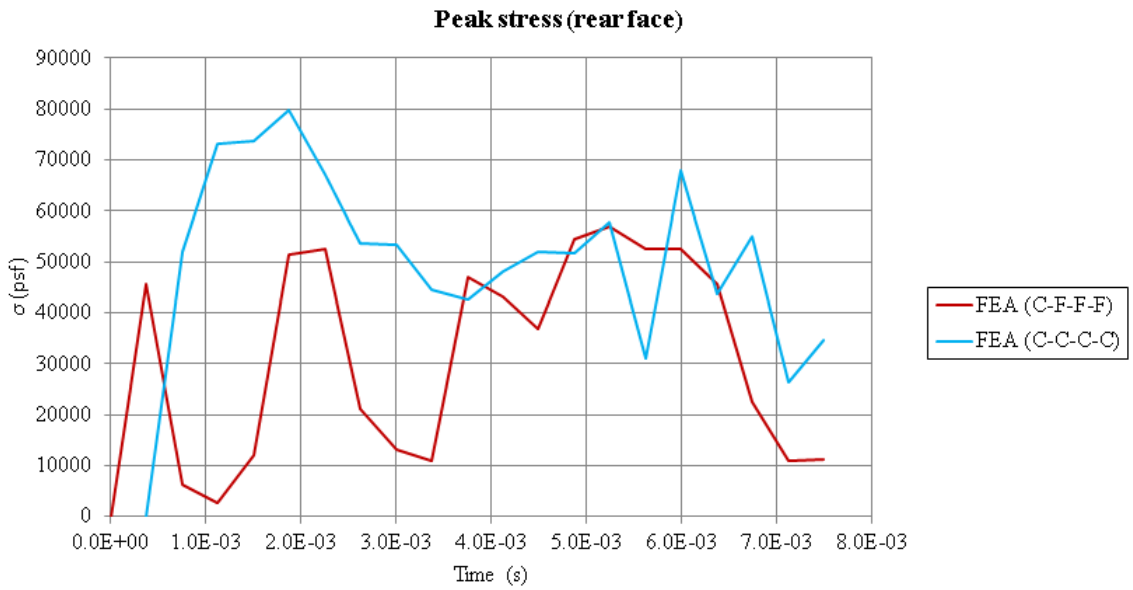


Figure A.194. FEA peak rear face stress, steel penetrator (30 lb), test date: 19 Sep 19

Consolidated test data for comparison

The following figures are a consolidation of penetration and peak values for each test case and the estimated impact energy or modified Weber number.

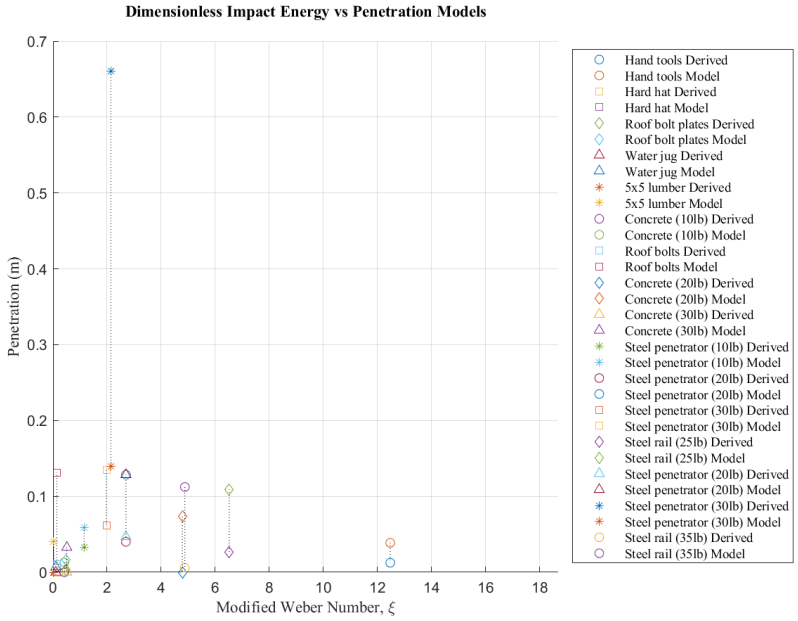


Figure A.195. Dimensionless impact energy for estimated and derived penetration

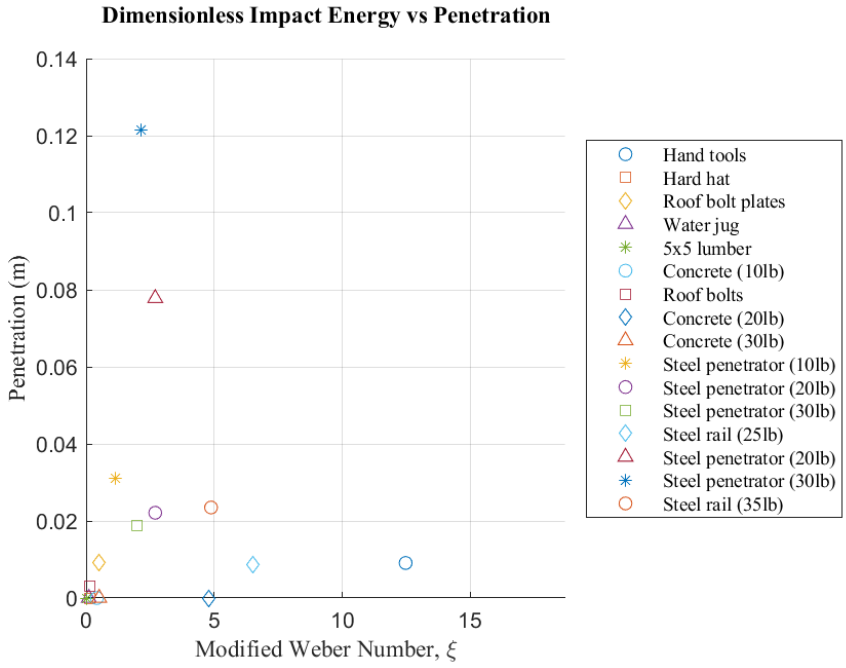


Figure A.196. Dimensionless impact energy and measured penetration

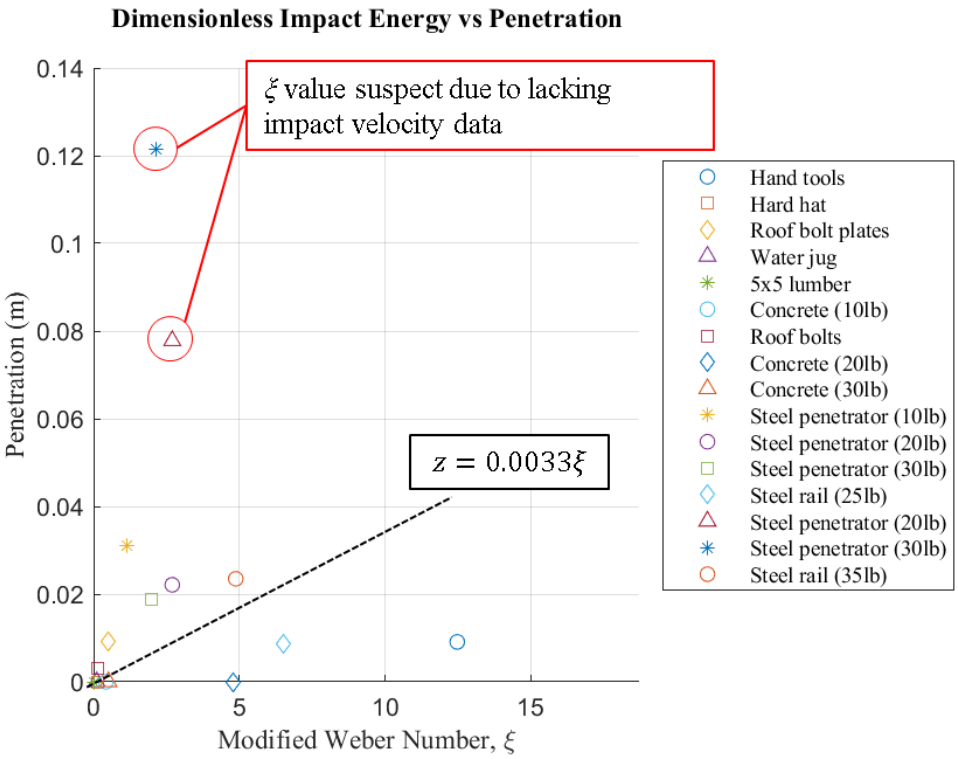


Figure A.197. Dimensionless impact energy and penetration trend

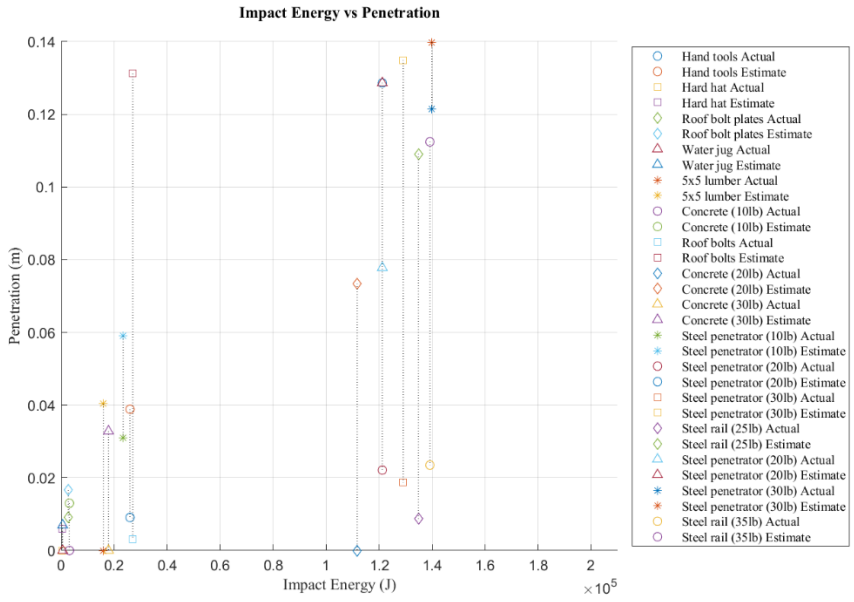


Figure A.198. Impact energy and models for estimated and volume loss derived penetration

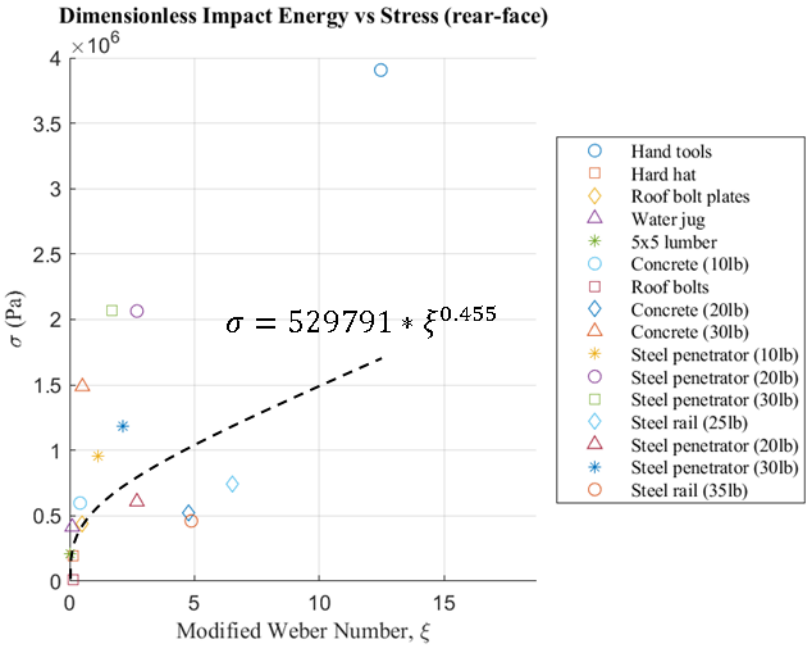


Figure A.199. Dimensionless impact energy and peak rear face stress

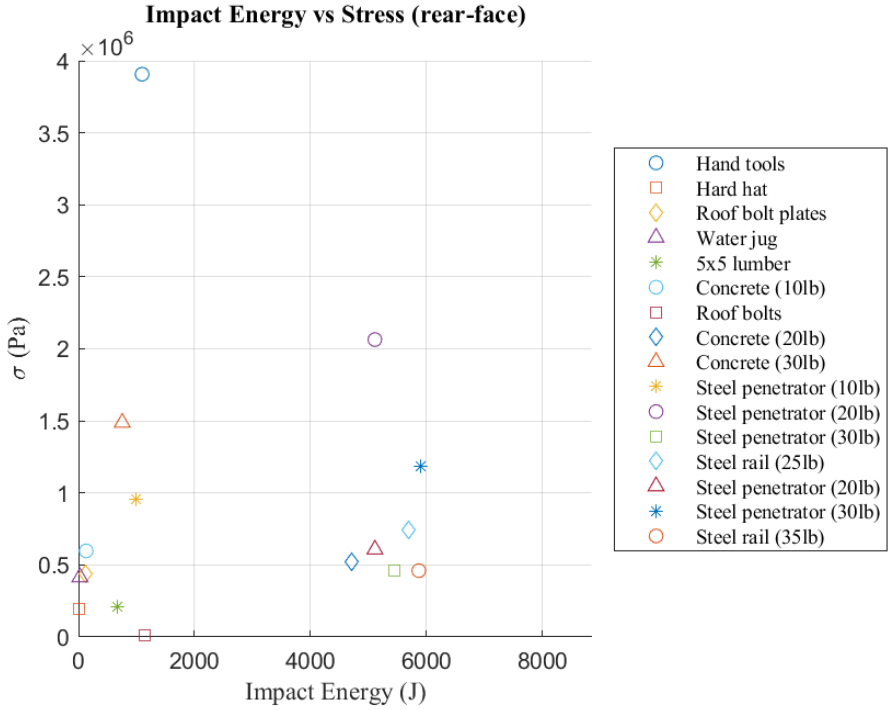


Figure A.200. Impact energy and peak rear face stress

APPENDIX B.

DATA PROCESSING SCRIPTS

The scripts developed for processing the strain sensor data are given in this appendix. The script flow follows the path depicted in Figure B.1. The raw channel data files are stored in the test case directory as tab delimited text files.

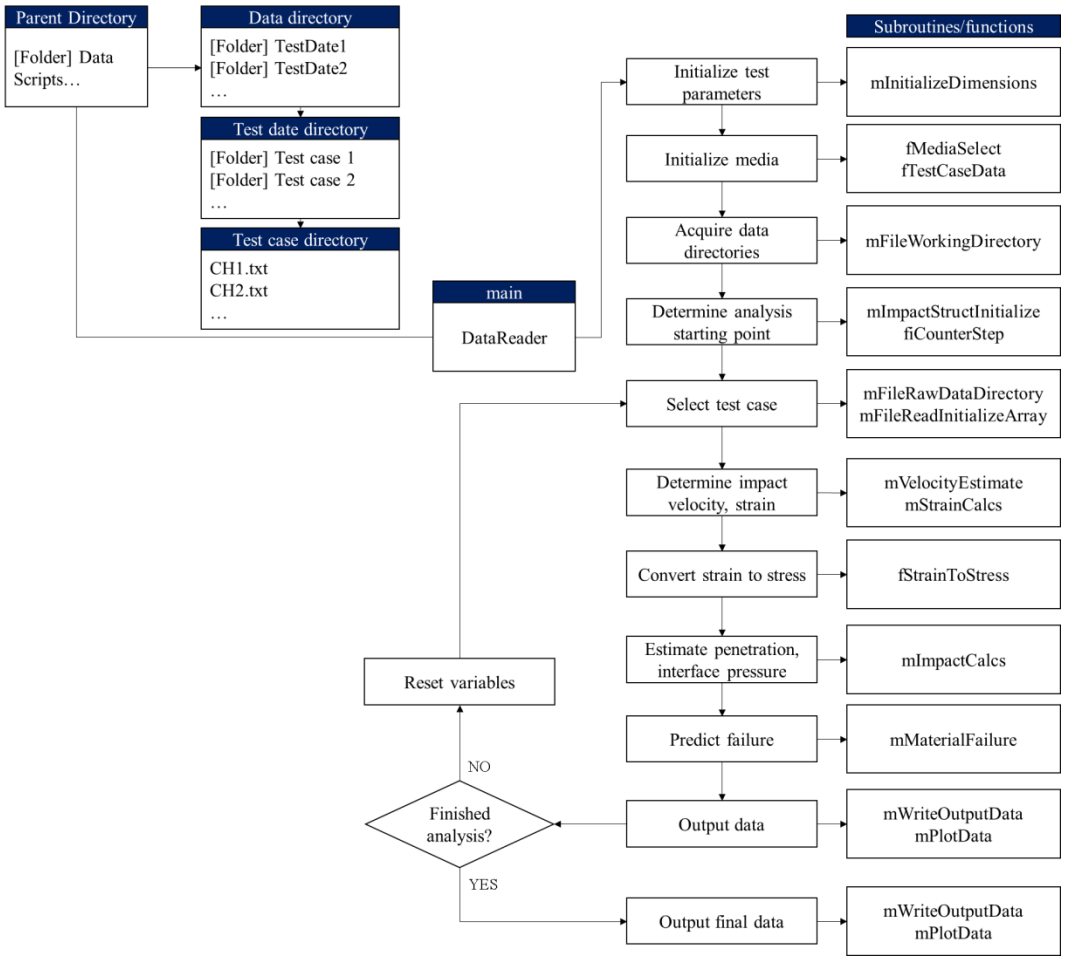


Figure B.1. Processing script flow diagram

Table B.1. Processing script output

Case	Hand tools	Hard hat	Roof bolt plates	Water jug	5x5 lumber	Concrete (10lb)	Roof bolts	Concrete (20lb)	Concrete (30lb)	Steel penetrator (10lb)	Steel penetrator (20lb)	Steel penetrator (30lb)	Steel rail (25lb)	Steel penetrator (20lb)	Steel penetrator (30lb)	Steel rail (35lb)
Number of projectiles	3	1	5	1	1	1	4	1	1	1	1	1	1	1	1	1
Impact energy (J)	26012.39	439.836	2691.688	601.53	15960.69	3140.296	27038.96	111766.1	17890.09	23427.234	121229.9	129113.1	134867.3	121229.9	139953.3	139200.1
Dimensionless impact energy	12.473	0.137	0.492	0.105	0.025	0.42	0.143	4.783	0.51	1.147	2.697	1.975	6.521	2.697	2.141	4.877
Measured velocity (m/s)	16.337	44.038	58.232	51.5	48.434	37.211	Inf	29.456	51.278	101.635	247.403	123.301	-101.716	Inf	Inf	Inf
Calculated velocity (m/s)	195.529	44.038	58.232	51.5	48.434	37.211	167.488	156.972	51.278	101.635	155.875	133.38	154.229	155.875	138.867	133.38
Projectile mass (kg)	1.361	0.454	1.588	0.454	13.608	4.536	1.928	9.072	13.608	4.536	9.979	14.515	11.34	9.979	14.515	15.649
Cumulative projectile mass (kg)	4.082	0.454	7.938	0.454	13.608	4.536	7.711	9.072	13.608	4.536	9.979	14.515	11.34	9.979	14.515	15.649
Penetration Estimate (m)	0.039	0.006	0.017	0.007	0.04	0.013	0.131	0.073	0.033	0.059	0.129	0.135	0.109	0.129	0.14	0.112
Penetration Derived (m)	0.013	0	0.003	0	0	0	0.011	0	0	0.033	0.04	0.062	0.026	0.047	0.661	0.006
Penetration Actual (m)	0.009	0	0.009	0	0	0	0.003	0	0	0.031	0.022	0.019	0.009	0.078	0.121	0.023
Volume loss (10^-3 m^3)	0.17	0	0.059	0	0	0	0.011	0	0	0.669	0.167	0.404	0.018	1.346	71.302	0.052
Peak rear-face stress (MPa)	3.9	0.2	0.4	0.4	0.2	0.6	0.0	0.5	1.5	1.0	2.1	0.5	0.7	0.6	1.2	0.5
Peak rear-face strain (1e-6)	56.116	5.781	13.208	12.567	4.991	17.919	0.24	16.407	27.546	12.857	61.938	13.813	14.737	16.775	35.637	13.813
Peak rear-face strain-rate (s^-1)	1.159	0.433	0.521	0.663	0.126	0.33	1.383	0.708	0.889	0.212	1.136	1.458	0.538	2.851	21.548	1.458
Safety Factor – Penetration	-0.883	1.327	-0.593	0.983	1.41	0.052	-0.862	-0.756	-0.239	-0.769	-0.852	-0.826	-0.85	-0.852	-0.833	-0.826
Safety Factor - Rear-face cracking	0.894	21.369	15.893	16.847	19.973	11.389	548.023	13.205	3.974	6.732	4.086	4.071	9.008	6.241	2.704	8.556
Safety Factor - Rear-face spalling	-0.968	-0.198	-0.895	-0.316	-0.17	-0.635	-0.962	-0.913	-0.735	-0.939	-0.96	-0.953	-0.959	-0.96	-0.955	-0.953
Z_p/Z_t	1.007	0.975	0.983	0.98	0.978	0.969	1.004	1.003	0.98	0.995	1.003	1	1.003	1.003	1.001	1

DataReader.m

```
clear variables; close('all'); fclose('all');
clc;

%
=========================================================================
=
% This code takes an input of text delimited data files titled 'CH1'
'CH2'
% etc. and reads text files, analyzes the voltage data, converts it to
% strain, determines principal strains and axes
% // For reading input data, data files must list Time and Voltage data
in
% 2 columns [Time, Voltage]
% -------------------------------------------------------------------
---
%
=========================================================================
=
% % % % % % % % % % % % % INITIALIZE VARIABLES % % % % % % % % % % % %
% %
%
=========================================================================
=
FileMatlabScripts = 'DataTrap Reader';
FileDataDirectory = 'Data';
FileTestDateFolder = '20190731';
FileExtension = '.txt';
% Directory for saving figures - change bSaveOutputPlots to enable
saving
% FileFigureDirectory = 'Figures';
FileDataPrefix = 'CH';                  % Prefix for input text files
FileDataDirectoryOUT = 'Output';
FileFigureExtensionOUT = 'png';        % Extension for saving files

% Boolean triggers
bAutoMode = true;                       % Auto analyze all files -
default:false
bOrganizeArray = false;                 % Sort data array?  Default:
false
bPlotInputCHData = false;               % Plot input data?
bPlotOutputCHData = true;               % Plot output data?
bPlotOutputStrainData = true;           % Plot output strain (cropped)?
bPlotOutputPrimaryStrainData = true;    % Plot eps_xx,eps_yy?
bPlotOutputPrincipalStressData = true;  % Plot output principal strain?
bPlotShockHEL = true;                   % Plot impact pressure vs HEL?
```

```
bPlotImpactEnergy = true;                % Plot impact parameters to
file?
bSaveOutputPlots = true;                 % Save figures to
FileFigureDirectory?
bWriteOutputFile = true;                 % Write output txt files?
bWriteInputFile = false;                 % Write channel input data?
bWriteCHStrain = true;                   % Write channel conversion to
strain?
bWritePrincipalStrain = true;            % Write principal strain data?
bWriteImpactEnergy = true;               % Write impact parameters to
file?
bCropToTimeZero = true;                  % Crop data before t = 0(true)
or trigger?
bReadInputImpactEnergy = true;           % Read ImpactEnergy.txt from
file

% ------------------------------------------------------------------------
---
% Initialize default dimensions and sensor serial references
FileTestCaseFolder = 'Hand tools';  % Required for initial states
checks
bCropOriginalState = bCropToTimeZero;
mInitializeDimensions;
% !!!!! Low-pass filter requires debugging
% Design low-pass filter
bFilter = false;                         % Filter signal? true/false
% lpf.filtertype = 'FIR';                  % Finite Impulse Response
filter
% lpf.samplerate = 10000;                  % (Hz) Sample rate
% lpf.Fpass = 8000;                        % (Hz) Passband frequency
% lpf.Fstop = 8500;                        % (Hz) Stopband frequency
% lpf.Rp = 0.1;                            % Passband ripple
% lpf.Astop = 70;                          % (dB) Stopband attenuation
%
% ------------------------------------------------------------------------
---
%
%
%
========================================================================
=
% % % % % % % % % % % % % % % SCRIPT START % % % % % % % % % % % % % % %
% %
%
========================================================================
=
% Output selection mode
fprintf('\n-----------------------------------------------------');
fprintf('\nNOTE>> Selection mode:\t');
if(bAutoMode)
    fprintf('AUTO');
else
    fprintf('MANUAL');
end
fprintf('\n-----------------------------------------------------');
fprintf('\n');
```

```
% ------------------------------------------------------------------------
---
% Initialize media mechanical properties for target and projectile
MediaLibrary = fMediaSelect();
Medium = MediaLibrary(1);
ProjectileLibrary = fTestCaseData(MediaLibrary);
% ------------------------------------------------------------------------
---
% Correct strain sensor sensitivity (ref: Product Manual, sect 8)
dataStrainSensorCal(:,2) = fSensitivityCorrection(...
    Medium,dataStrainSensorCal(:,2));
% ------------------------------------------------------------------------
---
% Acquire parent matlab script directory
mFileWorkingDirectory;
% ------------------------------------------------------------------------
---
% Store estimated impact energy
mImpactStructInitialize;
% ------------------------------------------------------------------------
---
% Generate symbol array
strGreeks = num2cell(char(945:969).');
% ------------------------------------------------------------------------
---
% Acquire test case for data analysis
bImproperSelection = true;
iCounter = 0;
iCounterLimit = 50;
% Initialize indices for auto file reading
ID_AutoTestDate = 3;
ID_AutoTestCase = 2;
% Check if ImpactEnergy struct is initialized from file and step
iCounter
[iCounter,ID_AutoTestDate,ID_AutoTestCase] = fiCounterStep(...
    bAutoMode,bReadInputImpactEnergy,...
    ImpactEnergy,FileParentFolder,FileDataDirectory);
% Suppress output text file and plot of impact energy until the end
bWriteImpactEnergyOriginal = bWriteImpactEnergy;
bPlotImpactEnergyOriginal = bPlotImpactEnergy;
bWriteImpactEnergy = false;
bPlotImpactEnergy = false;
% Perpetuate auto mode if selected until all test cases analyzed
bAutoModeTerminate = false;
bAnalyzeData = true;
while(bImproperSelection)
% ------------------------------------------------------------------------
---
%     Iterate counter --> prevent infinite loop
    iCounter = iCounter + 1;
    if(iCounter > iCounterLimit || iCounter < -iCounterLimit)
        fprintf('CAUTION>> Terminating selection mode due to high
iterations.\n');
        return
    end
```

```
% ------------------------------------------------------------------------
---
% Acquire and analyze individual test cases
bAnalyzeData = true;        % Default case to analyze data
mFileRawDataDirectory;
% Analyze data from selected date and case folders
if(bAnalyzeData)
% ------------------------------------------------------------------------
---
% Initialize dimensions and sensor serial references for test cases
mInitializeDimensions;
% ------------------------------------------------------------------------
---
% Initialize data structs and arrays
mFileReadInitializeArray;
% ------------------------------------------------------------------------
---
% Filter voltage data
mFilterChannelData;
% ------------------------------------------------------------------------
---
% Estimate impact velocity
mVelocityEstimate;
% ------------------------------------------------------------------------
---
% Convert Voltage signal to strain - ignore non-strain sensor data
mStrainCalcs;
% ------------------------------------------------------------------------
---
% Convert strain data to state of stress
[PrincipalStress,PrimaryStress] =
fStrainToStress2(Medium,StrainCrop,...
    PrimaryStrainCrop,StrainShearCrop,PrimaryStrainRateCrop,...
    [Serial_A;Serial_B;Serial_C;]);
% ------------------------------------------------------------------------
---
% Perform impact analysis
mImpactCalcs;
% ------------------------------------------------------------------------
---
% Write data to comma separated value
mWriteOutputData;
% ------------------------------------------------------------------------
---
% Plot and save data
mPlotData;
% ------------------------------------------------------------------------
---
% Plot state of stress in failure envelope
mMaterialFailure;
end
% ------------------------------------------------------------------------
---
% Terminate while loop for test case selection
if(bAutoMode)
    if(bAutoModeTerminate)
```

```
        % Reset booleans
        if(bWriteImpactEnergyOriginal)
            bWriteImpactEnergy = bWriteImpactEnergyOriginal;
        end
        if(bPlotImpactEnergyOriginal)
            bPlotImpactEnergy = bPlotImpactEnergyOriginal;
        end
        % Write final text file
        bWriteInputFile = false;
        bWriteCHStrain = false;
        bWritePrincipalStrain = false;
        mWriteOutputData;
        % Plot final chart
        bPlotInputCHData = false;
        bPlotOutputCHData = false;
        bPlotOutputStrainData = false;
        bPlotOutputPrimaryStrainData = false;
        bPlotOutputPrincipalStressData = false;
        bPlotShockHEL = false;
        mPlotData;
        fprintf('\nNOTE>>ALL TEST CASES HAVE BEEN ANALYZED. TERMINATING
SCRIPT\n');
        return
    else
        clearvars CHData CHDataStrain eps_A eps_B eps_C eps_xx eps_yy
eps_xy;
        clearvars PrimaryStrain PrimaryStrainCrop;
        clearvars PrimaryStrainRate PrimaryStrainRateCrop;
        clearvars PrincipalStress Strain StrainCrop StrainMax;
        clearvars StrainShearCrop;
        clearvars t_trigger tempThetaRosette V_stddev V_unstrained;
        clearvars velocity_impact vShock vShockLim vTime;
    end
else
fprintf('\nAnalyze another test case?\n');
ioTestCont = input('Selection [Y/N]:','s');
if(strcmp(ioTestCont,'Y'))
    bImproperSelection = true;
    clearvars CHData CHDataStrain eps_A eps_B eps_C eps_xx eps_yy
eps_xy;
    clearvars PrimaryStrain PrimaryStrainCrop;
    clearvars PrimaryStrainRate PrimaryStrainRateCrop;
    clearvars PrincipalStress Strain StrainCrop StrainMax;
    clearvars StrainShearCrop;
    clearvars t_trigger tempThetaRosette V_stddev V_unstrained;
    clearvars velocity_impact vShock vShockLim vTime;
elseif(strcmp(ioTestCont,'y'))
    bImproperSelection = true;
    clearvars CHData CHDataStrain eps_A eps_B eps_C eps_xx eps_yy
eps_xy;
    clearvars PrimaryStrain PrimaryStrainCrop;
    clearvars PrimaryStrainRate PrimaryStrainRateCrop;
    clearvars PrincipalStress Strain StrainCrop StrainMax;
    clearvars StrainShearCrop;
    clearvars t_trigger tempThetaRosette V_stddev V_unstrained;
    clearvars velocity_impact vShock vShockLim vTime;
```

```
elseif(strcmp(ioTestCont,'N'))
    % Reset booleans
    if(bWriteImpactEnergyOriginal)
        bWriteImpactEnergy = bWriteImpactEnergyOriginal;
    end
    if(bPlotImpactEnergyOriginal)
        bPlotImpactEnergy = bPlotImpactEnergyOriginal;
    end
    bImproperSelection = false;
    % Write final text file
    bWriteInputFile = false;
    bWriteCHStrain = false;
    bWritePrincipalStrain = false;
    mWriteOutputData;
    % Plot final chart
    bPlotInputCHData = false;
    bPlotOutputCHData = false;
    bPlotOutputStrainData = false;
    bPlotOutputPrimaryStrainData = false;
    bPlotOutputPrincipalStressData = false;
    bPlotShockHEL = false;
    mPlotData;
    return
elseif(strcmp(ioTestCont,'n'))
    % Reset booleans
    if(bWriteImpactEnergyOriginal)
        bWriteImpactEnergy = bWriteImpactEnergyOriginal;
    end
    if(bPlotImpactEnergyOriginal)
        bPlotImpactEnergy = bPlotImpactEnergyOriginal;
    end
    bImproperSelection = false;
    % Write final text file
    bWriteInputFile = false;
    bWriteCHStrain = false;
    bWritePrincipalStrain = false;
    mWriteOutputData;
    % Plot final chart
    bPlotInputCHData = false;
    bPlotOutputCHData = false;
    bPlotOutputStrainData = false;
    bPlotOutputPrimaryStrainData = false;
    bPlotOutputPrincipalStressData = false;
    bPlotShockHEL = false;
    mPlotData;
    return
else
    fprintf('WARNING>> Improper selection.\n');
    bImproperSelection = true;
end
end
end
```

---

fiCounterStep.m

```
---------------------------------------------------------------------------------------

function [iCounter,ID_AutoTestDate,ID_AutoTestCase] = fiCounterStep(...
    bAutoMode,bReadInputImpactEnergy,...
    ImpactEnergy,FileParentFolder,FileDataDirectory)
%
=========================================================================
=
% % Correct iCounter if impact data was read from file
% INPUTS:
%   bAutoMode = boolean, true if program is set to process
automatically
%   bReadInputImpactEnergy = boolean, true if impact energy was read
from
%   file
%   ImpactEnergy = struct, contains impact data for previous run tests
%   FileParentFolder = string, File path for Matlab scripts
%   FileDataDirectory = string, File path for location of DataTrap raw
data
%   folders
% OUTPUTS:
%   iCounter = index, Impact Energy struct index and while loop
protection
%   ID_AutoTestDate = index, folder for test dates being processed
%   ID_AutoTestCase = index, folder for current analysis
%
=========================================================================
=

if bReadInputImpactEnergy
    % Data was read from file --> step iCounter

    if(bAutoMode)
        % Match with test cases from file
        tempDirectory = strcat(FileParentFolder,'\',FileDataDirectory);
        dirTestDate = dir(tempDirectory);
        iCounter = 0;
    for i = 3:size(dirTestDate,1)
        FileTestDateFolder = dirTestDate(i).name;
        dirTestCase = dir(FileTestDateFolder);
        for j = 3:size(dirTestCase,1)
            FileTestCaseFolder = dirTestCase(j).name;
            % FilePath =
strcat(FileTestDateFolder,'\',FileTestCaseFolder,'\');
            for k = 1:size(ImpactEnergy,2)
                if(strcmp(FileTestCaseFolder,ImpactEnergy(k).Case))
                    iCounter = iCounter + 1;
                    break;
                end
            end
            if(abs(iCounter - size(ImpactEnergy,2)) < 1e-6)
                break;
            end
        end
        if(abs(iCounter - size(ImpactEnergy,2)) < 1e-6)
```

```
            break;
        end
    end
        % Determine folder indices for processing
        if(abs(j - size(dirTestCase,1)) < 1e-6)
            % Analysis completed all runs for test date
            ID_AutoTestDate = i+1;
            ID_AutoTestCase = 2;
        else
            fprintf('\nWARNING>> Input file read failure.  Default
indices set.\n');
            iCounter = 0;
            ID_AutoTestDate = 3;
            ID_AutoTestCase = 2;
        end
    end

else
    fprintf('\nWARNING>> Input file read failure.  Default indices
set.\n');
    iCounter = 0;
    ID_AutoTestDate = 3;
    ID_AutoTestCase = 2;
end
end
```

---

fMediaSelect.m

```
function [MediaLibrary] = fMediaSelect()
%=========================================================================
===
%Selection of media for impact and shock Hugoniot calculations
%=========================================================================
===
% Define Media material properties
%-------------------------------------------------------------------------
---
iMaterials = 5;
%-------------------------------------------------------------------------
---
% CONCRETE
% Reference:  Matweb; ACI318-05 Building Code Req for Structural Conc;
% "Behavior of plain concrete subjected to tensile loading at high
strain
% rate", Cadoni et al
Media = 'Concrete (35MPA)';
Density = 2627;                     % [kg/m^3] - Media density {2240-
2400}
Hardness_Brinell = -Inf;            % Brinell Hardness
% Elastic constants (Static)
E = 30e9;                           % [Pa] - Modulus of Elasticity
```

```
PoissonRatio = 0.2;
G = 21e9;                           % [Pa] - Modulus of Rigidity
(Shear)
% Elastic constants (Dynamic)
% Note:  dynamic elastic modulus adjusted based on material properties
and
% "Behavior of plain concrete subjected to tensile loading at high
strain
% rate", Cadoni et al
% E_dynamic = (54.8e9/43.1e9)*E;      % [Pa] - Dynamic Elastic Modulus
E_dynamic = 54.8e9;                 % [Pa] - Dynamic Elastic Modulus
% Lame's constants - HOMOGENEOUS ISOTROPIC LINEAR ELASTIC MATERIALS
% Stress Waves in Solids, H. Kolsky
lambda = (E * PoissonRatio)/((1 + PoissonRatio)*(1 - 2*PoissonRatio));
mu = E/(2*(1 + PoissonRatio));      % [Pa] - same as Modulus of
Rigidity
mu_dynamic = E_dynamic/(2*(1 + PoissonRatio));
K = lambda + (2*mu)/3;              % [Pa] - Bulk Modulus
% Elastic wave properties
C_0 = (E/Density)^(1/2);            % [m/s] - L wave velocity for slim
bar
C_L = ((lambda + 2 * mu)/Density)^(1/2);    % [m/s] - Longitudinal wave
velocity
C_S = (mu/Density)^(1/2);               % [m/s] - Shear wave velocity
C_R = ((0.862 + 1.14*PoissonRatio)/(1 + PoissonRatio)) * C_S;   %[m/s]
Rayleigh surface wave velocity
% Plastic wave properties
% Shock properties
% C_0 = 3.3;                          % [km/s] - L wave velocity for
slim bar
S = 1.33;                           % Hugoniot constant
% Strength of Materials (Static)
Tensile_Yield = 4.4e6;              % [Pa]
Tensile_Ultimate = 2.5e6;           % [Pa] {2e6 -> 5e6} Pa
Compressive_Yield = 35e6;           % [Pa]
Shear_Ultimate = 17e6;              % [Pa] {6e6 -> 17e6} Pa
% Strength of Materials (Dynamic)
% Correlated to strain-rate --> f_ty,dyn = [strain-rate, f_ty,d];
% Max aggregate size 5 mm
% Tensile_Yield_Dynamic = [10e-6, 2.7e6; 10e-2, 2.5e-6; 1, 9.2e6; 10,
10.5e6];
% Max aggregate size 10 mm
Tensile_Yield_Dynamic = [10e-6, 3.6e6;
    10e-2, 4.4e6;
    1, 7.4e6;
    10, 10.5e6;];
HEL = 1.75*1e8; % [kbar] -> [Pa] Transition for dynamic inelastic
behavior
SpallStr = 30e6;                    % [Pa] F_dty -> tensile/spall str -
> compressive/relief waves
% Failure Criterion [vonMises, Tresca, Coulomb, Mohr-Coulomb]
FailureCriteria = ["Mohr-Coulomb"; "Coulomb";];
Cohesion = 1186e3;                  % [Pa]
FrictionAngle = 54.9;               % [deg]
% ***************************** CORRECTIONS
***************************
```

```
% Elasticity Modulus correction (ACI318-05 Section 8.5)
% if(90 <= Density*0.062428 <= 155)
%     E = 57000 * sqrt(Compressive_Yield*0.000145038);
% else
%     E = (Density*0.062428)^(1.5) * 33 *
sqrt(Compressive_Yield*0.00145038);
% end
% G = E/2;                              % ACI318-05 Section R13.6.4.2
% % Dynamic Elastic Modulus
% E_dynamic = (54.8e9/43.1e9)*E;        % [Pa] - Dynamic Elastic Modulus
% % Lame's Constants - HOMOGENEOUS ISOTROPIC LINEAR ELASTIC MATERIALS
% % Stress Waves in Solids, H. Kolsky
% lambda = (E * PoissonRatio)/((1 + PoissonRatio)*(1 -
2*PoissonRatio));
% mu = E/(2*(1 + PoissonRatio));        % [Pa] - same as Modulus of
Rigidity
% K = lambda + (2*mu)/3;                % [Pa] - Bulk Modulus
% % Elastic wave properties
% C_0 = (E/Density)^(1/2);              % [m/s] - L wave velocity for
slim bar
% C_L = ((lambda + 2 * mu)/Density)^(1/2);    % [m/s] - Longitudinal
wave velocity
% C_S = (mu/Density)^(1/2);                % [m/s] - Shear wave velocity
% C_R = ((0.862 + 1.14*PoissonRatio)/(1 + PoissonRatio)) * C_S;
%[m/s] Rayleigh surface wave velocity
% Hardness correction
tempRB = 73.5 * log10(Compressive_Yield*0.000145038) - 187;   %Rockwell
Hardness test
tempRBlim = [52 100];
tempHBlim = [90 245];
Hardness_Brinell = tempHBlim(1) + ((tempRB - tempRBlim(1))/...
    (tempRBlim(2) - tempRBlim(1)))*(tempHBlim(2)-tempHBlim(1));
% Form struct
Media_ = struct('Media',Media,'Density',Density,...
    'ElasticityModulus',E,'ElasticityModulusDynamic',E_dynamic,...
    'ShearModulus',G,'BulkModulus',K,...
    'PoissonRatio',PoissonRatio,'lambda',lambda,...
    'mu',mu,'muDynamic',mu_dynamic,...
    'C_0',C_0,'C_L',C_L,'C_S',C_S,'C_R',C_R,'S',S,...
    'TensileYield',Tensile_Yield,'TensileUltimate',Tensile_Ultimate,...

'CompressiveYield',Compressive_Yield,'ShearUltimate',Shear_Ultimate,...
    'TensileYieldDynamic',Tensile_Yield_Dynamic,...
    'HugoniotElasticLimit',HEL,'SpallStrength',SpallStr,...
    'FailureCriteria',FailureCriteria,...
    'Cohesion',Cohesion,'FrictionAngle',FrictionAngle);
clearvars Media Density Hardness_Brinell E PoissonRatio G lambda mu K;
clearvars E_dynamic mu_dynamic Tensile_Yield_Dynamic HEL;
clearvars C_0 C_L C_S C_R S Tensile_Yield Tensile_Ultimate SpallStr;
clearvars Compressive_Yield Shear_Ultimate FailureCriteria;
clearvars Cohesion FrictionAngle;
clearvars tempRB tempRBlim tempHBlim;
%-------------------------------------------------------------------
---
% Initialize library structure
for i=1:iMaterials
```

```
    MediaLibrary_(i) = Media_;
end
%-------------------------------------------------------------------------
---
% High Density Polyethelene (HDPE)
% Reference:  Matweb, HDPE Injection Molded
Media = 'HDPE';
Density = 954;                      % [kg/m^3] - Blast media density
Hardness_Brinell = 147;             % Brinell Hardness {Rockwell|80 -
112}
% Elastic constants (Static)
tempRho = [924 2550];               % [kg/m^3]
tempE = [4.50e9 1.500e9];           % [Pa]
E = tempE(1) + (Density - tempRho(1))/(tempRho(2) - tempRho(1))...
    * (tempE(2) - tempE(1));
PoissonRatio = 0.3;                 % !!!!! ESTIMATED
G = E/(2*(1 + PoissonRatio));       % [Pa] - estimated
% Elastic constants (Dynamic)
E_dynamic = 1.3*E;                  % !!!!! FIX THIS !!!!!
% Lame's constants - HOMOGENEOUS ISOTROPIC LINEAR ELASTIC MATERIALS
% Stress Waves in Solids, H. Kolsky
lambda = (E * PoissonRatio)/((1 + PoissonRatio)*(1 - 2*PoissonRatio));
mu = E/(2*(1 + PoissonRatio));      % [Pa] - same as Modulus of
Rigidity
mu_dynamic = E_dynamic/(2*(1 + PoissonRatio));
K = lambda + (2*mu)/3;              % [Pa] - Bulk Modulus
% Elastic wave properties
C_0 = (E/Density)^(1/2);            % [m/s] - L wave velocity for slim
bar
C_L = ((lambda + 2 * mu)/Density)^(1/2);    % [m/s] - Longitudinal wave
velocity
C_S = (mu/Density)^(1/2);               % [m/s] - Shear wave velocity
C_R = ((0.862 + 1.14*PoissonRatio)/(1 + PoissonRatio)) * C_S;   %[m/s]
Rayleigh surface wave velocity
% Plastic wave properties
% Shock properties
% C_0 = 2.8098;                       % [km/s] - Hugoniot shock
constant
S = 1.576;                          % Hugoniot constant
% Strength of Materials (Static)
Tensile_Yield = 26.1e6;             % [Pa] {11.0e6 -> 43.0e6} Pa
Tensile_Ultimate = 21.3e6;          % [Pa] {7.60e6 -> 43.0e6} Pa
Compressive_Yield = 12.6e6;         % [Pa] {4.0e6 -> 23.0e6} Pa
Shear_Ultimate = 33.1e6;            % [Pa]
% Strength of Materials (Dynamic)
Tensile_Yield_Dynamic = 1.25*Tensile_Yield; % !!!!!!! FIX THIS !!!!!
HEL = 6e9;                          % [Pa] !!!!! FIX THIS ESTIMATE
!!!!!
SpallStr = 1e9;                     % [Pa] !!!!! FIX THIS ESTIMATE
!!!!!
% Failure Criterion [vonMises, Tresca, Coulomb, Mohr-Coulomb]
FailureCriteria = "Tresca";
Cohesion = -Inf;
FrictionAngle = -Inf;
% Form struct
Media_ = struct('Media',Media,'Density',Density,...
```

```
    'ElasticityModulus',E,'ElasticityModulusDynamic',E_dynamic,...
    'ShearModulus',G,'BulkModulus',K,...
    'PoissonRatio',PoissonRatio,'lambda',lambda,...
    'mu',mu,'muDynamic',mu_dynamic,...
    'C_0',C_0,'C_L',C_L,'C_S',C_S,'C_R',C_R,'S',S,...
    'TensileYield',Tensile_Yield,'TensileUltimate',Tensile_Ultimate,...

'CompressiveYield',Compressive_Yield,'ShearUltimate',Shear_Ultimate,...
    'TensileYieldDynamic',Tensile_Yield_Dynamic,...
    'HugoniotElasticLimit',HEL,'SpallStrength',SpallStr,...
    'FailureCriteria',FailureCriteria,...
    'Cohesion',Cohesion,'FrictionAngle',FrictionAngle);
MediaLibrary_(2) = Media_;
clearvars Media Density Hardness_Brinell E PoissonRatio G lambda mu K;
clearvars E_dynamic mu_dynamic Tensile_Yield_Dynamic;
clearvars C_0 C_L C_S C_R S Tensile_Yield Tensile_Ultimate HEL
SpallStr;
clearvars Compressive_Yield Shear_Ultimate FailureCriteria;
clearvars Cohesion FrictionAngle;
clearvars tempE tempRho;
%-------------------------------------------------------------------------
---
% LIMESTONE
% Reference:  Matweb, Calcium Oxide; LASL Dynamic Material Properties
Media = 'Limestone';
Density = 2920;                     % [kg/m^3] {1790 -> 2920} kg/m^3
Hardness_Brinell = 151;             % Brinell Hardness {2->5}Moh's
Hardness
% Elastic constants (Static)
E = 76e9;                           % [Pa] {10e9 -> 80e9} Pa
PoissonRatio = 0.2;
G = E/(2*(1 + PoissonRatio));       % [Pa] - Modulus of Rigidity
(Shear)
% Elastic constants (Dynamic)
E_dynamic = 1.25*E;                 % !!!!! FIX THIS !!!!!
% Lame's Constants - HOMOGENEOUS ISOTROPIC LINEAR ELASTIC MATERIALS
% Stress Waves in Solids, H. Kolsky
lambda = (E * PoissonRatio)/((1 + PoissonRatio)*(1 - 2*PoissonRatio));
mu = E/(2*(1 + PoissonRatio));      % [Pa] - same as Modulus of
Rigidity
mu_dynamic = E_dynamic/(2*(1 + PoissonRatio));
K = lambda + (2*mu)/3;              % [Pa] - Bulk Modulus
% Elastic wave properties
C_0 = (E/Density)^(1/2);            % [m/s] - L wave velocity for slim
bar
C_L = ((lambda + 2 * mu)/Density)^(1/2);    % [m/s] - Longitudinal wave
velocity
C_S = (mu/Density)^(1/2);               % [m/s] - Shear wave velocity
C_R = ((0.862 + 1.14*PoissonRatio)/(1 + PoissonRatio)) * C_S;   %[m/s]
Rayleigh surface wave velocity
% Plastic wave properties
% Shock properties
% C_0 = 3.5864;                       % [km/s] - Hugoniot shock
constant
S = 1.8193;                         % Hugoniot constant
% Strength of Materials (static)
```

```
Tensile_Yield = 7.8e6;                  % [Pa]
Tensile_Ultimate = 6e6;                 % [Pa] {5e6 -> 25e6} Pa
Compressive_Yield = 14e6;               % [Pa] {13.8e6 -> 255e6} Pa
Shear_Ultimate = 6.7e6;                 % [Pa]
% Strength of Materials (Dynamic)
Tensile_Yield_Dynamic = 1.25*Tensile_Yield;
HEL = 100e9;                            % [Pa] !!!!! FIX THIS ESTIMATE
!!!!!
SpallStr = 50e6;                        % [Pa] !!!!! FIX THIS ESTIMATE
!!!!!
% Failure Criterion [vonMises, Tresca, Coulomb, Mohr-Coulomb]
FailureCriteria = "Mohr-Coulomb";
Cohesion = 6.7e6;                       % [Pa]
FrictionAngle = 42.0;                   % [deg]
% Mohr-Coulomb Failure Criteria for brittle materials
% Cohesion = 6.7e6;                             % [Pa]
% FrictionAngle = 42.0;                         % [deg]
% RangeOfConfiningPressure = 7e6;               % [Pa] {0 -> 10e6} Pa
% Form struct
Media_ = struct('Media',Media,'Density',Density,...
    'ElasticityModulus',E,'ElasticityModulusDynamic',E_dynamic,...
    'ShearModulus',G,'BulkModulus',K,...
    'PoissonRatio',PoissonRatio,'lambda',lambda,...
    'mu',mu,'muDynamic',mu_dynamic,...
    'C_0',C_0,'C_L',C_L,'C_S',C_S,'C_R',C_R,'S',S,...
    'TensileYield',Tensile_Yield,'TensileUltimate',Tensile_Ultimate,...

'CompressiveYield',Compressive_Yield,'ShearUltimate',Shear_Ultimate,...
    'TensileYieldDynamic',Tensile_Yield_Dynamic,...
    'HugoniotElasticLimit',HEL,'SpallStrength',SpallStr,...
    'FailureCriteria',FailureCriteria,...
    'Cohesion',Cohesion,'FrictionAngle',FrictionAngle);
MediaLibrary_(3) = Media_;
clearvars Media Density Hardness_Brinell E PoissonRatio G lambda mu K;
clearvars E_dynamic mu_dynamic Tensile_Yield_Dynamic;
clearvars C_0 C_L C_S C_R S Tensile_Yield Tensile_Ultimate HEL
SpallStr;
clearvars Compressive_Yield Shear_Ultimate FailureCriteria;
clearvars Cohesion FrictionAngle;
%-------------------------------------------------------------------------
---
% STEEL
% Reference:  Matweb, AISI 1060 Steel; MMPDS-09
Media = 'Steel (1060)';
Density = 7850;                         % [kg/m^3] - Blast media density
Hardness_Brinell = 241;                 % Brinell Hardness
% Elastic constants (Static)
E = 205e9;                              % [Pa] - Modulus of Elasticity
PoissonRatio = 0.29;                    % Poisson's Ratio
G = 80e9;                               % [Pa] - Modulus of Rigidity
(Shear)
% Elastic constants (Dynamic)
E_dynamic = 1.5*E;                      % !!!!! FIX THIS !!!!!
% Lame's Constants - HOMOGENEOUS ISOTROPIC LINEAR ELASTIC MATERIALS
% Stress Waves in Solids, H. Kolsky
lambda = (E * PoissonRatio)/((1 + PoissonRatio)*(1 - 2*PoissonRatio));
```

```
mu = E/(2*(1 + PoissonRatio));      % [Pa] - same as Modulus of
Rigidity
mu_dynamic = E_dynamic/(2*(1 + PoissonRatio));
K = lambda + (2*mu)/3;              % [Pa] - Bulk Modulus
% Elastic wave properties
C_0 = (E/Density)^(1/2);            % [m/s] - L wave velocity for slim
bar (
C_L = ((lambda + 2 * mu)/Density)^(1/2);    % [m/s] - Longitudinal wave
velocity
C_S = (mu/Density)^(1/2);               % [m/s] - Shear wave velocity
C_R = ((0.862 + 1.14*PoissonRatio)/(1 + PoissonRatio)) * C_S;   %[m/s]
Rayleigh surface wave velocity
% Plastic wave properties
% Shock properties
% C_0 = 4.58;                           % [km/s] - Hugoniot shock
constant
S = 1.49;                           % Hugoniot constant
% Strength of Materials (Static)
Tensile_Yield = 485e6;              % [Pa]
Tensile_Ultimate = 814e6;           % [Pa]
Compressive_Yield = 12.6e6;         % [Pa]
Shear_Ultimate = 241.3e6;           % [Pa]
% Strength of Materials (Dynamic)
Tensile_Yield_Dynamic = 1.25*Tensile_Yield;  % !!!!! FIX THIS !!!!!
HEL = 1.5e9;    % [Pa] estimated from 'Dynamic Behavior of Materials'
SpallStr = 1.6e9;                   % [Pa] 'Explosives Engineering',
Cooper
% Failure Criterion [vonMises, Tresca, Coulomb, Mohr-Coulomb]
FailureCriteria = "vonMises";
Cohesion = -Inf;
FrictionAngle = -Inf;
% Form struct
Media_ = struct('Media',Media,'Density',Density,...
    'ElasticityModulus',E,'ElasticityModulusDynamic',E_dynamic,...
    'ShearModulus',G,'BulkModulus',K,...
    'PoissonRatio',PoissonRatio,'lambda',lambda,...
    'mu',mu,'muDynamic',mu_dynamic,...
    'C_0',C_0,'C_L',C_L,'C_S',C_S,'C_R',C_R,'S',S,...
    'TensileYield',Tensile_Yield,'TensileUltimate',Tensile_Ultimate,...

'CompressiveYield',Compressive_Yield,'ShearUltimate',Shear_Ultimate,...
    'TensileYieldDynamic',Tensile_Yield_Dynamic,...
    'HugoniotElasticLimit',HEL,'SpallStrength',SpallStr,...
    'FailureCriteria',FailureCriteria,...
    'Cohesion',Cohesion,'FrictionAngle',FrictionAngle);
MediaLibrary_(4) = Media_;
clearvars Media Density Hardness_Brinell E PoissonRatio G lambda mu K;
clearvars E_dynamic mu_dynamic Tensile_Yield_Dynamic;
clearvars C_0 C_L C_S C_R S Tensile_Yield Tensile_Ultimate HEL
SpallStr;
clearvars Compressive_Yield Shear_Ultimate FailureCriteria;
clearvars Cohesion FrictionAngle;
%-------------------------------------------------------------------
---
% WOOD
Media = 'Wood (Pine)';
```

```
Density = 450;                          % [kg/m^3] {350 -> 590} kg/m^3
Hardness_Brinell = 2.0;                 % Brinell Hardness
% Elastic constants (Static)
E = 7.03e9;                             % [Pa] - Modulus of Elasticity
PoissonRatio = 0.349;                   % Longitudinal-T
G = E/(2*(1 + PoissonRatio));           % [Pa] - Modulus of Rigidity
(Shear)
% Elastic constants (Dynamic)
E_dynamic = 1.25*E;                     % !!!!! FIX THIS !!!!!
% Lame's Constants - HOMOGENEOUS ISOTROPIC LINEAR ELASTIC MATERIALS
% Stress Waves in Solids, H. Kolsky
lambda = (E * PoissonRatio)/((1 + PoissonRatio)*(1 - 2*PoissonRatio));
mu = E/(2*(1 + PoissonRatio));          % [Pa] - same as Modulus of
Rigidity
mu_dynamic = E_dynamic/(2*(1 + PoissonRatio));
K = lambda + (2*mu)/3;                  % [Pa] - Bulk Modulus
% Elastic wave properties
C_0 = (E/Density)^(1/2);                % [m/s] - L wave velocity for slim
bar (
C_L = ((lambda + 2 * mu)/Density)^(1/2);    % [m/s] - Longitudinal wave
velocity
C_S = (mu/Density)^(1/2);                 % [m/s] - Shear wave velocity
C_R = ((0.862 + 1.14*PoissonRatio)/(1 + PoissonRatio)) * C_S;   %[m/s]
Rayleigh surface wave velocity
% Plastic wave properties
% Shock properties
% C_0 = 0.45;                             % [km/s] - Hugoniot shock
constant
S = 1.33;                               % Hugoniot constant
% Strength of Materials (Static)
Tensile_Yield = 80e6;                   % [Pa] {59e6 -> 100e6} Pa
Tensile_Ultimate = 78e6;                % [Pa]
Compressive_Yield = 23.7e6;             % [Pa] (Parallel to grain)
Shear_Ultimate = 7.86e6;                % [Pa] {6.1 -> 10.4} Pa
% Strength of Materials (Dynamic)
Tensile_Yield_Dynamic = 1.25*Tensile_Yield;  %!!!!! FIX THIS !!!!!
HEL = 4.5e9;                            % [Pa] !!!!! FIX THIS ESTIMATE
!!!!!
SpallStr = 90e6;                        % [Pa] !!!!! FIX THIS ESTIMATE
!!!!!
% Failure Criterion [vonMises, Tresca, Coulomb, Mohr-Coulomb]
FailureCriteria = "Coulomb";
Cohesion = -Inf;
FrictionAngle = -Inf;
% Form struct
Media_ = struct('Media',Media,'Density',Density,...
    'ElasticityModulus',E,'ElasticityModulusDynamic',E_dynamic,...
    'ShearModulus',G,'BulkModulus',K,...
    'PoissonRatio',PoissonRatio,'lambda',lambda,...
    'mu',mu,'muDynamic',mu_dynamic,...
    'C_0',C_0,'C_L',C_L,'C_S',C_S,'C_R',C_R,'S',S,...
    'TensileYield',Tensile_Yield,'TensileUltimate',Tensile_Ultimate,...

'CompressiveYield',Compressive_Yield,'ShearUltimate',Shear_Ultimate,...
    'TensileYieldDynamic',Tensile_Yield_Dynamic,...
    'HugoniotElasticLimit',HEL,'SpallStrength',SpallStr,...
```

```
    'FailureCriteria',FailureCriteria,...
    'Cohesion',Cohesion,'FrictionAngle',FrictionAngle);
MediaLibrary_(5) = Media_;
clearvars Media Density Hardness_Brinell E PoissonRatio G lambda mu K;
clearvars E_dynamic mu_dynamic Tensile_Yield_Dynamic;
clearvars C_0 C_L C_S C_R S Tensile_Yield Tensile_Ultimate HEL
SpallStr;
clearvars Compressive_Yield Shear_Ultimate FailureCriteria;
clearvars Cohesion FrictionAngle;
%-------------------------------------------------------------------------
---
% Convert to SI or Standard units
% DENSITY [kg/m^3] --> 3.61273e-5 [lb/in^3]
% PRESSURE [Pa] --> 0.000145038 [psi] (i.e. 1 ksi = 6.895e6 Pa)
% for i = 1:size(MediaLibrary_,2)
%     MediaLibrary_(i).Density = MediaLibrary_(i).Density * 3.61273e-5;
% end

MediaLibrary = MediaLibrary_;

end
```

---

fOrganizeTimeVoltage.m

```
function [outTime,outVoltage] =
fOrganizeTimeVoltage(inFileData,boolOrgA)
%fOrganizeTimeVoltage takes an input matrix inFileData and pulls the
time
%and voltage data from the matrix - then arranges the Time array in
%ascending order and correlates the voltage to the organized Time data

% % Organize data

% Ensure FileData has an even number of columns (Time&Voltage
correlation)
if(rem(size(inFileData,1),2)~=0)
    fprintf('Data set does not have an even number of columns');
    return;
end
outTime = ones(size(inFileData,1)/2,1);
outVoltage = ones(size(inFileData,1)/2,1);
for i=1:size(inFileData,1)
    j = i/2;
    j = int64(j);                %% !!!! - ENSURE intXX() SUFFICIENT
SIZE
    if(rem(i,2)~=0)
        outTime(j) = inFileData(i);
    else
        outVoltage(j) = inFileData(i);
```

```
    end
end
if(boolOrgA)
%     Arrange time data in ascending order and correlate voltage
    fprintf('Arranging Time data in ascending order\n');
    tempTime = outTime;
    outTime = sort(outTime,'ascend');
    tempVoltage = outVoltage;
    for i=1:size(outTime,1)
        for j=1:size(tempTime,1)
            dt = outTime(i) - tempTime(j);
            if(abs(dt)<1e-6)
                break;
            end
        end
        outVoltage(i) = tempVoltage(j);
    end
end

end
```

---

fPenetrationConcrete.m

```
function [zPenetration] = fPenetrationConcrete(Projectile,...
    V_impact,Target,dimThickness)
%
=========================================================================
=
% % Estimate penetration into concrete using terminal ballistic models
% Inputs:
% Projectile -> struct with projectile mechanical data
% vImpact -> impact velocity in (ft/s)
% Target -> struct with target mechanical data
% dimThickness -> thickness of seal (in)
%
=========================================================================
=
% Initialize penetration array
tempz = zeros(2,1);
% -------------------------------------------------------------------
---
% Reference:  "Ballisticts:  Theory and Design of Guns and Ammunition"
% Carlucci & Jacobson
% -------------------------------------------------------------------
---
% Section 15.2 - Penetration and Perforation of Concrete
% Force on nose of the projectile
% F_impact = (pi()*Projectile.Diameter^2/4) * (tau_0*A + N*B*rho*V^2);
% N = (8*psi - 1)/(24*psi^2);
% psi = s/d;                        % caliber radius head
% s -> ogive radius
```

```
% d -> projectile diameter
% V_impact -> projectile velocity
% B = 1;                              % compressive strength parameter
% tau_0*A = S*f'_c;                   % compressive strength parameter
% ------------------------------------------------------------------------
---
% Forrestal model
% Assumptions:  rigid body projectile, minimal projectile yaw (<5 deg)
% ------------------------------------------------------------------------
---
% Test case:
% Forrestal model requires d_p,s in (m) and m_p in (kg), f_c in (Pa),
% rho_t in (kg/m^3)
% d_p = 0.50*0.0254;                % (in) -> (m) caliber
% s = 0.0635;                       % (m) ogive radius
% psi = s/d_p;                      % caliber-radius-head
% m_p = 662*6.47989e-5;             % (grains) -> (kg)
% f_c = 2100 * 6894.76;             % (Pa) unconfined compressive str of
conc
% rho_t = 0.084 * 27679.9;          % (lb/in^3) -> (kg/m^3) density of
conc
% V_p = 2000 * 0.3048;              % (ft/s) -> (m/s) impact velocity
% z_p = 0.204;                      % (m) - expected penetration
% z_t = 1e3;                        % (m) target thickness

% Initialize variables
d_p = Projectile.Diameter*0.0254;   % (in) -> (m) Caliber
s = Projectile.OgiveRadius*0.0254;  % (in) -> (m) Ogive radius
psi = s/d_p;                        % caliber-radius-head
f_c = Target.CompressiveYield;      % (Pa)
rho_t = Target.Density;             % (kg/m^3)
m_p = Projectile.Mass*0.453592;     % (lb) -> (kg)
V_p = V_impact*0.3048;              % (ft/s) -> (m/s)
z_t = dimThickness * 0.0254;        % (in) -> (m)
% ------------------------------------------------------------------------
---
% F_impact = (pi()*Projectile.Diameter^2/4) * (tau_0*A + N*B*rho*V^2);
% Dimensionless strength parameter
% Convert f_c to MPa for the curve fit
S = 93.48*(f_c/1e6)^(-0.5603);            % ref eqn 15.128
% S varies from S = 86.431*f_c^-0.5158; to S=103.71*f_c^-0.6142;
% B = 1;
% Determine N, dimensionless nose shape parameter
if(psi < 1e-6)
    N = 1e6;
else
    N = (8*psi - 1)/(24*psi^2);
end

% If projectile penetrates beyond crater region
(~2*Projectile.Diameter) it
% enters tunnel region
% Instantaneous depth of penetration (z)
% For z > 2*Projectile.Diameter
```

```
% F = (pi()*Projectile.Diameter^2/4) *
(S*f_c+N*Target.Density*V_impact^2)

% Check if projectile will create a tunnel -> V_impact <= V_s2
V_s2 = sqrt((pi()*d_p^3*S*f_c)/(2*m_p));            % sect 15.119
% Velocity of projectile at 2*d_p, which occurs at t_1
V_1 = sqrt((2*m_p*V_p^2 - pi()*d_p^3*S*f_c)/...
        (2*m_p + pi()*d_p^3*N*rho_t));
V_1 = real(V_1);

c = (m_p/(4*d_p^2))*(V_p^2 - V_1^2);
    % sect 15.113

t_1 = sqrt(m_p/c)*acos(V_1/V_p);
% If V goes to 0 before time t_1, the projectile never penetrates
deeper
% than the crater region and our analysis would be complete

if(V_p - V_s2 <= 1e-6)
    % Projectile does not penetrate beyond cratering region
    % For 0 < z < 2*Projectile.Diameter
    % F = c*z

    % Estimate penetration depth
    tempz(1) = V_p*sqrt(m_p/c);             % sect 15.118
else
    % Projectile enters tunneling region (sect 15.124)
    tempz(1) = (2*m_p/(pi()*d_p^2*N*rho_t))*...
        log(1+(N*rho_t*V_1^2)/(S*f_c))-2*d_p;
    % Ensure penetration depth does not exceed seal thickness
    if(tempz(1) > z_t)
        tempz(1) = z_t;
    end
end

if(tempz(1) < 0)
    tempz(1) = 0;
end
% % Convert penetration depth from (m) to (in)
% tempz(1) = tempz(1)*39.3701;
% Ensure only real values are propagated
tempz(1) = real(tempz(1));
% ------------------------------------------------------------------
---
% National Defense Research Committee (NDRC) model
% ------------------------------------------------------------------
---
% ref: "Impact Analysis of Concrete Structural Components", Murthy et
al
% Imperial
% ref: "Impact effects of explosively formed projectiles on normal and
high
% stress concrete", Bookout
% G(x/d) = K * (N*) * d_p^(0.2) * (m_p/d_p^3) * (V_p/1000)^1.80
```

```
% K = 180/sqrt(f_c);        % Concrete penetrability factor (f_c in
psi)
% N* = 0.72 + 0.25*sqrt(n - 0.25);  % nose-shape factor (n is radius in
% calibers of a tangent-ogive projectile nose, aka caliber-radius-head)
% n = psi;
% D = (m_p/d_p^3);      % Caliber-density factor (lb/in^3)
% Initialize variables
d_p = Projectile.Diameter;
m_p = Projectile.Mass;

% Determine concrete penetrability factor -> ensure f_c is in psi
K = 180/sqrt(Target.CompressiveYield*0.000145038);
% Determine nose-shape factor
N = 0.72 + 0.25*sqrt(psi - 0.25);
% Determine caliber-density factor (lb/in^3)
D = (m_p/d_p^3);
% Determine penetration parameter in calibers -> V_impact in ft/s
G = K * N * d_p^0.2 * D * (V_impact/1000)^1.80;
% Determine penetration distance
tempz(2) = 2*d_p * sqrt(G);
if((tempz(2)/d_p) - 2 < 1e-6)
    tempz(2) = tempz(2);
else
    tempz(2) = d_p * (1 + G);
end
% Convert penetration estimate to (m)
tempz(2) = tempz(2)*0.0254;
% Ensure only real values are propagated
tempz(2) = real(tempz(2));
% ------------------------------------------------------------------------
---
%
////////////////////////////////////////////////////////////////////////
/
% ------------------------------------------------------------------------
---
% Determine final penetration
dz = abs(tempz(1) - tempz(2));
error = 0.30;
if(dz - min(tempz(:,1)) > 1e-6 || ...
        abs((dz - mean(tempz(:,1)))/mean(tempz(:,1))) - error > 1e-6)
    % The difference in penetration models is significant
    zPenetration  = min(tempz);
else
    zPenetration = mean(tempz);
end
end
```

---

fSensitivityCorrection.m

```
function [SensArrayOUT] = fSensitivityCorrection(Medium,SensArrayIN)
```

```
%=========================================================================
===
%Correct strain sensor sensitivity based on PCB product manual
% INPUT:
% Medium -> struct with mechanical properties, specifically Young's mod
% SensArrayIN -> Mx1 array with sensitivity data for strain sensors
%=========================================================================
===
% Reference PCB product manual for 740B02 section 8.0 Sensitivity
% Material;     Modulus of Elasticity (psi);    Sensitivity
(mV/microstrain)
% Steel;        30e6                            50
% Aluminum;     10e6                            41
% Acrylic;      0.4e6                           5
%-------------------------------------------------------------------------
---
% Manufacturer calibration correction data
%  [E (psi), S (mV/microstrain)]
SensCal = [30e6, 50; 10e6, 41; 0.4e6, 5];
SensCal(:,1) = SensCal(:,1)*6894.76;                % convert E to [Pa]
% Retrieve modulus of elasticity for target
E = Medium.ElasticityModulus;
SensArrayOUT = SensArrayIN;
for i = 1:size(SensArrayIN,1)
    for j = 1:size(SensCal,1)
        if(E - SensCal(1,1) > 1e-6)
            % When the stiffness modulus of the structure under test is
            % less than the modulus of steel, the actual sensitivity is
            % less than the calibrated sensitivity
            SensArrayOUT(i) = SensArrayIN(i);
            break
        elseif(E - SensCal(3,1) < 1e-6)
            % Linear extrapolation from calibration curve
            tempS = SensCal(2,2)-((SensCal(2,1)-E)/...
                (SensCal(2,1)-SensCal(3,1)))*(SensCal(2,2)-
SensCal(3,2));
            SensArrayOUT(i) = (tempS/SensCal(1,2))*SensArrayIN(i);
            break
        elseif(E - SensCal(j,1) > 1e-6)
            % Interpolate from calibration curve
            tempS = SensCal(j-1,2)-((SensCal(j-1,1)-E)/...
                (SensCal(j-1,1)-SensCal(j,1)))*...
                (SensCal(j-1,2)-SensCal(j,2));
            SensArrayOUT(i) = (tempS/SensCal(1,2))*SensArrayIN(i);
        end
    end
end
```

---

fShockCalcs.m

```
function [ShockWaves] = fShockCalcs(ImpactVelocity,...
    Projectile,Wad,nWads,TargetMaterial,dimStrainSensors)
%=========================================================================
===
%Perform impact estimates using Hugoniot shock equations
% Inputs:
% strTestDate = File path location for test dates
% strTestCase = specific test case (i.e. 'Hand tools')
% ImpactVelocity = impact velocity in (ft/s)
% ProjectileLib = Mechanical data library for projectile cases
% TargetMaterial = target material data
% dimStrainSensors = Nx5 [Serial x y z theta] strain sensor location
data
%=========================================================================
===
% Reference:
% "Dynamic Behavior of Materials",Meyers
% "Explosives Engineering", Cooper
% -------------------------------------------------------------------------
---
% Map impact locations
% -------------------------------------------------------------------------
---
% Determine distance between impact and strain sensors
% Calculate the distance for the compression, shear, and Rayleigh waves
% resulting from impact
% [dimCompression dimShear dimRayleigh]
% Create a 4D array to capture dX info for each impact case and each
sensor
tempdimWaveDistance = zeros(size(Projectile.Locations,1),...
    size(dimStrainSensors,1),size(Projectile.Locations,2));
for i = 1:size(Projectile.Locations,1)
    for j = 1:size(dimStrainSensors,1)
        tempdimWaveDistance(i,j,:) = Projectile.Locations(i,:) - ...
            dimStrainSensors(j,2:4);
    end
end
if(size(Projectile.Locations,1)-1 < nWads)
    % Missing impact location for wad
    nWads = 0;
elseif(size(Projectile.Locations,1)-1 > nWads)
    % There are multiple impact locations not accounted for (multiple
    % projectiles possibly)
end
% Initialize arrray to hold path distance for each type wave
% dimCompression -> direct line
% dimShear -> direct line
% dimRayleigh -> surface distance on rear face opposite from impact loc
% Wave distance array = [impactCase sensorCase P-wave S-wave Rayleigh]
dimWave = zeros(size(tempdimWaveDistance));
for i = 1:size(Projectile.Locations,1)
    for j = 1:size(dimStrainSensors,1)
        % Calculate P-wave distance
        dimWave(i,j,1) = sqrt(tempdimWaveDistance(i,j,1)^2 + ...
            tempdimWaveDistance(i,j,2)^2 +
tempdimWaveDistance(i,j,3)^2);
```

```
        % Calculate S-wave distance
        dimWave(i,j,2) = dimWave(i,j,1);
        % Calculate Rayleigh wave distance
        dimWave(i,j,3) = sqrt(tempdimWaveDistance(i,j,1)^2 + ...
            tempdimWaveDistance(i,j,2)^2);
    end
end
% Convert dimensions from (in) to (m)
dimWave = dimWave * 0.0254;
% ------------------------------------------------------------------------
---
% Estimate pulse arrival at sensor due to impact
dtWave = zeros(size(dimWave));
% Calculate P-wave time to penetrate thickness of target - this
triggers
% Rayleigh waves on rear surface
dz_target = mean(dimStrainSensors(:,4)) * 0.0254;   % Convert to (m)
dt_z = dz_target / Projectile.Media.C_L;
for i = 1:size(Projectile.Locations,1)-nWads
    for j = 1:size(dimStrainSensors,1)
        % Calculate P-wave pulse dt using longitudinal wave velocity
        dtWave(i,j,1) = dimWave(i,j,1) / Projectile.Media.C_L;
        % Calculate S-wave pulse dt using shear wave velocity
        dtWave(i,j,2) = dimWave(i,j,2) / Projectile.Media.C_S;
        % Calculate Rayleigh wave distance
        dtWave(i,j,3) = dt_z + dimWave(i,j,3) / Projectile.Media.C_R;
    end
end
clearvars i j;
% Estimate pulse arrival at sensor due to wad impact
for i = size(Projectile.Locations,1)-
nWads+1:size(Projectile.Locations,1)
    for j = 1:size(dimStrainSensors,1)
        % Calculate P-wave distance
        dtWave(i,j,1) = dimWave(i,j,1) / Wad.C_L;
        % Calculate S-wave distance
        dtWave(i,j,2) = dimWave(i,j,2) / Wad.C_S;
        % Calculate Rayleigh wave distance
        dtWave(i,j,3) = dt_z + dimWave(i,j,3) / Wad.C_R;
    end
end
clearvars i j;
% ------------------------------------------------------------------------
---
% Estimate pressure at interface due to impact
% P_R = rho_0 * C_0 *(u_1 - u_0) + rho_0 * s * (u_1 - u_0)^2
% P_L = rho_0 * C_0 *(u_0 - u_1) + rho_0 * s * (u_0 - u_1)^2
% U = C_0 + s*u;
% Z = rho_0 * U:
P_impact = zeros(size(Projectile.Locations,1),1);
U_target = P_impact;
U_projectile = U_target;
% Rarefaction wave of target
% R_st = zeros(size(U_target,1),1);
% Initialize material properties of target and projectile
u_0ti = 0;                          % (m/s) initial target velocity
```

```
rho_0t = TargetMaterial.Density;
S_t = TargetMaterial.S;
C_0t = TargetMaterial.C_0;
u_0pi = ImpactVelocity * 0.3048;     % (m/s) initial projectile
velocity
rho_0p = Projectile.Media.Density;
S_p = Projectile.Media.S;
C_0p = Projectile.Media.C_0;
l_p = Projectile.Length * 0.0254;   % (m) Projectile length
syms u_1;
u_2 = u_0pi;
u_0p = u_0pi;
u_0t = u_0ti;
% Save shock EOS parameters
ShockEOS_p = [];
ShockEOS_t = [];
% Shock pulse shape in target
tempPulseShape = [];
PulseShape = [];
% Determine projectile impact parameters
for i = 1:size(Projectile.Locations,1)-nWads
    while(u_2 > 1e-3)
    % Determine interface pressure
%     eqn1 = TargetMaterial.Density * TargetMaterial.C_0 * ...
%         (u_1 - u_0t) + TargetMaterial.Density * TargetMaterial.S *
...
%         (u_1 - u_0t)^2;
%     eqn2 = Projectile.Media.Density * Projectile.Media.C_0 * ...
%         (u_0p - u_1) + Projectile.Media.Density * ...
%         Projectile.Media.S * (u_0p - u_1)^2;
    A = rho_0t*S_t - rho_0p*S_p;
    B = rho_0t*C_0t + rho_0p*C_0p - 2*u_0t*rho_0t*S_t +
2*u_0p*rho_0p*S_p;
    C = -rho_0t*C_0t*u_0t - rho_0p*C_0p*u_0p + rho_0t*S_t*u_0t^2 - ...
        rho_0p*S_p*u_0p^2;
    eqn = [A B C];
    eqnRoots = roots(eqn);
    error = (eqnRoots(:) - u_0p)/u_0p;
    u_1 = eqnRoots(abs(error)==min(abs(error)),:);% take most feasible
root
    % Determine interface Prs --> P=rho_0*C_0*(u_1-u_0)+rho_0*s*(u_1-
u_0)^2
    P_1p = rho_0p*C_0p*(u_0p-u_1)+rho_0p*S_p*(u_0p-u_1)^2;
    P_1t = rho_0t*C_0t*(u_1-u_0t)+rho_0t*S_t*(u_1-u_0t)^2;
    error = (P_1t - P_1p)/P_1p;
    if(error > 1e-6)
        fprintf('WARNING>>Interface pressure mismatch
in:\t%s',mfilename);
    else
        if(P_impact(i,1) < 1e-6)
            P_impact(i,1) = P_1t;
        end
    end
%     if(P_impact(i,1) - TargetMaterial.HugoniotElasticLimit > 1e-6)
%         % !!!!! FIX THIS !!!!!
%         % Transition to inelastic wave behavior
```

```
%         % C_p^2 = (sigma - HEL)/(rho_HEL*(strain_B - strain_HEL));
%     else
%         % Compare shock velocity with local sound speed
%         CU_ratio = U_target/TargetMaterial.C_0;
%     end
    % Determine shock velocity --> U = C_0 + S*u_1
    U_st = C_0t + S_t*u_1;
    if(U_target(i,1) < 1e-6)
        U_target(i,1) = C_0t + S_t*u_1;
    end
    % Determine shock velocity in projectile --> U = C_0 + S*u_1
    U_sp = C_0p + S_p*(u_0p - u_1);
    if(U_projectile(i,1) < 1e-6)
        U_projectile(i,1) = C_0p + S_p*(u_0p - u_1);
        U_projectile(i,1) = ((P_impact(i,1) - 0)/(u_1 - u_0p)) *
(1/rho_0p);
    end
    % ---------------------------------------------------------------
---
    % Rarefaction wave of projectile free surface -> shock pressure
profile
    % Iterate to drive pressure step-down to 0
    R_sp = abs((2*rho_0p*S_p*u_1 - rho_0p*C_0p - 2*rho_0p*S_p*u_0p)/...
        rho_0p);
    % Determine time for pulse from initial impact to rarefaction
leading edge
    t_Rpulse = l_p/U_sp + l_p/R_sp;
    % Calculate the distance into the target the shock pulse goes
before
    % dropping pressure to the secondary shock from the projectile Rar
wave
    x_Rpulse = U_st*t_Rpulse;
    % Determine particle velocity at projectile free face
    u_2 = u_1 - (u_0p - u_1);
    % Pressure at the free surface is zero
    % Determine seconday/step-down pressure from rarefaction wave at
    % projectile-target interface
    P_2 = 0;
    % Store Shock EOS for rarefaction step
    % Pulse shape is stepped
    ShockEOS_p = [ShockEOS_p; u_1, P_1p, U_sp; u_2, P_2, R_sp;];
    ShockEOS_t = [ShockEOS_t; u_1, P_1t, U_st;];
    % Store pulse shape values
    tempPulseShape = [tempPulseShape; x_Rpulse, P_1t, u_1;];
    % Reassign velocity for rarefaction step-down
    u_0p = u_2;
    end
    % Preserve pulse shape and initial impact particle velocity
    if(abs(i - 1) < 1e-6)
        tempPulseShape2 = tempPulseShape;
        u_2t = tempPulseShape2(1,3) - (u_0t - tempPulseShape2(1,3));
    end
    % Clear variables for successive impact iterations
    tempPulseShape = [];
    % Reset projectile velocity for second iteration
    u_2 = u_0pi;
```

```
    u_0p = u_0pi;
end
clear i;
% ------------------------------------------------------------------------
---
% Redefine pulse shape
k = 1;
for j = 1:size(tempPulseShape2,1)
    PulseShape(k,:) = [sum(tempPulseShape2(1:end-(j-1),1)) ...
        tempPulseShape2(j,2) tempPulseShape2(j,3)];
    PulseShape(k+1,:) = [sum(tempPulseShape2(1:end,1))-...
        tempPulseShape2(1,1) tempPulseShape2(j,2)
tempPulseShape2(j,3)];
    k = k+1;
end
clearvars k j;
% Close figure for plotting -> come back to first point
PulseShape(size(PulseShape,1)+1,:) = [PulseShape(end,1) 0 0];
PulseShape(size(PulseShape,1)+1,:) = [PulseShape(1,1) 0 0];
PulseShape(size(PulseShape,1)+1,:) = [PulseShape(1,1) PulseShape(1,2)
...
    PulseShape(1,3)];
% ------------------------------------------------------------------------
---
% Determine thickness of target -> pull from strain sensor location
% [x,y,z,theta]
z_t = dimStrainSensors(1,end-1);
% Determine maximum spall stress on reflected pressure pulse
% !!!!! Simplified model -> does not attenuate shock !!!!!
% Reference: "Explosives Engineering" Cooper, p. 245
% Reflected rarefaction Hugoniot
u_0R = u_2t;
u_0L = -Inf;        % Initial particle velocity of rarefaction Hugoniot
u_0t = u_0t;
F_spall = TargetMaterial.SpallStrength;
for i = 1:size(PulseShape,1)
    % Rarefaction wave of target free surface
    % Initiate reflection at rear target face
    u_0L = PulseShape(i+1,end);
    if(abs(i - size(PulseShape,1)) < 1e-6)
        break;
    end
    % P = rho_0*C_0*(u_0-u_1) + rho_0*S*(u_0-u_1)^2
    % dP/du = 2*rho_0*S*u - rho_0*C_0 - 2*rho_0*S*u_0 --> evaluate at
u_1,u_0
    R_st = (2*rho_0t*S_t*u_0R - rho_0t*C_0t - ...
        2*rho_0t*S_t*u_0t)/rho_0t;

%     A = rho_0t*S_t - rho_0t*S_t;
%     B = 2*rho_0t*S_t*(u_0R-u_0L);
%     C = rho_0t*C_0t*(u_0L-u_0R)+rho_0t*S_t*(u_0L^2-u_0R^2);
    B = (-2*rho_0t*S_t*u_0R - rho_0t*C_0t)-...
        (-2*rho_0t*S_t*u_0L + rho_0t*C_0t);
    C = (rho_0t*S_t*u_0R^2+rho_0t*C_0t*u_0R) - ...
        (rho_0t*S_t*u_0L^2-rho_0t*C_0t*u_0L);
%     eqn = [A B C];
```

```
%     eqnRoots = roots(eqn);
%     error = (eqnRoots(:) - u_0p)/u_0p;
%     u_1 = eqnRoots(abs(error)==min(abs(error)),:);% take most
feasible root
    u_1 = -C/B;
    if(u_1 < 1e-6)
        u_1 = abs(u_1);
    end
    % Determine interface Prs --> P=rho_0*C_0*(u_1-u_0)+rho_0*s*(u_1-
u_0)^2
    P_1L = rho_0t*C_0t*(u_0L-u_1)+rho_0t*S_t*(u_0L-u_1)^2;
    P_1LR = rho_0t*C_0t*(u_0R-u_1)+rho_0t*S_t*(u_0R-u_1)^2;
    error = (P_1t - P_1p)/P_1p;
    if(error > 1e-6)
        fprintf('WARNING>>Interface spall pressure mismatch
in:\t%s',mfilename);
        P_spall = abs(P_1LR);
    else
        P_spall = abs(P_1L);
    end
    % Determine safety factor for spalling
    SF_spall = F_spall/P_spall - 1;
    if(SF_spall < 1e-6)
        % Potential spall condition
        break;
    end
end
% ------------------------------------------------------------------
---
% Determine wad impact parameters
for i = size(Projectile.Locations,1)-
nWads+1:size(Projectile.Locations,1)
    % Determine interface pressure
    A = TargetMaterial.Density * TargetMaterial.S - ...
        Wad.Density * Wad.S;
    B = TargetMaterial.Density*TargetMaterial.C_0 + ...
        Wad.Density * Wad.C_0 - 2*u_0t*...
        TargetMaterial.Density*TargetMaterial.S + 2*u_0p*...
        Wad.Density*Projectile.Media.S;
    C = -TargetMaterial.Density*TargetMaterial.C_0*u_0t - ...
        Wad.Density*Projectile.Media.S*u_0p + ...
        TargetMaterial.Density*TargetMaterial.S*u_0t^2 - ...
        Wad.Density*Wad.S*u_0p^2;
    eqn = [A B C];
    eqnRoots = roots(eqn);
    error = (eqnRoots(:) - u_0p)/u_0p;
    u_1 = min(abs(error));           % take most feasible root
    % Determine interface Prs --> P=rho_0*C_0*(u_1-u_0)+rho_0*s*(u_1-
u_0)^2
    P_impact(i,1) = A*u_1^2 + B*u_1 + C;
%     if(P_impact(i,1) - TargetMaterial.HugoniotElasticLimit > 1e-6)
%         % !!!!! FIX THIS !!!!!
%         % Transition to inelastic wave behavior
%         % C_p^2 = (sigma - HEL)/(rho_HEL*(strain_B - strain_HEL));
%     else
%         % Compare shock velocity with local sound speed
```

```
%         CU_ratio = U_target/TargetMaterial.C_0;
%     end
    % Determine shock velocity in target --> U = C_0 + S*u_1
    U_target(i,1) = TargetMaterial.C_0 + TargetMaterial.S*u_1;
    % Determine shock velocity in projectile --> U = C_0 + S*u_1
    U_projectile(i,1) = Projectile.Media.C_0 + ...
        Projectile.Media.S*(u_0p - u_1);
    U_projectile(i,1) = ((P_impact(i,1) - 0)/(u_1 - u_0p))*...
        (1/Projectile.Media.Density);
end
% ------------------------------------------------------------------------
---
% Shock impedance --> Z = rho_0 * U
Z_t = TargetMaterial.Density*U_target;
% Z_p = Projectile.Media.Density*...
%     (Projectile.Media.C_0 + Projectile.Media.S*u_1);
Z_p = Projectile.Media.Density*U_projectile;

Z_tmax = max(Z_t,1);
Z_pmax = max(Z_p,1);
if(Z_tmax < Z_pmax)
    % Shock pulse shape has square front and stepped back
    strResponse = 'Z_t < Z_p';
    strPulseShape = 'Stepped';
elseif(abs(Z_tmax - Z_pmax) < 1e-6)
    % Shock pulse shape is square
    strResponse = 'Z_t = Z_p';
    strPulseShape = 'Square';
else
    % Shock pulse shape is square
    strResponse = 'Z_t > Z_p';
    strPulseShape = 'Square';
end



% ------------------------------------------------------------------------
---
% Output
ShockWaves = struct('Pressure',P_impact,'ShockVelocity',U_target,...
    'ElasticWave',dtWave,'ShockImpedance_Projectile',Z_p,...
    'ShockImpedance_Target',Z_t,'Z_cmp',strResponse,...
    'PulseShape',PulseShape,'EOS_projectile',ShockEOS_p,...
    'EOS_target',ShockEOS_t,'SF_spall',SF_spall);
end
```

---

fStrainToStress.m

```
function [PrincipalStress, PrimaryStress] =
fStrainToStress(Medium,StrainStruct,...
    StrainArray,ShearArray,StrRateArray,RosetteSerials)
```

```
%=========================================================================
===
%Convert principal strains to principal stresses - adapt dynamic
tensile
%strength using strain-rate data
% Inputs:
% Medium -> struct with material properties for static and dynamic
moduli
% elasticity and modulus of rigidity
% StrainStruct -> 1xM struct with strain data for each sensor
% StrainArray -> Nx2 array with primary strain data [eps_xx eps_yy]
% ShearArray -> Nx1 array with shear strain data [eps_xy]
% StrRateArray -> Nx2 array with strain-rate data [epsdot_xx epsdot_yy]
% RosetteSerials -> 3x1 array with serials for strain sensors
% Outputs:
% PrincipalStress -> struct with principal stress data
%=========================================================================
===
% Reference:
% "Behavior of plain concrete subjected to tensile loading at high
strain
% rate", E. Cadoni, C. Albertini, K. Labibes, G. Solomos
%
% -------------------------------------------------------------------
---
% Test properties
% -------------------------------------------------------------------
---
% Concrete Mix;                       Average properties;
% Aggregate:  1880                    Static elastic modulus:     43.1
GPa
% CEM I 42.5: 395                     Dynamic elastic modulus:    54.8
GPa
% Superplasticizer:   6.5             Vibration elastic modulus:  41.4
GPa
% Water:              201             Compressive strength:       46.2
MPa
% Volumetric mass:    2482            Fracture energy, G_F:       147.6
N/m
%                                     Flexural tensile strength:  4.4
MPa
% -------------------------------------------------------------------
---
% Assumed uniaxial state of stress since pulse wavelength (100 m) >>
% transverse length of the test bar (~0.3m)
% -------------------------------------------------------------------
---
% Strain rate effects:
% -------------------------------------------------------------------
---
% Strain-rate          Tensile Str (5mm agg)       Tensile Str (10mm
agg)
% [s^-1]               [MPa]                       [MPa]
% 10                   11.8                        10.5
% 1                    9.2                         7.4
% 10^-2                2.5                         4.4
```

```
% 10^-6                2.7                          3.6
% ------------------------------------------------------------------------
---
% From Figure 4. Stress vs Strain curves for two strain-rates
% ------------------------------------------------------------------------
---
% Strain-rate 1 [s^-1],     E = 67e9 [Pa]
% Strain-rate 10 [s^-1],    E = 78.67e9 [Pa]
% Or an average dynamic modulus of 54.8 GPa for both strain-rates
% ------------------------------------------------------------------------
---

% Determine strain rate bins
limStrainRate = [10e-6; 10e-2; 1; 10;];
iSR_static = limStrainRate(2,1);
% Strain -> StrainRate arrays with correlator array
% ID_array = 1:size(StrainArray,1);
% ID_array = transpose(ID_array);
ID_array = StrainStruct(1).Data(:,1);

zzArray = zeros(size(StrainArray,1),1);
% tempStress = struct('LimitStrainRate',limStrainRate(1),...
%     'Strain_StrainRate',[ID_array StrainArray ShearArray
StrRateArray]);
tempStress = struct('LimitStrainRate',limStrainRate(1),...
    'Strain_StrainRate',[ID_array StrainArray zzArray ShearArray
StrRateArray]);
% Generate index for strain-rate comparisons
ilim = size(limStrainRate,1)*size(StrRateArray,2);
% Generate index for strain and strain-rate location
cEPS = 2;
cEPSzz = 4;
cGAMMA = 5;
cSR = size(tempStress(1).Strain_StrainRate,2)-1;
j = 1;
k = 0;
for i = 2:ilim
    if(i < ilim/2 + 1)
        j = j + 1;
        tempStress(i).LimitStrainRate = limStrainRate(j);
        tempStress(i).Strain_StrainRate = ...
            [ID_array StrainArray zzArray ShearArray StrRateArray];
    else
        k = k + 1;
        tempStress(i).LimitStrainRate = limStrainRate(k);
        tempStress(i).Strain_StrainRate = ...
            [ID_array StrainArray zzArray ShearArray StrRateArray];
    end
end
clearvars i j k;
% ------------------------------------------------------------------------
---
% Eliminate elements from strain bins not within strain-rate limits
% Blend the transition from bin to bin
templim = zeros(size(limStrainRate,1)-1,1);
```

```
templim(1) = (limStrainRate(2) - limStrainRate(1))/2;
templim(2) = (limStrainRate(3) - limStrainRate(2))/2;
templim(3) = (limStrainRate(4) - limStrainRate(3))/2;

j = 0;
k = 0;
for i = 1:ilim
    if(i < ilim/2 + 1)
        if(abs(i - 1) < 1e-6)
            j = j + 1;
            tempStress(i).Strain_StrainRate( ...
                abs(tempStress(i).Strain_StrainRate(:,cSR)) >
templim(j),:,:) = [];
        elseif(abs(i - ilim/2) < 1e-6)
            j = size(templim,1);
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR)) <
templim(j),:,:) = [];
        else
            j = j + 1;
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR)) >
templim(j)...
                ,:,:) = [];
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR)) <
templim(j-1)...
                ,:,:) = [];
        end
    else
        % Reset indices for strain-rate limit checks -> check 2nd SR
array
        k = i - ilim/2;
        if(abs(k - 1) < 1e-6)
            j = k;
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR+1)) >
templim(j),:,:) = [];
        elseif(abs(k - ilim/2) < 1e-6)
            j = size(templim,1);
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR+1)) <
templim(j),:,:) = [];
        else
            j = j + 1;
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR+1)) >
templim(j)...
                ,:,:) = [];
            tempStress(i).Strain_StrainRate(...
                abs(tempStress(i).Strain_StrainRate(:,cSR+1)) <
templim(j-1)...
                ,:,:) = [];
        end
    end
end
```

```
clearvars i j k;
% -------------------------------------------------------------------
---
% Retrieve Dynamic Elasticity Moduli and Modulus of Rigidity
E_stc = Medium.ElasticityModulus;
E_dyn = Medium.ElasticityModulusDynamic;
% G = Medium.ShearModulus;
% -------------------------------------------------------------------
---
% Convert strain to stress
% ref: "Advanced Mechanics of Material", Boresi & Schmidt (p.87)
% For thick-plate assumption, plane strain -> eps_zz = eps_xz = eps_yz
= 0
% For plane strain - the displacement components (u,v) are functions of
% (x,y) only and w = constant -> eps_zz = dw/dz
% sigma_xx = (E/((1+nu)*(1-2*nu))*((1-nu)*eps_xx + nu*eps_yy);
% sigma_yy = (E/((1+nu)*(1-2*nu))*(nu*eps_xx + (1-nu)*eps_yy);
% sigma_zz = (nu*E/((1+nu)(1-2*nu)))*(eps_xx + eps_yy);
% sigma_xy = (E/(1+nu))*eps_xy
nu = Medium.PoissonRatio;
for i = 1:size(tempStress,2)
    if(~isempty(tempStress(i).Strain_StrainRate))
        tempSigma = tempStress(i).Strain_StrainRate(:,cEPS:cGAMMA);
        if(tempStress(i).LimitStrainRate - iSR_static < 1e-6)
            C1 = E_stc/((1+nu)*(1-2*nu));           % sigma_xx and
sigma_yy
            C2 = E_stc/(1+nu);                      % sigma_xy
            C3 = nu*E_stc/((1+nu)*(1-2*nu));        % sigma_zz
            % sigma_xx and sigma_yy
            tempSigma(:,1) = C1 * ((1-nu)*...
                tempStress(i).Strain_StrainRate(:,cEPS) + ...
                nu*tempStress(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,2) = C1 *
(nu*tempStress(i).Strain_StrainRate(:,cEPS) + (...
                1-nu)*tempStress(i).Strain_StrainRate(:,cEPS+1));
            % sigma_zz
            tempSigma(:,3) = C3 * ...
                (tempStress(i).Strain_StrainRate(:,cEPS) + ...
                tempStress(i).Strain_StrainRate(:,cEPS+1));
            % sigma_xy
            tempSigma(:,4) = C2 * ...
                tempStress(i).Strain_StrainRate(:,cGAMMA);
        else
            C1 = E_dyn/((1+nu)*(1-2*nu));           % sigma_xx and
sigma_yy
            C2 = E_dyn/(1+nu);                      % sigma_xy
            C3 = nu*E_dyn/((1+nu)*(1-2*nu));        % sigma_zz
            % sigma_xx and sigma_yy
            tempSigma(:,1) = C1 * ((1-nu)*...
                tempStress(i).Strain_StrainRate(:,cEPS) + ...
                nu*tempStress(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,2) = C1 * (nu*...
                tempStress(i).Strain_StrainRate(:,cEPS) + (...
                1-nu)*tempStress(i).Strain_StrainRate(:,cEPS+1));
            % sigma_zz
            tempSigma(:,3) = C3 * ...
```

```
                (tempStress(i).Strain_StrainRate(:,cEPS) + ...
                tempStress(i).Strain_StrainRate(:,cEPS+1));
            % sigma_xy
            tempSigma(:,4) = C2 * ...
                tempStress(i).Strain_StrainRate(:,2);
        end
        tempStress(i).Strain_StrainRate(:,cEPS:cGAMMA) = tempSigma;
    end
end
clearvars tempSigma C1 C2 i;
% ------------------------------------------------------------------------
---
% Organize primary stress arrays
PrimaryStress = struct('LimitStrainRate',limStrainRate(1),...
    'sigma_xx',tempStress(1).Strain_StrainRate(:,cEPS),...
    'sigma_yy',tempStress(1).Strain_StrainRate(:,cEPS+1),...
    'sigma_zz',tempStress(1).Strain_StrainRate(:,cEPSzz),...
    'sigma_xy',tempStress(1).Strain_StrainRate(:,cGAMMA));
% PrimaryStress = struct('LimitStrainRate',limStrainRate(1),...
%     'sigma_xx',tempSigma(:,1),...
%     'sigma_yy',tempSigma(:,2),...
%     'sigma_zz',tempSigma(:,3),...
%     'sigma_xy',tempSigma(:,4));
for i = 1:size(tempStress,2)/2
    PrimaryStress(i).LimitStrainRate = tempStress(i).LimitStrainRate;
    PrimaryStress(i).sigma_xx =
tempStress(i).Strain_StrainRate(:,cEPS);
    PrimaryStress(i).sigma_yy =
tempStress(i).Strain_StrainRate(:,cEPS+1);
    PrimaryStress(i).sigma_zz =
tempStress(i).Strain_StrainRate(:,cEPSzz);
    PrimaryStress(i).sigma_xy =
tempStress(i).Strain_StrainRate(:,cGAMMA);
%     PrimaryStress(i).LimitStrainRate = tempStress(i).LimitStrainRate;
%     PrimaryStress(i).sigma_xx = tempSigma(:,1);
%     PrimaryStress(i).sigma_yy = tempSigma(:,2);
%     PrimaryStress(i).sigma_zz = tempSigma(:,3);
%     PrimaryStress(i).sigma_xy = tempSigma(:,4);
end
clearvars tempStress i;
% ------------------------------------------------------------------------
---
% Convert strain to stress for each channel except rosette
tempEpsilonArray = [];
% Pull serial numbers of rosette sensors
Serial_A = RosetteSerials(1,1);
Serial_B = RosetteSerials(2,1);
Serial_C = RosetteSerials(3,1);
% Sensor angles
tempSensorAngles = [];
for i = 1:size(StrainStruct,2)
    if(abs(StrainStruct(i).Serial - Serial_A) < 1e-6 || ...
            abs(StrainStruct(i).Serial - Serial_B) < 1e-6 || ...
            abs(StrainStruct(i).Serial - Serial_C) < 1e-6)
    else
        % Populate strain arrays and concatenate with successive data
```

```
        if(isempty(tempEpsilonArray))
            tempEpsilonArray = StrainStruct(i).Data;
        else
            tempEpsilonArray = [tempEpsilonArray
StrainStruct(i).Data(:,2)];
        end
        % Populate sensor angle array
        tempSensorAngles = [tempSensorAngles
StrainStruct(i).SensorAngle];
    end
end
clearvars i;
% Calculate strain-rate for each strain data set
tempSRArray = zeros(size(tempEpsilonArray,1),size(tempEpsilonArray,2)-
1);
for i = 2:size(tempEpsilonArray,2)
    for j = 2:size(tempSRArray,1)
        tempSRArray(j,i-1) = (tempEpsilonArray(j,i) - ...
            tempEpsilonArray(j-1,i)) / ...
            (tempEpsilonArray(j,1) - tempEpsilonArray(j-1,1));
    end
end
tempEpsilonArray = [tempEpsilonArray tempSRArray];
clearvars tempSRArray i j;
% Initialize column indices for strain and strain-rate
cEPS_2 = 2;
cSR_2 = 5;
% Initialize strain struct
tempStrain = struct('LimitStrainRate',limStrainRate(1),...
    'Strain_StrainRate',tempEpsilonArray,'SensorAngle',0);
for i = 1:size(limStrainRate,1)
    tempStrain(i).LimitStrainRate = limStrainRate(i);
    tempStrain(i).Strain_StrainRate = tempEpsilonArray;
    tempStrain(i).SensorAngle = tempSensorAngles;
end
clearvars tempEpsilonArray tempSensorAngles;
% Eliminate strains outside of strain-rate bins for all channels
k = 1;
for j = (size(tempStrain(1).Strain_StrainRate,2)-1)/2+2:...
        (size(tempStrain(1).Strain_StrainRate,2))
    for i = 1:size(tempStrain,2)
        if(abs(k - 1) < 1e-6)
            tempStrain(i).Strain_StrainRate(...
                abs(tempStrain(i).Strain_StrainRate(:,j)) >
templim(k)...
                ,:,:) = [];
            k = k+1;
        elseif(k - size(templim,1) > 1e-6)
            k = size(templim,1);
            tempStrain(i).Strain_StrainRate(...
                abs(tempStrain(i).Strain_StrainRate(:,j)) <
templim(k)...
                ,:,:) = [];
        else
            tempStrain(i).Strain_StrainRate(...
```

```
                abs(tempStrain(i).Strain_StrainRate(:,j)) >
templim(k)...
                ,:,:) = [];
            tempStrain(i).Strain_StrainRate(...
                abs(tempStrain(i).Strain_StrainRate(:,j)) < templim(k-
1)...
                ,:,:) = [];
            k = k + 1;
        end
    end
    k = 1;
end
clearvars i j k;
% Initialize stress struct
tempStress = struct('LimitStrainRate',limStrainRate(1),...
    'Stress',0);
% Check whether to transform strain
for ID_SA = 1:size(tempStrain(1).SensorAngle,2)
    bTransformStrain = abs(tempStrain(1).SensorAngle(ID_SA) - 0) > 1e-
6;
    if(bTransformStrain)
        break
    end
end
if(bTransformStrain)
% Transform strain from measured axis to primary axis
syms eps_x eps_y eps_xy;
% Initialize Reuter's matrix for strain transformations
% [eps_x';eps_y';eps_xy';] = R*A*R^-1*[eps_x;eps_y;eps_xy;];
R = [1,0,0;0,1,0;0,0,2;];
% Initialize transformed strain struct
tempStrainTransformed = tempStrain;
for i = 1:size(tempStrain,2)
    % Initialize index for inserting derived strain arrays
    l = 1;
    % Assign stress struct
    tempStress(i).LimitStrainRate = tempStrain(i).LimitStrainRate;
    % Assign time array for indexing
    tempStress(i).Stress = tempStrain(i).Strain_StrainRate(:,1);
    if(~isempty(tempStrain(i).Strain_StrainRate))
%         tempSigma = tempStrain(i).Strain_StrainRate(:,cEPS_2:cSR_2-
1);
        tempSigma = tempStrain(i).Strain_StrainRate(:,cEPS:cGAMMA);
        for j = cEPS_2:cSR_2-1
            % Initialize index for sensor angle
            j2 = j - 1;
            % Convert strain to principal axis
            thS = tempStrain(i).SensorAngle(j2) * pi()/180;
            % Initialize transformation matrix
            A = [cos(thS)^2, sin(thS)^2, 2*sin(thS)*cos(thS);
                sin(thS)^2, cos(thS)^2, -2*sin(thS)*cos(thS);
                -sin(thS)*cos(thS), sin(thS)*cos(thS),cos(thS)^2-
sin(thS)^2;];
            eqns = R*A*(R^(-1));
            for k = 1:size(tempStrain(i).Strain_StrainRate,1)
```

```
                eqn1 =
eqns(1,1)*tempStrain(i).Strain_StrainRate(k,j)+...
                    eqns(1,2)*0 + eqns(1,3)*0 == eps_x;
                eqn2 =
eqns(2,1)*tempStrain(i).Strain_StrainRate(k,j)+...
                    eqns(2,2)*0 + eqns(2,3)*0 == eps_y;
                eqn3 =
eqns(3,1)*tempStrain(i).Strain_StrainRate(k,j)+...
                    eqns(3,2)*0 + eqns(3,3)*0 == eps_xy;
                [A1,B1] = equationsToMatrix([eqn1,eqn2,eqn3],...
                    [eps_x,eps_y,eps_xy]);
                C1 = linsolve(A1,B1);
                tempEpsilon(k,1) = C1(1);
                tempEpsilon(k,2) = C1(2);
                tempEpsilon(k,3) = C1(3);
            end
            % Insert derived eps_x eps_y eps_xy arrays
            tempStrainTransformed(i).Strain_StrainRate = ...
                [tempStrain(i).Strain_StrainRate(:,1:l) ...
                tempEpsilon tempStrain(i).Strain_StrainRate(:,l:end)];
            l = l + 3;
        end
        if(tempStrain(i).LimitStrainRate - iSR_static < 1e-6)
            C1 = E_stc/((1+nu)*(1-2*nu));
            C2 = E_stc/(1+nu);
            C3 = nu*E_stc/((1+nu)*(1-2*nu));        % sigma_zz
            tempSigma(:,1) = C1 * ((1-nu)*...
                tempStrain(i).Strain_StrainRate(:,cEPS) + ...
                nu*tempStrain(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,2) = C1 * (nu*...
                tempStrain(i).Strain_StrainRate(:,cEPS) + (1-nu)*...
                tempStrain(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,3) = C3 * ...
                (tempStrain(i).Strain_StrainRate(:,cEPS) + ...
                tempStrain(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,4) = C2 * ...
                tempStrain(i).Strain_StrainRate(:,cGAMMA);
        else
            C1 = E_dyn/((1+nu)*(1-2*nu));           % sigma_xx sigma_yy
            C2 = E_dyn/(1+nu);                      % sigma_xy
            C3 = nu*E_dyn/((1+nu)*(1-2*nu));        % sigma_zz
            tempSigma(:,1) = C1 * ((1-nu)*...
                tempStrain(i).Strain_StrainRate(:,cEPS) + ...
                nu*tempStrain(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,2) = C1 * (nu*...
                tempStrain(i).Strain_StrainRate(:,cEPS) + (...
                1-nu)*tempStrain(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,3) = C3 * ...
                (tempStrain(i).Strain_StrainRate(:,cEPS) + ...
                tempStrain(i).Strain_StrainRate(:,cEPS+1));
            tempSigma(:,4) = C2 * ...
                tempStrain(i).Strain_StrainRate(:,2);
        end
        tempStress(i).Stress = [tempStress(i).Stress tempSigma];
    end
end
```

```
clearvars i j k l;
else
    % Convert strain to stress with single strain variable
    tempStress2 = [];
    for i = 1:size(tempStrain,2)
        % Assign stress struct
        tempStress(i).LimitStrainRate = tempStrain(i).LimitStrainRate;
        % Assign time array for indexing
        tempStress(i).Stress = tempStrain(i).Strain_StrainRate(:,1);
        if(~isempty(tempStrain(i).Strain_StrainRate))
            tempSigma = tempStrain(i).Strain_StrainRate(:,cEPS_2:cSR_2-
1);
            for j = cEPS_2:cSR_2-1
                %Convert strain to stress
            if(tempStrain(i).LimitStrainRate - iSR_static < 1e-6)
                C1 = E_stc/((1+nu)*(1-2*nu));
                C2 = E_stc/(1+nu);
                C3 = nu*E_stc/((1+nu)*(1-2*nu));        % sigma_zz
                tempSigma(:,1) = C1 * ((1-nu)*...
                    tempStrain(i).Strain_StrainRate(:,j) + ...
                    nu*(0));
                tempSigma(:,2) = C1 * (nu*...
                    tempStrain(i).Strain_StrainRate(:,j) + (1-nu)*(0));
                % sigma_zz = sigma_yy with single variable conversion
                % !!!!!! FIX THIS !!!!!
%                 tempSigma(:,3) = C3 * ...
%                     (tempStrain(i).Strain_StrainRate(:,j) + ...
%                     0);
%                 tempSigma(:,4) = C2 * (0);
                tempSigma(:,3) = C2 * (0);
            else
                C1 = E_dyn/((1+nu)*(1-2*nu));
                C2 = E_dyn/(1+nu);
                C3 = nu*E_dyn/((1+nu)*(1-2*nu));        % sigma_zz
                tempSigma(:,1) = C1 * ((1-nu)*...
                    tempStrain(i).Strain_StrainRate(:,j) + ...
                    nu*(0));
                tempSigma(:,2) = C1 * (nu*...
                    tempStrain(i).Strain_StrainRate(:,j) + (...
                    1-nu)*(0));
%                 tempSigma(:,3) = C3 * ...
%                     (tempStrain(i).Strain_StrainRate(:,j) + ...
%                     0);
%                 tempSigma(:,4) = C2 * (0);
                tempSigma(:,3) = C2 * (0);
            end
            tempStress2 = [tempStress2 tempSigma];
            end
        end
        tempStress(i).Stress = [tempStress(i).Stress tempStress2];
        tempStress2 = [];
    end
end
clearvars cEPS_2 cSR_2 ID_SA i j;
% ------------------------------------------------------------------------
---
```

```
% Determine principal stresses
% ref:  "Advanced Mechanics of Materials",Boresi & Schmidt
% l*(sigma_xx - sigma) + m*sigma_xy + n*sigma_xz = 0;
% l*sigma_xy + m*(sigma_yy - sigma) + n*sigma_yz = 0;
% l*sigma_xz + m*sigma_yz + n*(sigma_zz - sigma) = 0;
% sigma^3 - I_1*sigma_2 + I_2*sigma - I_3 = 0;
% I_1 = sigma_xx + sigma_yy + sigma_zz;
% I_2 = sigma_xx*sigma_yy + sigma_xx*sigma_zz + sigma_yy*sigma_zz -
% sigma_xy^2 - sigma_xz^2 - sigma_yz^2;
% I_3 = det(sigma_xx,sigma_xy,sigma_xz; sigma_xy,sigma_yy,sigma_yz;
% sigma_xz,sigma_yz,sigma_zz);
% For plane strain assumption:  sigma_x, sigma_y, sigma_z, sigma_xy may
be
% non-zero
% Solve for stress invariants:
syms l m n;                             % Direction cosines for principal
axes
% tempPrincipalStress = struct('LimitStrainRate',1,...
%     'sigma_1',0,'sigma_2',0,'pTheta',0);
tempPrincipalStress = struct('LimitStrainRate',1,...
    'sigma_1',0,'sigma_2',0,'sigma_3',0,'pTheta',0);
for i = 1:size(PrimaryStress,2)
%     Define primary stress arrays
    xx = PrimaryStress(i).sigma_xx;
    yy = PrimaryStress(i).sigma_yy;
    zz = PrimaryStress(i).sigma_zz;
    xy = PrimaryStress(i).sigma_xy;
%     Pre-allocate principal stress arrays and principal axes
    sigma_1 = xx;
    sigma_2 = yy;
    sigma_3 = zz;
    pTheta = zeros(size(xx,1),2);
%     I_1 = xx + yy;
    I_1 = xx + yy + zz;
%     I_2 = xx.*yy - xy.^2;
    I_2 = xx.*yy + xx.*zz + yy.*zz - xy.^2;
    % Find invariant using determinant of simplified matrix
    I_3 = zeros(size(xx,1),1);
    for j = 1:size(xx,1)
%         I_3(j) = det([xx(j),xy(j);xy(j),yy(j)]);
        I_3(j) = det([xx(j),xy(j),0;xy(j),yy(j),0;0,0,zz(j)]);
%         Find roots to sigma^3 - I_1*sigma^2 + I_2*sigma - I_3 = 0
%         Roots are principal stresses sigma_1, sigma_2, sigma_3
        tempSigma = [1 -I_1(j) I_2(j) I_3(j)];
        tempSigmaRoots = roots(tempSigma);
        sigma_1(j) = real(tempSigmaRoots(1));
        sigma_2(j) = real(tempSigmaRoots(2));
        sigma_3(j) = real(tempSigmaRoots(3));
        % Determine principal axes for sigma_1 sigma_2

        eqn1 = l*(xx(j)-sigma_1(j)) + m*xy(j) + n*0 == 0;
        eqn2 = l*xy(j) + m*(yy(j)-sigma_1(j)) + n*0== 0;
        eqn3 = l*0 + m*0 + n*(zz(j)-sigma_1(j)) == 0;
%         [A,B] = equationsToMatrix([eqn1,eqn2],[l,m]);
        [A,B] = equationsToMatrix([eqn1,eqn2,eqn3],[l,m,n]);
%         C = linsolve(A,B);
```

```
        C = linsolve(A,B);
        % l and m are direction cosines of x' and y' axes respectively
        pTheta(j,1) = acos(C(1));   % Convert direction cosine to
radians
        pTheta(j,2) = acos(C(2));
        pTheta(j,:) = pTheta(j,:) * 180/pi();   % Convert to degrees
        if(abs(90 - pTheta(j,1)) < 1e-1)
            pTheta(j,1) = 0;
        end
        if(abs(90 - pTheta(j,2)) < 1e-1)
            pTheta(j,2) = 0;
        end
    end
    tempPrincipalStress(i).LimitStrainRate =
PrimaryStress(i).LimitStrainRate;
    tempPrincipalStress(i).sigma_1 = sigma_1;
    tempPrincipalStress(i).sigma_2 = sigma_2;
    tempPrincipalStress(i).sigma_3 = sigma_3;
    tempPrincipalStress(i).pTheta = pTheta;
end
% ------------------------------------------------------------------------
---
% Convert each sensor strain to principal stress
for i = 1:size(tempPrincipalStress,2)
    bTransformStrain = abs(tempPrincipalStress(i).pTheta - 0) > 1e-6;
    if(bTransformStrain)
        break
    end
end
clearvars i;
% Initialize principal strain struct
PrincipalStress = struct('LimitStrainRate',1,...
    'sigma_1',0,'sigma_2',0,'sigma_3',0,'pTheta',0,'StressChannel',0);
if(bTransformStrain)
    error('INSERT STRAIN TRANSFORMATION FOR PRINCIPAL STRESS')
else
    for i = 1:size(tempPrincipalStress,2)
        PrincipalStress(i).LimitStrainRate =
tempPrincipalStress(i).LimitStrainRate;
        PrincipalStress(i).sigma_1 = tempPrincipalStress(i).sigma_1;
        PrincipalStress(i).sigma_2 = tempPrincipalStress(i).sigma_2;
        PrincipalStress(i).sigma_3 = tempPrincipalStress(i).sigma_3;
        PrincipalStress(i).pTheta = tempPrincipalStress(i).pTheta;
        PrincipalStress(i).StressChannel = tempStress(i).Stress;
    end
end
end
```

---

fTestCaseData.m

```
function [TestCaseData] = fTestCaseData(MediaLibrary)
```

```
%=========================================================================
===
%Set impact conditions for case, material, projectile mass, impact
%location etc.
%=========================================================================
===

% Initialize struct
ImpactLoc =
struct('Case','NaN','Media',0,'nProjectiles',0,'Mass',0,'Diameter',0,..
.
    'Length',0,'OgiveRadius',0,'Locations',[0,0,0],'VolumeLoss',0,...
    'CraterDiameters',0,'PenetrationMeasured',0);
nTestCases = 16;
ImpactLoc_ = ImpactLoc;
for i = 1:nTestCases
    ImpactLoc(i) = ImpactLoc_;
end

% Set impact cases
% -------------------------------------------------------------------
---
% HARD HAT
strTestCase = 'Hard hat';
mMedia = MediaLibrary(2);
dMass = 1;                          % (lb)
Diameter = 8;                       % (in)
Length = 5;                         % (in)
OgiveRadius = 8;                    % (in)
VolumeLoss = 0;                     % (in^3) - measured volume loss
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [5.25*12, 5.125*12, 0;
    6.9*12, 4.125*12, 0;
    3.8*12, 2.8*12, 0;
    7.75*12, 1.9*12,0;];
dimCrater = zeros(size(dimImpact,1),1);
dimPenetration = 0;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(1) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact;
% -------------------------------------------------------------------
---
% WATER JUG
strTestCase = 'Water jug';
mMedia = MediaLibrary(2);
dMass = 1;                          % (lb)
```

```
Diameter = 6;                           % (in)
Length = 12;                            % (in)
OgiveRadius = 1.75;                     % (in) - estimated flat
VolumeLoss = 0;                         % (in^3) - actual penetration
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [7.125*12, 4.8*12, 0;
    7.125*12, 2.6*12, 0;
    8.5*12, 0.5*12, 0;];
dimCrater = zeros(size(dimImpact,1),1);
dimPenetration = 0;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(2) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact;
% ------------------------------------------------------------------------
---
% HAND TOOLS
strTestCase = 'Hand tools';
mMedia = MediaLibrary(4);
dMass = 9;                              % (lb)
Diameter = 6;                           % (in)
Length = 2;                             % (in)
OgiveRadius = 0.25*Diameter;            % (in)
VolumeLoss = 0.170376*61.0237;          % (L) -> (in^3)
nProjectiles = 3;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [9.3*12, 4.125*12, 0;
    9.125*12, 4.0*12, 0;
    9.15*12, 4.25*12, 0;
    6.5*12, 4.125*12, 0;
    6.125*12, 0.75*12, 0;];
dimCrater = [5.1703;
    5.1703;
    0;
    0;];
dimPenetration = 0.358;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(3) = ImpactLoc_;
```

```
clearvars strTestCase mMedia dMass dimImpact;
% ------------------------------------------------------------------------
---
% ROOF BOLT PLATES
strTestCase = 'Roof bolt plates';
mMedia = MediaLibrary(4);
dMass = 17.5;                           % (lb)
Diameter = 4;                           % (in) - length of plates
Length = 2;                             % (in) - thickness of plates
Thickness = 0.125;
OgiveRadius = 0.25*Diameter;            % (in) - estimated flat
VolumeLoss = 0.058931*61.0237;          % (L) -> (in^3)
nProjectiles = 5;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [8.0*12, 3.45*12, 0;
    7.75*12, 2.95*12, 0;
    7.5*12, 2.5*12, 0;
    8.125*12, 3.9*12, 0;
    8.125*12, 3.9*12, 0;];
dimCrater = [7.868;
    6.969;
    4.496;
    0;
    0;];
dimPenetration = 0.361;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(4) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact;
% ------------------------------------------------------------------------
---
% 5X5 LUMBER
strTestCase = '5x5 lumber';
mMedia = MediaLibrary(5);
dMass = 30;                             % (lb)
Diameter = 4.5;                         % (in)
Length = 60;                            % (in)
OgiveRadius = 1.25;                     % (in) - estimated flat
VolumeLoss = 0;                         % (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [4.0625*12, 1.75*12, 0;
    2.75*12, 1.45*12, 0;];
dimCrater = zeros(size(dimImpact,1),1);
dimPenetration = 0;
```

```
ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(5) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact;
% ------------------------------------------------------------------------
---
% ROOF BOLTS
strTestCase = 'Roof bolts';
mMedia = MediaLibrary(4);
dMass = 17;                         % (lb)
Diameter = 0.75;                    % (in)
Length = 18;                        % (in)
OgiveRadius = 0.25*Diameter;        % (in) - estimated flat
VolumeLoss = 0.010958*61.0237;      % (L) -> (in^3)
nProjectiles = 4;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [4.8*12, 0.3*12, 0;
    4.125*12, 2.25*12, 0;
    3.85*12, 0.9*12,0;
    4.0625*12, 3.125*12, 0;];
dimCrater = [2.136;
    1.461;
    0;
    0;];
dimPenetration = 0.121;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(6) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact;
% ------------------------------------------------------------------------
---
% CONCRETE (10LB)
strTestCase = 'Concrete (10lb)';
mMedia = MediaLibrary(1);
dMass = 10;                         % (lb)
Diameter = 10;                      % (in)
OgiveRadius = 2.5;                  % (in) - estimated flat
Length = 2;                         % (in)
VolumeLoss = 0;                     % (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
```

```
dimImpact = [3.75*12, 0.5*12, 0;
    3.9*12, 1.75*12, 0;];
dimCrater = zeros(size(dimImpact,1),1);
dimPenetration = 0;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(7) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------
---
% CONCRETE (20LB)
strTestCase = 'Concrete (20lb)';
mMedia = MediaLibrary(1);
dMass = 20;                         % (lb)
Diameter = 8;                       % (in)
OgiveRadius = 2;                    % (in) - estimated flat
Length = 4.35;                      % (in)
VolumeLoss = 0;                     % (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [5.2*12, 5*12, 0;
    6.9*12, 5.45*12, 0;];
dimCrater = zeros(size(dimImpact,1),1);
dimPenetration = 0;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(8) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------
---
% CONCRETE (30LB)
strTestCase = 'Concrete (30lb)';
mMedia = MediaLibrary(1);
dMass = 30;                         % (lb)
Diameter = 8;                       % (in)
OgiveRadius = 2;                    % (in) - estimated flat
Length = 6.52;                      % (in)
VolumeLoss = 0;                     % (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [4.8*12, 3.33*12, 0;
```

```
    5.4*12, 2.25*12, 0;];
dimCrater = zeros(size(dimImpact,1),1);
dimPenetration = 0;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(9) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------
---
% STEEL PENETRATOR (10LB)
strTestCase = 'Steel penetrator (10lb)';
mMedia = MediaLibrary(4);
dMass = 10;                         % (lb)
Diameter = 3.5;                     % (in)
OgiveRadius = 7;                    % (in) - estimated using 45 deg
chamfer
Length = 4.5;                       % (in)
Diameter_Meplat = 0.5;              % (in)
VolumeLoss = 0.669*61.0237;         % (L) -> (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [5.125*12, 3.45*12, 0;
    5.2*12, 3.5*12, 0;];
dimCrater = [11.015;
    0;];
dimPenetration = 1.224;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(10) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------
---
% STEEL RAIL (25LB)
strTestCase = 'Steel rail (25lb)';
mMedia = MediaLibrary(4);
dMass = 25;                         % (lb)
Diameter = 5.5;                     % (in)
OgiveRadius = 7;                    % (in) - estimated using 45 deg
chamfer
Length = 36;                        % (in)
Diameter_Meplat = 0.5;              % (in)
VolumeLoss = 0.018144*61.0237;      % (L) -> (in^3)
nProjectiles = 1;
```

```
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [5.6*12, 0.4*12, 0;
    7.9*12, 5*12, 0;
    5.95*12, 3.3*12, 0;
    6.4*12, 2.8*12, 0;];
dimCrater = [2.014625;
    0;
    0;
    0;];
dimPenetration = 0.342;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(11) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% ------------------------------------------------------------------
---
% STEEL PENETRATOR (20LB)
strTestCase = 'Steel penetrator (20lb)';
mMedia = MediaLibrary(4);
dMass = 22;                         % (lb)
Diameter = 3.5;                     % (in)
OgiveRadius = 7;                    % (in) - estimated using 45 deg
chamfer
Length = 8.92;                      % (in)
Diameter_Meplat = 0.5;              % (in)
VolumeLoss = 0.167158*61.0237;      % (L) -> (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [5.4*12, 1.3*12, 0;
    5.3*12, 1.0625*12, 0;
    6.4*12, 2.85*12, 0;];
dimCrater = [3.149625;
    4.14275;
    0;];
dimPenetration = 0.871;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(12) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% ------------------------------------------------------------------
---
```

```
% STEEL PENETRATOR (30LB)
strTestCase = 'Steel penetrator (30lb)';
mMedia = MediaLibrary(4);
dMass = 32;                         % (lb)
Diameter = 3.5;                     % (in)
OgiveRadius = 7;                    % (in) - estimated using 45 deg
chamfer
Length = 12.63;                     % (in)
Diameter_Meplat = 0.5;              % (in)
VolumeLoss = 0.404048*61.0237;      % (L) -> (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [6.4*12, 2.65*12, 0;
    6.5*12, 2.2*12, 0;
%     6.9*12, 1.75*12, 0;
    6.4*12, 2.8*12, 0];
dimCrater = [4.3;
    4.65;
%     1.06;
    0;];
dimPenetration = 0.739;

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(13) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% ------------------------------------------------------------------------
---
% STEEL RAIL (35LB)
strTestCase = 'Steel rail (35lb)';
mMedia = MediaLibrary(4);
dMass = 34.5;                       % (lb)
Diameter = 5.5;                     % (in)
OgiveRadius = 7;                    % (in) - estimated using 45 deg
chamfer
Length = 49.5;                      % (in)
Diameter_Meplat = 0.5;              % (in)
VolumeLoss = 0.0524166*61.0237;     % (L) -> (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [2.8*12, 0.125*12, 0;
    3.125*12, 3.95*12, 0;
    3.125*12, 3.95*12, 0;];
dimCrater = [4.294;
    6.261;
    0;];
dimPenetration = 0.925;     % Avg 0.393 in
```

```
ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(14) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------
---
% STEEL PENETRATOR (20LB)
strTestCase = 'Steel penetrator (20lb)';
mMedia = MediaLibrary(4);
dMass = 22;                             % (lb)
Diameter = 3.5;                         % (in)
OgiveRadius = 7;                        % (in) - estimated using 45 deg
chamfer
Length = 8.92;                          % (in)
Diameter_Meplat = 0.5;                  % (in)
VolumeLoss = 1.34564*61.0237;           % (L) -> (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [5.4*12, 1.3*12, 0;
    5.3*12, 1.0625*12, 0;
    6.4*12, 2.85*12, 0;];
dimCrater = [11.145;
    8;
    0;];
dimPenetration = 3.062992;          % Avg 0.909 in

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(15) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------
---
% STEEL PENETRATOR (30LB)
strTestCase = 'Steel penetrator (30lb)';
mMedia = MediaLibrary(4);
dMass = 32;                             % (lb)
Diameter = 3.5;                         % (in)
OgiveRadius = 7;                        % (in) - estimated using 45 deg
chamfer
Length = 12.63;                          % (in)
Diameter_Meplat = 0.5;                  % (in)
VolumeLoss = 71.3023*61.0237;           % (L) -> (in^3)
nProjectiles = 1;
% Location of impact is measured from bottom-right when viewing front
face
```

```
% [x,y,z] - all units in inches, use magnitude for all units
% For multiple impacts, create another row in matrix
dimImpact = [1.75*12, 3.6*12, 0;
    1.75*12, 3.6*12, 0;
    0, 0, 0;];
dimCrater = [11.462;
    2*12;
    0;];
dimPenetration = 4.7795276;     % Avg 5.428 in

ImpactLoc_ = struct('Case',strTestCase,'Media',mMedia,...
    'nProjectiles',nProjectiles,'Mass',dMass,...
    'Diameter',Diameter,'Length',Length,'OgiveRadius',OgiveRadius,...
    'Locations',dimImpact,'VolumeLoss',VolumeLoss,...
    'CraterDiameters',dimCrater,'PenetrationMeasured',dimPenetration);
ImpactLoc(16) = ImpactLoc_;
clearvars strTestCase mMedia dMass dimImpact ImpactLoc_;
% -------------------------------------------------------------------------
---
%
*************************************************************************
*
% -------------------------------------------------------------------------
---
% Return test case mechanical data
TestCaseData = ImpactLoc;
end
```

---

fVelocityInterpolate.m

```
function [velocity_impact] = fVelocityInterpolate(...
    ProjectileMass,CalibrationCurve)
%
=========================================================================
=
% Interpolate projectile velocity from calibration curve to provide
% comparison basis
% INPUTS:
% Projectile - Struct with projectile mass (lb)
% CalibrationCurve - 2xn array with [Mass (lb), Velocity (ft/s)]
calibrated values
%
=========================================================================
=

for i = 2:size(CalibrationCurve,1)
    if(ProjectileMass - CalibrationCurve(i,1) < 1e-6...
            && ProjectileMass - ...
            CalibrationCurve(i-1,1) > 1e-6)
        % Interpolate impact velocity based on calibration data
        velocity_impact = CalibrationCurve(i-1,2) - ...
```

```
            (CalibrationCurve(i-1,1) - ...
            ProjectileMass)/...
            (CalibrationCurve(i-1,1)-...
            CalibrationCurve(i,1)) * ...
            (CalibrationCurve(i-1,2) - ...
            CalibrationCurve(i,2));
        break
    elseif(ProjectileMass - CalibrationCurve(1,1) < 1e-6)
        % Extrapolate from calibration curve
        velocity_impact = CalibrationCurve(2,2) - ...
            (CalibrationCurve(2,1) - ...
            ProjectileMass)/...
            (CalibrationCurve(2,1)-...
            CalibrationCurve(1,1)) * ...
            (CalibrationCurve(2,2) - ...
            CalibrationCurve(1,2));
        break
    elseif(ProjectileMass - CalibrationCurve(end,1) > 1e-6)
        % Extrapolate from calibration curve
        velocity_impact = CalibrationCurve(end,2) + ...
            (ProjectileMass - ...
            CalibrationCurve(end,1))/...
            (CalibrationCurve(end,1)-...
            CalibrationCurve(end-1,1)) * ...
            (CalibrationCurve(end,2) - ...
            CalibrationCurve(end-1,2));
        break
    else
        % Default to maximum calibration velocity
        velocity_impact = CalibrationCurve(end,1);
    end
end

end
```

---

mFileRawDataDirectory.m

```
%
=========================================================================
=
% Acquire raw data directory --> navigate from working directory to
test
% dates and test cases
%
=========================================================================
=

% ------------------------------------------------------------------
---
% USER I/O
```

```
% Select test date
cd(FileParentFolder);
cd(FileDataDirectory);
dirTestDate = dir();
if(bAutoMode)
    % Automatically process all test dates
    % Initialize directories for first iteration
    if(abs(ID_AutoTestDate - 3) < 1e-6 && abs(ID_AutoTestCase - 2) <
1e-6)
        FileTestDateFolder = dirTestDate(ID_AutoTestCase).name;
        dirTestCase = dir(FileTestDateFolder);
    elseif(bReadInputImpactEnergy)
        % Data may have been read from file
        FileTestDateFolder = dirTestDate(ID_AutoTestDate).name;
        dirTestCase = dir(FileTestDateFolder);
        fprintf('\nNOTE>> Resuming analysis at:\n');
        fprintf('\t%s\n',FileTestDateFolder);
        fprintf('\t%s\n',dirTestCase(ID_AutoTestCase+1).name);
        bReadInputImpactEnergy = false;
    end
    if(abs(ID_AutoTestDate - size(dirTestDate,1)) < 1e-6 && ...
            abs(ID_AutoTestCase - size(dirTestCase,1)) < 1e-6)
        % All test cases for all test dates have been analyzed
        fprintf('NOTE>> All test dates and cases have been
analyzed\n');
        bAnalyzeData = false;
        bAutoModeTerminate = true;
    elseif(ID_AutoTestDate - size(dirTestDate,1) < 1e-6 && ...
            abs(ID_AutoTestCase - size(dirTestCase,1)) > 1e-6)
        % Test cases still require analysis
        ID_AutoTestDate = ID_AutoTestDate;
    elseif(abs(ID_AutoTestCase - size(dirTestCase,1)) < 1e-6)
        % All test cases for selected testdate have been analyzed
        % Increment test date and reset test case index
        ID_AutoTestDate = ID_AutoTestDate + 1;
        ID_AutoTestCase = 2;
    else
        % Increment test date
        ID_AutoTestDate = ID_AutoTestDate;
    end
    ID_testdate = ID_AutoTestDate;
else
    % Initiate I/O mode with user
    fprintf('\nTest dates:\n');
    for i = 3:size(dirTestDate,1)
        fprintf('[%d]%s\n',i-2,dirTestDate(i).name)
    end
    fprintf('Select test to analyze or 0 to exit\n');
    ioTestDate = input('Selection:');
    if(ioTestDate - (size(dirTestDate,1)-2) > 1e-6)
        fprintf('WARNING>> Improper selection.  Please select test to
analyze.\n');
    elseif(abs(ioTestDate - 0) < 1e-6)
%         error('TERMINATING SCRIPT');
        bAnalyzeData = false;
        bAutoModeTerminate = true;
```

```
        return
    else
        bImproperSelection = false;
    end
    ID_testdate = ioTestDate+2;
end
%Check if selected folder is a file
if(isfile(dirTestDate(ID_testdate).name))
    fprintf('WARNING>> Selected test date folder is a file:\t%s',...
        dirTestDate(ID_testdate).name);
    fprintf('NOTE>> All test dates and cases have been analyzed\n');
    bAnalyzeData = false;
    bAutoModeTerminate = true;
    iCounter = iCounter - 1;
    return
else
    FileTestDateFolder = dirTestDate(ID_testdate).name;
end
% Select test case
dirTestCase = dir(FileTestDateFolder);
if(bAutoMode)
    % Auto analyze all test cases for test date
%     if(ID_AutoTestCase - size(dirTestCase,1) < 1e-6)
%         ID_AutoTestCase = ID_AutoTestCase + 1;
    if(abs(ID_AutoTestCase - size(dirTestCase,1))-1 < 1e-6 && ...
            abs(ID_AutoTestDate - size(dirTestDate,1)) < 1e-6)
        % This is the last test case to be analyzed for the last test
date
        ID_AutoTestCase = ID_AutoTestCase + 1;
        bAutoModeTerminate = true;
    elseif(ID_AutoTestCase - size(dirTestCase,1) < 1e-6)
        ID_AutoTestCase = ID_AutoTestCase + 1;
%     elseif(abs(ID_AutoTestCase - size(dirTestCase,1))-1 < 1e-6 && ...
%             abs(ID_AutoTestDate - size(dirTestDate,1)) < 1e-6)
%         % This is the last test case to be analyzed for the last test
date
%         ID_AutoTestCase = ID_AutoTestCase + 1;
%         bAutoModeTerminate = true;
    else
        ID_AutoTestCase = ID_AutoTestCase;
    end
    ID_testcase = ID_AutoTestCase;
else
    % Intiate I/O mode with user
    fprintf('\nTest cases:\n');
    for i = 3:size(dirTestCase,1)
        fprintf('[%d]%s\n',i-2,dirTestCase(i).name)
    end
    fprintf('Select test case to analyze or 0 to exit\n');
    ioTestCase = input('Selection:');
    if(ioTestCase - (size(dirTestCase,1)-2) > 1e-6)
        fprintf('WARNING>> Improper selection.  Please select test to
analyze.\n');
    elseif(abs(ioTestCase - 0) < 1e-6)
        error('TERMINATING SCRIPT');
        return
```

```
    else
        bImproperSelection = false;
    end
    ID_testcase = ioTestCase + 2;
end
FileTestCaseFolder = dirTestCase(ID_testcase).name;
FilePath = strcat(FileTestDateFolder,'\',FileTestCaseFolder,'\');
% ------------------------------------------------------------------------
---
% Display which test case the program is analyzing
fprintf('\n-----------------------------------------------------');
fprintf('\nTest date:\t%s',FileTestDateFolder);
fprintf('\nTest dase:\t%s',FileTestCaseFolder);
fprintf('\n-----------------------------------------------------');
% ------------------------------------------------------------------------
---
% Acquire raw data directory
% Auto read all channel outputs
dirFile = dir(FilePath);
ID_dirlimit = 1;                    % index for FileIDs
for i=1:size(dirFile,1)
    FileName = dirFile(i).name;
    tempFileName = strsplit(FileName,'.');
    if(strncmpi(tempFileName(1),FileDataPrefix,2))    % compare first 2
characters
        vFileIDs(ID_dirlimit) = i;
        ID_dirlimit = ID_dirlimit+1;
    end
end
cd(FilePath);
```

---

mFileReadInitializeArray.m

```
%
=========================================================================
=
% Acquire raw data directory and populate data structs and arrays with
% input data from DataTrap channels
%
=========================================================================
=

% ------------------------------------------------------------------------
---
% Initialize struct - pull data from first channel
FileName = dirFile(vFileIDs(1)).name;
% Verify file selected is .txt file
tempFileName = strsplit(FileName,'.');
if(strcmp(tempFileName(2),'txt'))
```

```
%     File selected is .txt
else
%     File selected is not .txt
    fprintf('File selected is not .txt\n');
    return;
end
% Test File Reader
% Text file reader - current method reads row by row
ID_FileIN = fopen(FileName,'r');
FileData = sscanf(fread(ID_FileIN,[1,inf],'*char'),'%f');
fclose(ID_FileIN);
[Time,Voltage] = fOrganizeTimeVoltage(FileData,bOrganizeArray);
% tempCHData = struct('Channel','Data');
tempCHData.Channel = tempFileName{1};
tempCHData.Data = [Time,Voltage];
tempCHData.Serial = ID_SensorSerial(1);
% Pre-allocate channel struct
for j = 1:ID_dirlimit-1
    CHData(j) = tempCHData;
end
for j = 2:ID_dirlimit-1
    FileName = dirFile(vFileIDs(j)).name;
    % Verify file selected is .txt file
    tempFileName = strsplit(FileName,'.');
    if(strcmp(tempFileName(2),'txt'))
    %     File selected is .txt
    else
    %     File selected is not .txt
        fprintf('File selected is not .txt\n');
        return;
    end
    % Test File Reader
    % Text file reader - current method reads row by row
    ID_FileIN = fopen(FileName,'r');
    FileData = sscanf(fread(ID_FileIN,[1,inf],'*char'),'%f');
    fclose(ID_FileIN);
    [Time,Voltage] = fOrganizeTimeVoltage(FileData,bOrganizeArray);
    tempCHData.Channel = tempFileName{1};
    tempCHData.Data = [Time,Voltage];
    tempCHData.Serial = ID_SensorSerial(j);
    CHData(j) = tempCHData;
end
clearvars Time Voltage j_dirlimit tempCHData FileData;
% Determine time step
dt = CHData(1).Data(2,1) - CHData(1).Data(1,1);     % (ms)
dt = dt/1000;                                       % (s)
```

---

mFileWorkingDirectory.m

```
%
=========================================================================
=
% Acquire parent working directory
%
=========================================================================
=
% % Acquire parent matlab script directory
FileCurrentFolder = pwd;
tempFilePath = strsplit(FileCurrentFolder,'\');
for i = 1:size(tempFilePath,2)
    if(strcmp(tempFilePath{i},FileMatlabScripts))
        break
    end
end
tempFilePath2 = tempFilePath{1};
for j = 2:i
    tempFilePath2 = strcat(tempFilePath2,'\',tempFilePath{j});
end
FileParentFolder = tempFilePath2;
cd(FileParentFolder);
addpath(FileParentFolder);
clearvars tempFilePath tempFilePath2 i j;
```

---

mFilterChannelData.m

```
%
=========================================================================
=
% % Filter input channel data from low pass filter
%
=========================================================================
=
if(bFilter)
    % Construct signal filter
    LowPassFilter = dsp.LowpassFilter('SampleRate',lpf.samplerate,...
        'FilterType',lpf.filtertype,...
        'PassbandFrequency',lpf.Fpass,...
        'StopbandFrequency',lpf.Fstop,...
        'PassbandRipple',lpf.Rp,...
        'StopbandAttenuation',lpf.Astop);
    CHDataFiltered = CHData;
    for j = 1:size(CHData,2)
        CHDataFiltered(j).Data(:,2) =
step(LowPassFilter,CHData(j).Data(:,2));
    end
    CHData = CHDataFiltered;
    clearvars CHDataFiltered j;
end
```

---

mImpactCalcs.m

```
%=========================================================================
===
% Subroutine for impact anaylsis
%=========================================================================
===
% Determine the number of projectiles
% nProjectiles = size(Projectile.CraterDiameters,1) - nWads;
nProjectiles = Projectile.nProjectiles;
ImpactEnergy(iCounter).nProjectiles = nProjectiles;
% !!!!! Adjust the projectile mass to number of projectiles !!!!!
Projectile.Mass = Projectile.Mass / nProjectiles;
% Perform shock analysis
% -------------------------------------------------------------------
---
[ShockWaves] = fShockCalcs(velocity_impact,Projectile,Wad,nWads,...
    Medium,dimStrainSensorLoc);
% -------------------------------------------------------------------
---
% Estimate arrival times of shock waves/elastic waves
% -------------------------------------------------------------------
---
% Store estimated impact energy and maximum derived stress and strain-
rate
ImpactEnergy(iCounter).Case = Projectile.Case;
ImpactEnergy(iCounter).ImpactEnergy = 0.5 * (Projectile.Mass)*...
    (velocity_impact)^2;
% Convert impact energy to (J)
ImpactEnergy(iCounter).ImpactEnergy = ...
    ImpactEnergy(iCounter).ImpactEnergy * 0.0421401101;
% Store measured and actual (used) velocity -> convert to (m/s)
ImpactEnergy(iCounter).V_measured = velocity_measured * 0.3048;
ImpactEnergy(iCounter).V_actual = velocity_impact * 0.3048;
ImpactEnergy(iCounter).ProjectileMass = Projectile.Mass*0.453592;
ImpactEnergy(iCounter).ProjectileMassCumulative = Projectile.Mass * ...
    nProjectiles * 0.453592;
tempMax = -1e10;
for j = 1:size(PrincipalStress,2)
    if(~isempty(PrincipalStress(j).sigma_1))
        tempMax1 = max(PrincipalStress(j).sigma_1);
    else
        tempMax1 = -Inf;
    end
    if(~isempty(PrincipalStress(j).sigma_2))
        tempMax2 = max(PrincipalStress(j).sigma_2);
    else
        tempMax2 = -Inf;
    end
    if(~isempty(PrincipalStress(j).StressChannel))
        tempMax3 = max(max(PrincipalStress(j).StressChannel));
    else
        tempMax3 = -Inf;
```

```
    end
    tempMax = max(tempMax,tempMax1);
    tempMax = max(tempMax,tempMax2);
    tempMax = max(tempMax,tempMax3);
end
ImpactEnergy(iCounter).StressMax = tempMax;
for k = 1:size(PrincipalStress,2)
    if(isempty(PrincipalStress(k).sigma_1))
        k = k - 1;
        break;
    end
end
% Store maximum strain
ImpactEnergy(iCounter).StrainMax = max(StrainMax(:,2));
% Determine maximum strain-rate
ImpactEnergy(iCounter).StrainRateMax = max(StrainRateMax(:,2));
% Add dimensionless impact energy parameter
KE_p = ImpactEnergy(iCounter).ImpactEnergy * 0.0421401101;
mass_p = Projectile.Mass*0.453592;
dia_p = Projectile.Diameter*0.0254;
v_p = velocity_impact * 0.3048;
R_mod = 1/2 *
(Medium.HugoniotElasticLimit^2/Medium.ElasticityModulusDynamic);
% ImpactEnergy(iCounter).E_dimensionless = ...
%     ImpactEnergy(iCounter).ImpactEnergy * 0.0421401101...
%     / (Projectile.Mass * 0.453592 * Projectile.Media.C_0^2);
% ImpactEnergy(iCounter).E_dimensionless = KE_p/(((mass_p/dia_p^3)/...
%     Medium.Density)* Medium.C_0^2);
% Use a modified Weber number (ratio of kinetic energy on impact to
surface
% energy
ImpactEnergy(iCounter).E_dimensionless = 1/2 *
Projectile.Media.Density*...
    v_p^2*dia_p^2/R_mod;
% Convert volume loss to (m^3)
ImpactEnergy(iCounter).VolumeLoss = Projectile.VolumeLoss*1.63871e-5;
clearvars KE_p mass_p dia_p v_p R_mod j k ;
clearvars tempMax tempMax1 tempMax2 tempMax3;
% ---------------------------------------------------------------------
---
% Estimate actual penetration depth using VolumeLoss data
% Volume of a cone -> concrete fails in a conical manner
% V = pi() * r^2 * h/3
tempVol = Projectile.VolumeLoss;        % (in^3)
% Default - distribute volume lost to all identified craters for the
case
tempCraters = Projectile.CraterDiameters(:,1) > 1e-6;
tempCraters = Projectile.CraterDiameters(tempCraters,1);
% Distribute volume based on relative diameter/volume size to the data
set
tempD = Projectile.CraterDiameters(1:end,1)/2;
tempVD = 0;
VolRatio = zeros(size(Projectile.CraterDiameters,1),1);
for j = 1:size(tempD,1)
    VolRatio(j,1) = pi()*tempD(j,1)^2*...
        (2*tempD(j,1)/Projectile.Mass*2.20462)/3;
```

```
    tempVD = tempVD + VolRatio(j,1);
end
clearvars tempD;
% tempVol = tempVol/size(tempCraters,1);
if(isempty(tempVol))
    % No volume lost for this test case
    ImpactEnergy(iCounter).PenetrationModel = 0;
else
    zPenetrationVolume = -Inf;
    for i = 1:size(Projectile.CraterDiameters,1)
        if(Projectile.CraterDiameters(i,1) < 1e-6)
            tempZ = 0;
        else
            tempR = Projectile.CraterDiameters(i,1)/2;
            tempZ = 3 * (tempVol*(VolRatio(i,1)/tempVD) / ...
                (pi() * tempR^2));
%             tempZ = 3 * (tempVol / (pi() * tempR^2));
            if(tempZ > 1e6)
                tempZ = 0;
            end
        end
        zPenetrationVolume = max(zPenetrationVolume,tempZ);
    end
    ImpactEnergy(iCounter).PenetrationVolume = zPenetrationVolume;
    % Convert derived measured penetration to (m)
    ImpactEnergy(iCounter).PenetrationVolume =
zPenetrationVolume*0.0254;
    clearvars tempVol tempVD VolRatio tempR tempZ zPenetrationModel i;
end
% Save measured penetration depths from test case data (in) -> (m)
ImpactEnergy(iCounter).PenetrationActual = ...
    Projectile.PenetrationMeasured*0.0254;
% Write penetration outputs
fprintf('\n----------------------------------------------------------------
');
fprintf('\nNOTE>> Derived measured penetration depth (in):\t%.1f',...
    ImpactEnergy(iCounter).PenetrationVolume/0.0254);
fprintf('\n----------------------------------------------------------------
');
fprintf('\n');
% ----------------------------------------------------------------------
---
% Estimate penetration depth using terminal ballistics models
ImpactEnergy(iCounter).PenetrationEst = fPenetrationConcrete(...
    Projectile,velocity_impact,Medium,t_Seal);
fprintf('\n----------------------------------------------------------------
');
fprintf('\nNOTE>> Estimated penetration depth (in):\t%.1f',...
    ImpactEnergy(iCounter).PenetrationEst/0.0254);
fprintf('\n----------------------------------------------------------------
');
fprintf('\n');
% Actual measured value
fprintf('\n----------------------------------------------------------------
');
fprintf('\nNOTE>> Measured penetration depth (in):\t%.1f',...
```

```
    ImpactEnergy(iCounter).PenetrationActual/0.0254);
fprintf('\n-----------------------------------------------------------------
');
fprintf('\n');
% ------------------------------------------------------------------------
---
% Store safety factors for writing to file
% Determine penetration safety factor -> HEL/P_impact
ImpactEnergy(iCounter).SF_Penetration = Medium.HugoniotElasticLimit/...
    max(ShockWaves.Pressure(:)) - 1;
% Determine spall safety factor --> spall strength/relief wave
pressures
ImpactEnergy(iCounter).SF_Spalling = ShockWaves.SF_spall;
% Compare shock impedance for projectile/target
ImpactEnergy(iCounter).SF_Zt =
max(ShockWaves.ShockImpedance_Projectile./...
    ShockWaves.ShockImpedance_Target + 1);
```

---

mImpactStructInitialize.m

```
%
=========================================================================
=
% % Purposes:
% 1 - Initialize ImpactEnergy struct for storage of relevant variables
for
%   comparison between test cases
% 2 - Input Impact data from previous test runs and initialize
ImpactEnergy
%   variable
% INPUTS:
%     'ImpactData_OUTPUT.txt' - csv text file in output directory from
%     previous script runs
%
=========================================================================
=

% n - Rows --> Parameters i.e. Test case, number of projectiles,
velocity etc.
% m - Columns --> Values for each test case + header column with titles

% ------------------------------------------------------------------------
---
% Initialize ImpactEnergy struct
ImpactEnergyFields = [
    "Case";
    "nProjectiles";
    "ImpactEnergy";
    "E_dimensionless";
    "V_measured";
    "V_actual";
```

```
    "ProjectileMass";
    "ProjectileMassCumulative";
    "PenetrationEst";
    "PenetrationVolume";
    "PenetrationActual";
    "VolumeLoss";
    "StressMax";
    "StrainMax";
    "StrainRateMax";
    "SF_Penetration";
    "SF_Cracking";
    "SF_Spalling";
    "SF_Zt";];
% tempFieldSubstrings = [
%     "case";"Projectile";"Energy";"dimensionless";"measured";"actual";
%
"Penetration";"Est";"Volume";"Actual";"Loss";"Stress";"Max";"Strain";
%     "StrainRate";"Safety Factor";"Z_t";];
% % Statically define struct fields
% % ImpactEnergy =
struct('Case',0,'nProjectiles',0,'ImpactEnergy',0,...
% %
'StressMax',0,'StrainMax',0,'StrainRateMax',0,'E_dimensionless',0,...
% %     'V_measured',0,'V_actual',0,'PenetrationEst',0,...
% %     'PenetrationVolume',0,'PenetrationActual',0,'VolumeLoss',0,...
% %     'SF_Penetration',0,'SF_Cracking',0,'SF_Spalling',0,'SF_Zt',0);
% Generate field names from variable and initialize struct with 0's
for i = 1:size(ImpactEnergyFields,1)
    % Initialize struct
    ImpactEnergy.(ImpactEnergyFields(i)) = 0;
end

% ------------------------------------------------------------------------
---
% Check for output text file
% Navigate to output directory
FilePathOUT_final = strcat(FileParentFolder,'\',...
        FileDataDirectoryOUT,'\');
cd(FilePathOUT_final);

% Output file name
tempImpactDataFileName = strcat('ImpactData_OUTPUT',FileExtension);

if(bReadInputImpactEnergy)
    % Attempt to read Impact Energy data from file
    fprintf('\nNOTE>> Attempting to read Impact data from file.');
if(isfile(tempImpactDataFileName))
    % Output file exists
    fprintf('\nNOTE >> File found:\t%s',tempImpactDataFileName);
    fprintf('\n');
    % Read input file and convert to array
    tempIDImpactFile = fopen(tempImpactDataFileName,'r');
%     tempInputImpactData = fscanf(tempIDImpactFile,'%s');
    tempInputImpactData = tdfread(tempImpactDataFileName,'\t'); % tab-
delimited
```

```
%     tempInputImpactData =
sscanf(fread(tempIDImpactFile,[1,inf],'*char'),'%f');
    fclose(tempIDImpactFile);

    % Convert input data to object
    % !!!!! Check compatibility with current ImpactEnergy struct fields
    tempfieldnames = fieldnames(tempInputImpactData);
    tempfieldnamesString = tempfieldnames;
    % Field name substring replacement
    for i = 2:size(tempfieldnames,1)
        tempfieldnamesString{i} =
convertCharsToStrings(tempfieldnamesString{i});
        tempfieldnamesString{i} = strrep(tempfieldnamesString{i},"_","
");
        tempfieldnamesString{i} =
strrep(tempfieldnamesString{i},"0x28","(");
        tempfieldnamesString{i} =
strrep(tempfieldnamesString{i},"0x29",")");
        % Special cases --> caution using numbers as first letter in
string
        tempfieldnamesString{i} =
strrep(tempfieldnamesString{i},"x5x5","5x5");
    end
    % !!!!! FIX THIS - find substrings to index parameters !!!!!
%     iFieldIndex =
zeros(size(tempInputImpactData.tempfieldnames{1},1));
%     for i = 1:size(tempInputImpactData.tempfieldnames{1},1)
%         tempInputHeader(i) = convertCharsToStrings(...
%             tempInputImpactData.(tempfieldnames{1})(i,:));
%         for j = 1:size(ImpactEnergyFields,1)
%             tempID =
strfind(ImpactEnergyFields(j),tempInputHeader(i));
%             if(~isempty(tempID))
%                 iFieldIndex(j) = i;
%                 break;
%             end
%         end
%     end
    % Assign values to struct
    % Define test cases
    for i = 2:size(tempfieldnames,1)
        ImpactEnergy(i-1).(ImpactEnergyFields(1)) = string(...
            tempfieldnamesString(i));
    end
    % Ensure the array sizes match between input and defined struct
    if(abs(size(tempInputImpactData.(tempfieldnames{1}),1) - ...
            (size(ImpactEnergyFields,1)-1)) > 1e-6)
        % Array dimension mismatch -> struct will not be constructed
proper
        fprintf('\nWARNING >> ARRAY DIMENSION MISMATCH BETWEEN INPUT
AND DEFINED STRUCT PARAMETERS FOR IMPACT ENERGY');
        fprintf('\n');
    end
    % Assign values to fields
    for i = 1:size(ImpactEnergy,2)
        for j = 1:size(ImpactEnergyFields,1)-1
```

```
            ImpactEnergy(i).(ImpactEnergyFields(j+1)) = ...
                tempInputImpactData.(tempfieldnames{i+1})(j);
            % Shift values for special cases

if(contains(tempInputImpactData.(tempfieldnames{1})(j,:),'10^-3'))
                ImpactEnergy(i).(ImpactEnergyFields(j+1)) = ...
                    ImpactEnergy(i).(ImpactEnergyFields(j+1)) * 1e-3;

elseif(contains(tempInputImpactData.(tempfieldnames{1})(j,:),'1e-6'))
                ImpactEnergy(i).(ImpactEnergyFields(j+1)) = ...
                    ImpactEnergy(i).(ImpactEnergyFields(j+1)) * 1e-6;
            end
        end
    end
    bReadInputImpactEnergy = true;
else
    % Output file does not exist in directory
    fprintf('\nNOTE >> File not found:\t%s',tempImpactDataFileName);
    fprintf('\n');
    bReadInputImpactEnergy = false;
end
clearvars tempImpactDataFileName tempfieldnames tempfieldnamesString;
clearvars tempIDImpactFile tempInputImpactData;
else
    % Do not attempt to read Impact data from file
end
```

---

mMaterialFailure.m

```
%=========================================================================
===
%Determine the failure condition of a material for a given state of
stress
%given a medium and principal stress arrays (sigma_1, sigma_2) and
%corresponding strain-rates
% Input:
% Medium -> struct with material properties and failure criteria
% PStress -> struct with principal stress and strain-rate bin data
%=========================================================================
===
%
% Common Failure Criteria:
% 1 - Maximum Distortion Energy (von Mises)
% 2 - Maximum Shear Stress (Tresca)
% 3 - Maximum Normal Stress (Coulomb) (termed Mohr-Coulomb where
CompressiveUlt > TensileUlt)
%
% Failure Modes:
% 1 - Deflection (i.e. Deflection e = P*L/(A*E) or Twist theta =
T*L/(G*J))
% 1a - Elastic deflection
```

```
% 1b - Creep
% 2 - General Yielding
% 3 - Fracture
% 3a - Brittle fracture
% 3b - Crack propagation (cracked or flawed members)
% 3c - Fatigue (progressive fracture)
% 4 - Instability (buckling, i.e. P_cr = pi()^2*E*I/(L^2)
%

for fi = 1:size(Medium.FailureCriteria,1)
    % Store failure criteria
    tempFailureCriteria = Medium.FailureCriteria(fi);
switch(tempFailureCriteria)
    case "vonMises"
%-----------------------------------------------------------------------
---
% Maximum Distortion Energy (von Mises)
% Good for metal materials - failure is caused by material distortion
% Tensile stress is positive
% (sigma_1 - sigma_2)^2 + (sigma_2 - sigma_3)^2 + (sigma_3 - sigma_1)^2
= 2*TensileYield^2
% sigma_1 > sigma_2 > sigma_3
% K^2 = (1/3)*TensileYield^2;       % K is yield stress in pure shear
% Envelope curve
% TensileYield^2 = sigma_1^2 - sigma_1*sigma_2 + sigma_2^2;
% Tau = TensileYield/sqrt(3);
for i = 1:size(PrincipalStress,2)
    ID_figure = ID_figure + 1;
    figure(ID_figure);
    x = -
1.15470053*Medium.TensileYield:200:1.15470053*Medium.TensileYield;
    y1 = (1/2)*(x - sqrt(4*Medium.TensileYield^2-3*x.^2));
    y2 = (1/2)*(x + sqrt(4*Medium.TensileYield^2-3*x.^2));
    y = [y1;y2];
    plot(x,y,'k-','LineWidth',2);
    grid on
    xlabel('\sigma_1')
    ylabel('\sigma_2')
    title('Maximum Distortion Energy Failure Criteria')
    legend('Failure envelope','Location','southeast')
end
%-----------------------------------------------------------------------
---
    case "Tresca"
% Maximum Shear Stress (Tresca)
% Good for ductile materials - failure mechanism is by slippage along
shear planes
% Tensile stress is positive
% (sigma_1 - sigma_2) < TensileYield; (sigma_1 - sigma_3) <
TensileYield;
% (sigma_2 - sigma_3) < TensileYield;
% (sigma_1 - sigma_2)^2 < 4*ShearUlt^2; (sigma_2 - sigma_3)^2 <
4*ShearUlt^2;
% (sigma_1 - sigma_3)^2 < 4*ShearUlt^2; can use ShearUlt/2 =
TensileYield
% sigma_1 = +/- ShearUlt
```

```
% sigma_2 = +/- ShearUlt
for i = 1:size(PrincipalStress,2)
    ID_figure = ID_figure + 1;
figure(ID_figure);
xneg = -Medium.ShearUltimate;
xpos = Medium.ShearUltimate;
yneg = -Medium.ShearUltimate;
ypos = Medium.ShearUltimate;
x = [xneg, xneg, 0, xpos, xpos, 0, xneg];
y = [yneg, 0, ypos, ypos, 0, yneg, yneg];
plot(x,y,'k-','LineWidth',2);
grid on
xlabel('\sigma_1')
ylabel('\sigma_2')
title('Maximum Shear Stress Failure Criteria')
legend('Failure envelope','Location','southeast')
end
%-------------------------------------------------------------------------
---
    case "Coulomb"
% Maximum Normal Stress (Coulomb)
% Normal stress will cause failure
% Tensile stress positive
% sigma_1, sigma_2, sigma_3 < TensileUlt (or CompressiveUlt Mohr-
Coulomb)
% Plot failure envelope
% for i = 1:size(PrincipalStress,2)
%     ID_figure = ID_figure + 1;
% figure(ID_figure);
% xneg = -Medium.TensileYield;
% xpos = Medium.TensileYield;
% yneg = -Medium.TensileYield;
% ypos = Medium.TensileYield;
% x = [xneg, xneg, xpos, xpos, xneg];
% y = [yneg, ypos, ypos, yneg, yneg];
% plot(x,y,'k-','LineWidth',2);
% grid on
% xlabel('\sigma_1')
% ylabel('\sigma_2')
% title('Maximum Normal Stress Failure Criteria')
% legend('Failure envelope','Location','southeast')
% end
for i = 1:size(PrincipalStress,2)
    ID_figure = ID_figure + 1;
    % Adjust failure criteria for dynamic response
    if(PrincipalStress(i).LimitStrainRate <=
Medium.TensileYieldDynamic(i,1))
        F_ty = Medium.TensileYieldDynamic(i,2);
    else
        F_ty = Medium.TensileYield;
    end
F_cy = Medium.CompressiveYield;
temptitle = 'Maximum Normal Stress Failure Criteria';
tempsubtitle = strcat('Strain-rate:',...
    string(PrincipalStress(i).LimitStrainRate),' s^{-1}');
tempheader = strcat({temptitle;tempsubtitle});
```

```
if(~isempty(PrincipalStress(i).sigma_1))
imgFigure = figure(ID_figure);
xneg = -F_cy;
xpos = F_ty;
yneg = -F_cy;
ypos = F_ty;
x = [xneg, xneg, 0, xpos, xpos, 0, xneg];
y = [yneg, 0, ypos, ypos, 0, yneg, yneg];
plot(x,y,'k-','LineWidth',2);
grid on
xlabel('\sigma_1 (Pa)','FontName','Times New Roman')
ylabel('\sigma_2 (Pa)','FontName','Times New Roman')
title(tempheader,'FontName','Times New Roman')
tempLegend = cell(2,1);
tempLegend{1} = 'Failure envelope';
% legend('Failure envelope','Location','southeast')
xlim([1.1*xneg 1.5*xpos])
ylim([1.1*yneg 1.5*ypos])
hold on
% Plot stress instances
scatter(PrincipalStress(i).sigma_1,PrincipalStress(i).sigma_2);
tempLegend{2} = 'Location_A';
if(~isempty(PrincipalStress(i).StressChannel))
cS1 = -1;
k = 3;
% tempAlphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
for j = 1:(size(PrincipalStress(i).StressChannel,2)-4)/2
    cS1 = cS1 + 3;
    scatter(PrincipalStress(i).StressChannel(:,cS1),...
        PrincipalStress(i).StressChannel(:,cS1+1));
%     strLegend = tempAlphabet(k-1);
    strLegend = CHSerial2(j,2);
%     tempLegend{k} = strcat('Location_',strLegend);
    tempLegend{k} = strcat('S/N:',strLegend);
    k = k + 1;
end
end
legend(tempLegend,'FontName','Times New Roman','Location','southeast')
hold off

% % Test failure criteria
% sigmaFailure = [PStress(i).sigma_1,PStress(i).sigma_2] > [xpos,ypos]
|| ...
%     [PStress(i).sigma_1,PStress(i).sigma_2] < [xneg,yneg];
if(bSaveOutputPlots)
    FileFigureOUT = strcat('Max Normal Failure Criteria',...
        '(',string(i),')');
%     FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
end
if(bAutoMode)
    close(imgFigure)
end
end
end
```

```
clearvars cS1 tempLegend tempAlphabet strLegend i j k;

% Determine cracking safety factor -> max normal stress criteria
tempTensile = -Inf;
% tempCompressive = Inf;
tempSR = 0;
F_tdy = 0;
for i = 1:size(PrincipalStress,2)
    if(~isempty(PrincipalStress(i).sigma_1))
        tempTensile = max(tempTensile,max(PrincipalStress(i).sigma_1));
%         tempCompressive = min(tempCompressive,...
%             min(PrincipalStress(i).sigma_1));
        % Sigma_1 and sigma_2 are paired so finding SR for one is the
same
        tempSR = PrincipalStress(i).LimitStrainRate;
        % Determine F_tdy -> !!!!! HARD CODED MATCHING - FIX THIS !!!!!
        if(tempSR <= Medium.TensileYieldDynamic(i,1))
            F_tdy = Medium.TensileYieldDynamic(i,2);
        else
            F_tdy = Medium.TensileYield;
        end
    end
    if(~isempty(PrincipalStress(i).sigma_2))
        tempTensile = max(tempTensile,max(PrincipalStress(i).sigma_2));
%         tempCompressive = min(tempCompressive,...
%             min(PrincipalStress(i).sigma_2));
    end
    % Determine max sigma_1, sigma_2 tensile stress for all channels
    if(~isempty(PrincipalStress(i).StressChannel))
        tempTensile = max(tempTensile,...
            max(PrincipalStress(i).StressChannel(:,2)));
        tempTensile = max(tempTensile,...
            max(PrincipalStress(i).StressChannel(:,3)));
        tempTensile = max(tempTensile,...
            max(PrincipalStress(i).StressChannel(:,5)));
        tempTensile = max(tempTensile,...
            max(PrincipalStress(i).StressChannel(:,6)));
        tempTensile = max(tempTensile,...
            max(PrincipalStress(i).StressChannel(:,8)));
        tempTensile = max(tempTensile,...
            max(PrincipalStress(i).StressChannel(:,9)));
    end
end
% Save failure safety factor
SF_failure = F_tdy/tempTensile - 1;
clearvars tempTensile;

%-------------------------------------------------------------------------
---
    case "Mohr-Coulomb"
% Mohr-Coulomb
% Good for rocks and soil - failure mechanism is internal friction
% Compressive stress is positive
% tau = sigma * tan(phi) + c
```

```
% tau = shear strength; sigma = normal stress; c = (cohesion) intercept
of the failure
% envelope with the shear axis; and tan(phi) (phi = angle of internal
friction)
% is the slope of the failure envelope
% C = Cohesion
% phi = FrictionAngle
% (sigma_1 - sigma_3) = (sigma_1 + sigma_3)*sin(phi) + 2*c*cos(phi);

for i = 1:size(PrincipalStress,2)
    ID_figure = ID_figure + 1;
% Define failure limits from material properties
if(Medium.Cohesion < 1e-6 || isempty(Medium.Cohesion))
    % Define cohesion from tensile and compressive limits
    % Adjust failure criteria for dynamic response
    if(PrincipalStress(i).LimitStrainRate <=
Medium.TensileYieldDynamic(i,1))
        F_ty = Medium.TensileYieldDynamic(i,2);
    else
        F_ty = Medium.TensileYield;
    end
F_cy = Medium.CompressiveYield;
% Derive cohesion and internal friction angle
% Reference:  ("Advanced Mechanics of Materials", Boresi, p. 126)
c = 1/2 * sqrt(F_ty*F_cy);
phi = asin((F_cy - F_ty)/(F_cy + F_ty));
% Determine characteristic lengths (pi() plane)
r_c = (sqrt(6) * F_cy * (1 - sin(phi)))/(3 - sin(phi));
r_t = (sqrt(6) * F_ty * (1 + sin(phi)))/(3 + sin(phi));
else
    % Use specified limits - !!!! Does not account for strain-rate
effects
    c = Medium.Cohesion;         % [Pa]
    phi = Medium.FrictionAngle; % [deg]
    phi = phi * pi()/180;        % [rad]
    r_c = (2*sqrt(6)*c*cos(phi))/(3-sin(phi));
    r_t = (2*sqrt(6)*c*cos(phi))/(3+sin(phi));
end

temptitle = 'Mohr-Coulomb Failure Criteria';
tempsubtitle = strcat('Strain-rate:',...
    string(PrincipalStress(i).LimitStrainRate),' s^{-1}');
tempheader = strcat({temptitle;tempsubtitle});
if(~isempty(PrincipalStress(i).sigma_1))
imgFigure = figure(ID_figure);
% xneg = -F_cy;
% xpos = F_ty;
% yneg = -F_cy;
% ypos = F_ty;
xneg = -(2*c*cos(phi)/(1 - sin(phi)));
xpos = (2*c*cos(phi)/(1 + sin(phi)));
yneg = -(2*c*cos(phi)/(1 - sin(phi)));
ypos = (2*c*cos(phi)/(1 + sin(phi)));
tempth = acos(r_t/r_c);
```

```
% f = sigma_1 - sigma_3 + (sigma_1 + sigma_3) * sin(phi) -
2*c*cos(phi);
% x = [xneg, xneg, 0, xpos, xpos, 0, xneg];
% y = [yneg, 0, ypos, ypos, 0, yneg, yneg];
x = [xneg, xneg, 0, xpos, xpos, 0, xneg];
y = [yneg, 0, ypos, ypos, 0, yneg, yneg];
plot(x,y,'k-','LineWidth',2);
grid on
xlabel('\sigma_1 (Pa)','FontName','Times New Roman')
ylabel('\sigma_2 (Pa)','FontName','Times New Roman')
title(tempheader,'FontName','Times New Roman')
tempLegend = cell(2,1);
tempLegend{1} = 'Failure envelope';
% legend('Failure envelope','Location','southeast')
xlim([1.1*xneg 1.5*xpos])
ylim([1.1*yneg 1.5*ypos])
hold on
% Plot stress instances
scatter(PrincipalStress(i).sigma_1,PrincipalStress(i).sigma_2);
tempLegend{2} = 'Location_A';
% Initialize containre for failure criteria test
tempTensile = -Inf;
tempCompressive = Inf;
SF_tfailure = -Inf;
SF_cfailure = -Inf;
SF_failure = -Inf;
% Plot individual stress cases
if(~isempty(PrincipalStress(i).StressChannel))
cS1 = -1;
k = 3;
% tempAlphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
for j = 1:(size(PrincipalStress(i).StressChannel,2)-4)/2
    cS1 = cS1 + 3;
    scatter(PrincipalStress(i).StressChannel(:,cS1),...
        PrincipalStress(i).StressChannel(:,cS1+1));
%     strLegend = tempAlphabet(k-1);
    strLegend = CHSerial2(j,2);
%     tempLegend{k} = strcat('Location_',strLegend);
    tempLegend{k} = strcat('S/N:',strLegend);
    k = k + 1;
    % Update worst case stress state
    tempTensile = max(tempTensile,...
        max(PrincipalStress(i).StressChannel(:,cS1)));
    tempCompressive = min(tempCompressive,...
        min(PrincipalStress(i).StressChannel(:,cS1)));
    tempTensile = max(tempTensile,...
        max(PrincipalStress(i).StressChannel(:,cS1+1)));
    tempCompressive = min(tempCompressive,...
        min(PrincipalStress(i).StressChannel(:,cS1+1)));
end
end
legend(tempLegend,'FontName','Times New Roman','Location','southeast')
hold off

% % Test failure criteria
```

```
% sigmaFailure = [PStress(i).sigma_1,PStress(i).sigma_2] > [xpos,ypos]
|| ...
%     [PStress(i).sigma_1,PStress(i).sigma_2] < [xneg,yneg];
if(bSaveOutputPlots)
    FileFigureOUT = strcat('Mohr-Coulomb Failure Criteria',...
        '(',string(i),')');
%     FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
end
if(bAutoMode)
    close(imgFigure)
end
% Determine yield function
% Reference:  "Advanced Mechanics of Materials", Boresi p. 126
% f = sigma_1 - sigma_3 + (sigma_1 + sigma_3) * sin(phi) -
2*c*cos(phi);
% f -> 0, yield function
% F_t = 2*c * cos(phi)/(1 + sin(phi));
% F_c = 2*c * cos(phi)/(1 - sin(phi));
% Update failure criteria
SF_tfailure = max(SF_tfailure,...
    tempTensile/(2*c*cos(phi)/(1 + sin(phi)))-1);
SF_failure = max(SF_failure,SF_tfailure);
SF_cfailure = max(SF_cfailure,...
    tempCompressive/(-(2*c*cos(phi)/(1-sin(phi))))-1);
SF_failure = max(SF_failure,SF_cfailure);
end
end
clearvars tempTensile tempCompressive tempth F_ty F_cy c phi r_c r_t;
clearvars cS1 tempLegend tempAlphabet strLegend i j k;

% Ductile-Coulomb-Mohr Theory (DCMT)
% sigma_1 < TensileYield/n;
% sigma_2 > -CompressiveYield/n
% sigma_1/TensileYield - sigma_2/CompressiveYield < 1/n

%-------------------------------------------------------------------------
---
    otherwise
% Maximum Shear Stress (Tresca)
% Conservative failure criteria
for i = 1:size(PrincipalStress,2)
    ID_figure = ID_figure + 1;
figure(ID_figure);
xneg = -Medium.ShearUltimate;
xpos = Medium.ShearUltimate;
yneg = -Medium.ShearUltimate;
ypos = Medium.ShearUltimate;
x = [xneg, xneg, 0, xpos, xpos, 0, xneg];
y = [yneg, 0, ypos, ypos, 0, yneg, yneg];
plot(x,y,'k-','LineWidth',2);
grid on
xlabel('\sigma_1')
ylabel('\sigma_2')
```

```
title('Maximum Shear Stress Failure Criteria')
legend('Failure envelope','Location','southeast')
end
end
%-------------------------------------------------------------------------
---
end
clearvars tempFailureCriteria;
%-------------------------------------------------------------------------
---
% Store failure safety factor
ImpactEnergy(iCounter).SF_Cracking = SF_failure;
```

---

mPlotData.m

```
%
=========================================================================
=
% % Plot data from strain sensor channels and converted strain and
% principal strain values
%
=========================================================================
=
% Create figure output directory
if(bSaveOutputPlots)
%     FileFigDirOUT =
strcat(FileParentFolder,'\',FileFigureDirectory,'\',...
%         FileTestDateFolder,'\',FileTestCaseFolder,'\');
%     FileFigDirOUT_final = strcat(FileParentFolder,'\',...
%         FileFigureDirectory,'\');
    FileFigDirOUT =
strcat(FileParentFolder,'\',FileDataDirectoryOUT,...
        '\',FileTestDateFolder,'\',FileTestCaseFolder,'\');
    FileFigDirOUT_final = strcat(FileParentFolder,'\',...
        FileDataDirectoryOUT,'\');
    % Create figure directory if one does not already exist
    if(~exist(FileFigDirOUT,'dir'))
        mkdir(FileFigDirOUT);
    end
end

% -------------------------------------------------------------------------
---
% Plot input data
ID_figure = 0;              % generate handle to avoid figure overwrite
if(bPlotInputCHData)
    for j = 1:size(CHData,2)
        ID_figure = ID_figure + 1;
        if(strcmp(CHData(j).Channel,'CH1') || ...
                strcmp(CHData(j).Channel,'CH2'))
            ylim_minus = 0;
```

```
            ylim_plus = 10.0;
            temptitle = strcat(CHData(j).Channel,...
            ' Input');
            tempsubtitle = strcat('Sensor Serial#','N/A');
        else
            ylim_minus = -2.5;
            ylim_plus = 2.5;
            temptitle = strcat(CHData(j).Channel,...
            ' Input');
            tempsubtitle = strcat('Sensor
Serial#',string(CHData(j).Serial));
        end
        tempheader = strcat({temptitle;tempsubtitle});
        imgFigure = figure(ID_figure);
        plot(CHData(j).Data(:,1),CHData(j).Data(:,2))
        axis([CHData(j).Data(1,1) CHData(j).Data(end,1) ...
            ylim_minus ylim_plus])
        xlabel('Time (ms)','FontName','Times New Roman')
        ylabel('Voltage (V)','FontName','Times New Roman')
        title(tempheader,'FontName','Times New Roman')
        grid on
        % Save figure
        if(bSaveOutputPlots)
            FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
        end
        if(bAutoMode)
            close(imgFigure);
        end
    end
    clearvars temptitle tempsubtitle tempheader j;
end
% ------------------------------------------------------------------------
---
% Plot output shock/dynamic response data
if(bPlotOutputCHData)
    for j = 1:size(Strain,2)
        temptitle = strcat(Strain(j).Channel,...
            ' Total Dynamic Strain');
        tempsubtitle = strcat('Sensor
Serial#',string(Strain(j).Serial));
        tempheader = strcat({temptitle;tempsubtitle});
        ID_figure = ID_figure + 1;
        imgFigure = figure(ID_figure);
        plot(Strain(j).Data(:,1),Strain(j).Data(:,2))
        xlim([Strain(j).Data(1,1) Strain(j).Data(end,1)])
        xlabel('Time (ms)','FontName','Times New Roman')
%         ylim([-StrainMax(j,2) StrainMax(j,2)])
        ylabel('Strain (\epsilon)','FontName','Times New Roman')
        title(tempheader,'FontName','Times New Roman')
        grid on
        % Save figure
        if(bSaveOutputPlots)
            FileFigureOUT = temptitle;
```

```
saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
        end
        if(bAutoMode)
            close(imgFigure)
        end
    end
    clearvars temptitle tempsubtitle tempheader j;
end
% ------------------------------------------------------------------------
---
% Plot output shock/dynamic response data
if(bPlotOutputStrainData)
    for j = 1:size(StrainCrop,2)
        temptitle = strcat(StrainCrop(j).Channel,...
            ' Dynamic Strain');
        tempsubtitle = strcat('Sensor
Serial#',string(StrainCrop(j).Serial));
        tempheader = strcat({temptitle;tempsubtitle});
        ID_figure = ID_figure + 1;
        imgFigure = figure(ID_figure);
        plot(StrainCrop(j).Data(:,1),StrainCrop(j).Data(:,2))
        xlim([StrainCrop(j).Data(1,1) StrainCrop(j).Data(end,1)])
        xlabel('Time (ms)','FontName','Times New Roman')
        ylim([-StrainMax(j,2) StrainMax(j,2)])
        ylabel('Strain (\epsilon)','FontName','Times New Roman')
        title(tempheader,'FontName','Times New Roman')
        grid on
        % Save figure
        if(bSaveOutputPlots)
            FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
        end
        if(bAutoMode)
            close(imgFigure)
        end
    end
    clearvars temptitle tempsubtitle tempheader j;
end
% ------------------------------------------------------------------------
---
% Plot output strain and strain-rate data
if(bPlotOutputStrainData)
    for j = 1:size(StrainCrop,2)
        temptitle = strcat(StrainCrop(j).Channel,...
            ' Strain and Strain-rate');
        tempsubtitle = strcat('Sensor
Serial#',string(StrainCrop(j).Serial));
        tempheader = strcat({temptitle;tempsubtitle});
        ID_figure = ID_figure + 1;
        imgFigure = figure(ID_figure);
        xlim([StrainCrop(j).Data(1,1) StrainCrop(j).Data(end,1)])
        xlabel('Time (ms)','FontName','Times New Roman')
        yyaxis left
        plot(StrainCrop(j).Data(:,1),StrainCrop(j).Data(:,2))
```

```
        ylim([-StrainMax(j,2) StrainMax(j,2)])
        ylabel('Strain (\epsilon)','FontName','Times New Roman')
        yyaxis right
        plot(StrainRateCrop(j).Data(:,1),StrainRateCrop(j).Data(:,2))
        ylim([-2*round(StrainRateMax(j,2),1),...
            2*round(StrainRateMax(j,2),1)])
        yl = ylabel('Strain-rate ($\dot{\epsilon}$)','FontName','Times
New Roman');
        set(yl, 'Interpreter','latex');
        title(tempheader,'FontName','Times New Roman')
        grid on
        % Save figure
        if(bSaveOutputPlots)
            FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
        end
        if(bAutoMode)
            close(imgFigure)
        end
    end
    clearvars temptitle tempsubtitle tempheader yl j;
end
% ------------------------------------------------------------------------
---
% Plot output primary strain and primary strain-rate data
if(bPlotOutputPrimaryStrainData)
    temptitle = 'Primary Strain and Strain-rate';
    tempheader = temptitle;
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    xlim([vTime(1,1) vTime(end,1)])
    xlabel('Time (ms)','FontName','Times New Roman')
    yyaxis left
    pl =
plot(vTime,PrimaryStrainCrop(:,1),vTime,PrimaryStrainCrop(:,2));
    pl(1).Color = [0 0.4470 0.7410];
    pl(1).LineStyle = '-';
    pl(2).Color = [0 0.4470 0.7410];
    pl(2).LineStyle = '-.';
    ylim([-max(StrainMax(:,2)) max(StrainMax(:,2))])
    ylabel('Strain (\epsilon)','FontName','Times New Roman')
    yyaxis right
    pl = plot(vTime,PrimaryStrainRateCrop(:,1),...
        vTime,PrimaryStrainRateCrop(:,2));
    pl(1).Color = [0.8500 0.3250 0.0980];
    pl(1).LineStyle = '-';
    pl(2).Color = [0.8500 0.3250 0.0980];
    pl(2).LineStyle = '-.';
    ylim([-2*round(max(PrimaryStrainRate(:,1)),1),...
        2*round(max(PrimaryStrainRate(:,1)),1)])
    yl = ylabel('Strain-rate ($\dot{\epsilon}$)','FontName','Times New
Roman');
    set(yl, 'Interpreter','latex');
    fl = legend('$\epsilon_{xx}$','$\epsilon_{yy}$',...
        '$\dot{\epsilon_{xx}}$','$\dot{\epsilon_{yy}}$',...
```

```
        'FontName','Times New Roman','Location','southeast');
    set(fl, 'Interpreter','latex');
    title(tempheader,'FontName','Times New Roman')
    grid on
    % Save figure
    if(bSaveOutputPlots)
        FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
clearvars temptitle tempsubtitle tempheader yl fl pl j;
% ------------------------------------------------------------------------
---
% Plot output principal stress sigma_1 sigma_2 sigma_3 data
if(bPlotOutputPrincipalStressData)
    for i = 1:size(PrincipalStress,2)
        if(~isempty(PrincipalStress(i).sigma_1))
    temptitle = 'Principal Stresses at A';
    temptitle2 = strcat(' for Strain-rate:',...
        string(PrincipalStress(i).LimitStrainRate),' s^{-1}');
    temptitle = strcat(temptitle,temptitle2);
    tempsubtitle = 'Principal Axis (\theta_p) =';
    tempsubtitle = strcat(tempsubtitle,...
        num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
    tempheader = strcat({temptitle;tempsubtitle});
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    tlim = size(PrincipalStress(i).sigma_1,1);
    pl = plot(vTime(1:tlim),PrincipalStress(i).sigma_1(:,1));
    tempLegend = cell(1,1);
    tempLegend{1} = '\sigma_1';
    hold on
    pl2 = plot(vTime(1:tlim),PrincipalStress(i).sigma_2(:,1));
    tempLegend{2} = '\sigma_2';
    pl3 = plot(vTime(1:tlim),PrincipalStress(i).sigma_3(:,1));
    tempLegend{3} = '\sigma_3';
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
    ylabel('\sigma (Pa)','FontName','Times New Roman')
    xupper = ceil(tlim);
    if(xupper - size(vTime,1) > 1e-6)
        xupper = size(vTime,1);
    end
    if(xupper - 1 < 1e-6)
        xupper = 2;                     % !!!!! HARD-CODE !!!!!
    end
%     xupper = roundn(tlim,1);
    xlim([vTime(1,1) vTime(xupper,1)])
    xlabel('Time (ms)','FontName','Times New Roman')
%     fl = legend('\sigma_1','\sigma_2','Location','southeast');
    fl = legend(tempLegend,'Location','southeast');
    set(fl,'FontName','Times New Roman');
```

```
    tl = title(tempheader,'FontName','Times New Roman');
%     set(tl,'Interpreter','latex');
    grid on
    hold off
    % Save figure
    if(bSaveOutputPlots)
        FileFigureOUT = strcat('Principal Stress','(',string(i),')');

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
        end
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tlim cS1 tl yl
fl j;
clearvars tempLegend tempString pl pl2;
% ------------------------------------------------------------------------
---
% Plot output principal stress data
if(bPlotOutputPrincipalStressData)
    for i = 1:size(PrincipalStress,2)
        if(~isempty(PrincipalStress(i).sigma_1))
    temptitle = 'Principal Stress';
    temptitle2 = strcat(' for Strain-rate:',...
        string(PrincipalStress(i).LimitStrainRate),' s^{-1}');
    temptitle = strcat(temptitle,temptitle2);
    tempsubtitle = 'Principal Axis (\theta_p) =';
    tempsubtitle = strcat(tempsubtitle,...
        num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
    tempheader = strcat({temptitle;tempsubtitle});
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    tlim = size(PrincipalStress(i).sigma_1,1);
%     pl = plot(vTime(1:tlim),PrincipalStress(i).sigma_1(:,1),...
%         vTime(1:tlim),PrincipalStress(i).sigma_2(:,1));
    pl = plot(vTime(1:tlim),PrincipalStress(i).sigma_1(:,1));
%     pl(1).Color = [0 0.4470 0.7410];
%     pl(1).LineStyle = '-';
    pl.Color = [0 0.4470 0.7410];
    pl.LineStyle = '-';
    hold on
    pl2 = plot(vTime(1:tlim),PrincipalStress(i).sigma_2(:,1));
    pl2.Color = pl.Color;
    pl2.LineStyle = '-.';
    set(get(get(pl2,'Annotation'),'LegendInformation'),...
            'IconDisplayStyle','off');
%     pl(2).Color = [0 0.4470 0.7410];
%     pl(2).LineStyle = '-.';
    tempLegend = cell(1,1);
    tempLegend{1} = '\sigma_A';
    if(~isempty(PrincipalStress(i).StressChannel))
    cS1 = -1;
    for j = 1:(size(PrincipalStress(i).StressChannel,2)-4)/2
```

```
        cS1 = cS1 + 3;
        pl = plot(PrincipalStress(i).StressChannel(:,1),...
            PrincipalStress(i).StressChannel(:,cS1),'-');
%         dColor = 1/(2*(size(PrincipalStress(i).StressChannel,2)-
4)/2);
%         pl(1).Color = [j/10*dColor j/2*dColor j/2*dColor];
%         pl(1).LineStyle = '-';
        pl2 = plot(PrincipalStress(i).StressChannel(:,1),...
            PrincipalStress(i).StressChannel(:,cS1+1),'-.');
        pl2.Color = pl.Color;
        set(get(get(pl2,'Annotation'),'LegendInformation'),...
            'IconDisplayStyle','off');
%         pl2.LineStyle = '-.';
%         tempString = strcat(strGreeks{19,1},'_','{CH',...
%             num2str(j+5),'}');
        tempString = strcat('\sigma','_','{S/N:',...
            CHSerial(j,2),'}');
        tempLegend{j+1} = tempString;
    end
    end
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
    ylabel('\sigma (Pa)','FontName','Times New Roman')
    xupper = ceil(tlim);
    if(xupper - size(vTime,1) > 1e-6)
        xupper = size(vTime,1);
    end
    if(xupper - 1 < 1e-6)
        xupper = 2;                     % !!!!! HARD-CODE !!!!!
    end
%     xupper = roundn(tlim,1);
    xlim([vTime(1,1) vTime(xupper,1)])
    xlabel('Time (ms)','FontName','Times New Roman')
%     fl = legend('\sigma_1','\sigma_2','Location','southeast');
    fl = legend(tempLegend,'Location','southeast');
    set(fl,'FontName','Times New Roman');
    tl = title(tempheader,'FontName','Times New Roman');
%     set(tl,'Interpreter','latex');
    grid on
    hold off
    % Save figure
    if(bSaveOutputPlots)
        FileFigureOUT = strcat('Principal Stress','(',string(i),')');

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
        end
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tlim cS1 tl yl
fl j;
clearvars tempLegend tempString pl pl2;
```

```
% ------------------------------------------------------------------------
---
% Plot impact pressure and Hugoniot-elastic-limit
if(bPlotShockHEL)
ID_figure = ID_figure + 1;
F_HEL = Medium.HugoniotElasticLimit;
temptitle = 'Impact interface pressure vs Hugoniot-elastic-limit';
tempsubtitle = strcat(Projectile.Case);
% tempheader = strcat({temptitle,tempsubtitle});
tempheader = {temptitle;tempsubtitle};
imgFigure = figure(ID_figure);
% xneg = -F_cy;
% xpos = F_ty;
x = 0:1:size(ShockWaves.Pressure,1)+1;
yneg = zeros(size(x,2),1);
ypos = yneg;
for k = 1:size(x,2)
    yneg(k) = -F_HEL;
    ypos(k) = F_HEL;
end
clear k;
% x = [xneg, xneg, 0, xpos, xpos, 0, xneg];
% y = [yneg, 0, ypos, ypos, 0, yneg, yneg];
plot(x,yneg,'k-','LineWidth',2);
hold on
pl2 = plot(x,ypos,'k-','LineWidth',2);
grid on
% set(gca,'XTick',[], 'YTick', [])
set(gca,'XTick',[])
set(get(get(pl2,'Annotation'),'LegendInformation'),...
            'IconDisplayStyle','off');
% xlabel('Impacts','FontName','Times New Roman')
ylabel('\sigma (Pa)','FontName','Times New Roman')
title(tempheader,'FontName','Times New Roman')
tempLegend = cell(3,1);
tempLegend{1} = '\sigma_{HEL}';
% legend('Failure envelope','Location','southeast')

% tempLegend{2} = 'Location_A';
if(~isempty(ShockWaves.Pressure))
for j = 1:size(ShockWaves.Pressure,1)-nWads
    pl = scatter(x(j+1),ShockWaves.Pressure(j,1),'bo');
    if(j - 1 > 1e-6)
        set(get(get(pl,'Annotation'),'LegendInformation'),...
            'IconDisplayStyle','off');
    end
end
tempLegend{2} = Projectile.Case;
if(nWads < 1e-6)
else
for j = size(ShockWaves.Pressure,1)-nWads+1:size(ShockWaves.Pressure,1)
    pl = scatter(x(j+1),ShockWaves.Pressure(j,1),'g^');
end
tempLegend{3} = 'Wad';
end
end
```

```
% xlim([0 size(ShockWaves.Pressure,1)+nWads+1])
% ylim([1.1*yneg(1,1) 1.1*ypos(1,1)])
% fl = legend(tempLegend,'FontName','Times New
Roman','Location','southeast')
fl = legend(tempLegend,'Location','southeast');
set(fl,'FontName','Times New Roman');
hold off
if(bSaveOutputPlots)
    FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
end
if(bAutoMode)
    close(imgFigure)
end
clearvars tempLegend strLegend pl fl x yneg ypos i j k;
end
% ------------------------------------------------------------------------
---
% Plot impact pressure pulse profile
if(bPlotShockHEL)
    temptitle = 'Impact interface pressure pulse';
    tempsubtitle = strcat(Projectile.Case);
%     tempheader = strcat({temptitle;tempsubtitle});
    tempheader = {temptitle,tempsubtitle};
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    y_Seal = 0:1e6:ceil(max(ShockWaves.PulseShape(:,2)));
    x_Seal(1,1:size(y_Seal,2)) = t_Seal*0.0254;
    plot(x_Seal,y_Seal,'k-');
    hold on
    plot(ShockWaves.PulseShape(:,1),ShockWaves.PulseShape(:,2));
    ylabel('P (Pa)','FontName','Times New Roman')
%     xlim([0 t_Seal*0.0254])
    xlabel('x (m)','FontName','Times New Roman')
    tl = title(tempheader,'FontName','Times New Roman');
    grid on
    hold off
    % Save figure
    if(bSaveOutputPlots)
        FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT,FileFigureOUT),FileFigureExtensionOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tempLegend;
clearvars E_impact E_dimensionless;
clearvars xupper2 xupper tempXlabel x_Seal y_Seal cS1 tl yl fl i;
% ------------------------------------------------------------------------
---
% Plot output impact energy and strain data
if(bPlotImpactEnergy)
    temptitle = 'Impact Energy vs Stress (rear-face)';
```

```
    tempsubtitle = ' ';
%     tempsubtitle = strcat(tempsubtitle,...
%
num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
    tempheader = {temptitle,tempsubtitle};
%     tempheader = strcat({temptitle;tempsubtitle});
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    k = 1;
    for i = 1:size(ImpactEnergy,2)
        if(~isempty(ImpactEnergy(i).ImpactEnergy))
        E_impact = ImpactEnergy(i).ImpactEnergy;
        % Plot dimensionless impact energy parameter
        E_dimensionless = ImpactEnergy(i).E_dimensionless;
        if(abs(k-1) < 1e-6)
            pl = scatter(E_impact,ImpactEnergy(i).StressMax,'o');
        elseif(abs(k-2) < 1e-6)
            pl = scatter(E_impact,ImpactEnergy(i).StressMax,'s');
        elseif(abs(k-3) < 1e-6)
            pl = scatter(E_impact,ImpactEnergy(i).StressMax,'d');
        elseif(abs(k-4) < 1e-6)
            pl = scatter(E_impact,ImpactEnergy(i).StressMax,'^');
        elseif(abs(k-5) < 1e-6)
            pl = scatter(E_impact,ImpactEnergy(i).StressMax,'*');
            % Reset k index
            k = 0;
        end
        tempLegend{i} = ImpactEnergy(i).Case;
        hold on
        end
        k = k + 1;
    end
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
%     ylabel('\epsilon','FontName','Times New Roman')
    ylabel('\sigma (Pa)','FontName','Times New Roman')
    xupper = -Inf;
    xupper2 = 0;
    for j = 1:size(ImpactEnergy,2)
        % Convert energy to (J)
        xupper2 = max(ImpactEnergy(j).ImpactEnergy);
        xupper = max(xupper2,xupper);
    end
    xupper = 1.5*roundn(xupper,2);
    xlim([0 xupper])
    xlabel('Impact Energy (J)','FontName','Times New Roman')
    fl = legend(tempLegend,'Location','eastoutside');
    set(fl,'FontName','Times New Roman');
%     tl = title(tempheader,'FontName','Times New Roman');
    tl = title(tempheader,'FontName','Times New Roman');
%     set(tl,'Interpreter','latex');
    grid on
    hold off
    % Save figure
    if(bSaveOutputPlots)
        FileFigureOUT = strcat('Impact Energy');
```

```
saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tempLegend;
clearvars E_impact E_dimensionless;
clearvars xupper2 xupper cS1 tl yl fl i j k;
% ------------------------------------------------------------------
---
% Plot output impact energy and strain data
if(bPlotImpactEnergy)
    temptitle = 'Impact Energy vs Penetration';
    tempsubtitle = ' ';
%     tempsubtitle = strcat(tempsubtitle,...
%
num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
%     tempheader = strcat({temptitle;tempsubtitle});
    tempheader = {temptitle,tempsubtitle};
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    k = 1;
    j = 0;
    for i = 1:size(ImpactEnergy,2)
        if(~isempty(ImpactEnergy(i).ImpactEnergy))
        E_impact = ImpactEnergy(i).ImpactEnergy;
        % Plot penetration depth (in) -> (m)
%         tempZa = ImpactEnergy(i).PenetrationActual * 0.0254;
%         tempZe = ImpactEnergy(i).PenetrationEst * 0.0254;
        tempZa = ImpactEnergy(i).PenetrationActual;
        tempZe = ImpactEnergy(i).PenetrationEst;
        if(abs(k-1) < 1e-6)
            pl = scatter(E_impact,tempZa,'o');
            hold on
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
            pl2 = scatter(E_impact,tempZe,'o');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
            pl3 = plot([E_impact;E_impact;],[tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-2) < 1e-6)
            pl = scatter(E_impact,tempZa,'s');
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
            pl2 = scatter(E_impact,tempZe,'s');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
            pl3 = plot([E_impact;E_impact;],[tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-3) < 1e-6)
```

```
            pl = scatter(E_impact,tempZa,'d');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
            pl2 = scatter(E_impact,tempZe,'d');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
            pl3 = plot([E_impact;E_impact;],[tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-4) < 1e-6)
            pl = scatter(E_impact,tempZa,'^');
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
            pl2 = scatter(E_impact,tempZe,'^');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
            pl3 = plot([E_impact;E_impact;],[tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-5) < 1e-6)
            pl = scatter(E_impact,tempZa,'*');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
            pl2 = scatter(E_impact,tempZe,'*');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
            pl3 = plot([E_impact;E_impact;],[tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
            % Reset k index
            k = 0;
        end
        hold on
        end
        k = k + 1;
    end
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
%     ylabel('\epsilon','FontName','Times New Roman')
    ylabel('Penetration (m)','FontName','Times New Roman')
    xupper = -Inf;
    xupper2 = 0;
    for j = 1:size(ImpactEnergy,2)
        % Convert energy to (J)
        xupper2 = max(ImpactEnergy(j).ImpactEnergy);
        xupper = max(xupper2,xupper);
    end
    xupper = 1.5*roundn(xupper,2);
    xlim([0 xupper])
    xlabel('Impact Energy (J)','FontName','Times New Roman')
    fl = legend(tempLegend,'Location','eastoutside');
    set(fl,'FontName','Times New Roman');
%     tl = title(tempheader,'FontName','Times New Roman');
    tl = title(tempheader,'FontName','Times New Roman');
%     set(tl,'Interpreter','latex');
    grid on
```

```
    hold off
    % Save figure
    if(bSaveOutputPlots)
        FileFigureOUT = strcat(temptitle);

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tempLegend;
clearvars E_impact E_dimensionless;
clearvars xupper2 xupper cS1 tl yl fl i j k;
% ------------------------------------------------------------------
---
% Plot output impact energy and strain data
if(bPlotImpactEnergy)
    temptitle = 'Dimensionless Impact Energy vs Stress (rear-face)';
    tempsubtitle = ' ';
%     tempsubtitle = 'Principal Axis (\theta_p) =';
%     tempsubtitle = strcat(tempsubtitle,...
%
num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
%     tempheader = strcat({temptitle;tempsubtitle});
    tempheader = {temptitle,tempsubtitle};
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    k = 1;
    for i = 1:size(ImpactEnergy,2)
        if(~isempty(ImpactEnergy(i).ImpactEnergy))
        E_impact = ImpactEnergy(i).ImpactEnergy;
        % Plot dimensionless impact energy parameter
        E_dimensionless = ImpactEnergy(i).E_dimensionless;
        if(abs(k-1) < 1e-6)
            pl =
scatter(E_dimensionless,ImpactEnergy(i).StressMax,'o');
        elseif(abs(k-2) < 1e-6)
            pl =
scatter(E_dimensionless,ImpactEnergy(i).StressMax,'s');
        elseif(abs(k-3) < 1e-6)
            pl =
scatter(E_dimensionless,ImpactEnergy(i).StressMax,'d');
        elseif(abs(k-4) < 1e-6)
            pl =
scatter(E_dimensionless,ImpactEnergy(i).StressMax,'^');
        elseif(abs(k-5) < 1e-6)
            pl =
scatter(E_dimensionless,ImpactEnergy(i).StressMax,'*');
            % Reset k index
            k = 0;
        end
        tempLegend{i} = ImpactEnergy(i).Case;
        hold on
        end
```

```
        k = k + 1;
    end
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
%     ylabel('\epsilon','FontName','Times New Roman')
    ylabel('\sigma (Pa)','FontName','Times New Roman')
    xupper = -Inf;
    xupper2 = 0;
    for j = 1:size(ImpactEnergy,2)
        xupper2 = max(ImpactEnergy(j).E_dimensionless);
        xupper = max(xupper2,xupper);
    end
    xupper = 1.5*xupper;
    xlim([0 xupper])
%     tempXlabel = strcat('Dimensionless Impact Energy (',...
%         'm_p*v^2*\rho_t*d_p^3/(m_p*C_{t,0}^2))');
    tempXlabel = strcat('Modified Weber Number, \xi');
    xlabel(tempXlabel,'FontName','Times New Roman')
    fl = legend(tempLegend,'Location','eastoutside');
    set(fl,'FontName','Times New Roman');
%     tl = title(tempheader,'FontName','Times New Roman');
    tl = title(tempheader,'FontName','Times New Roman');
    grid on
    hold off
    % Save figure
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    % Zoom in on clustered data
    xlim([0 xupper/2])
%     ylim([0 1e6])
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = strcat(temptitle,'(2)');

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    % Zoom in on clustered data
    xlim([0 xupper/4])
%     ylim([0 1e6])
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = strcat(temptitle,'(3)');

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
```

```
clearvars temptitle tempsubtitle temptitle2 tempheader tempLegend;
clearvars E_impact E_dimensionless;
clearvars xupper2 xupper tempXlabel cS1 tl yl fl i;
% ------------------------------------------------------------------------
---
% Plot output impact energy and penetration estimate
if(bPlotImpactEnergy)
    temptitle = 'Dimensionless Impact Energy vs Penetration Models';
    tempsubtitle = ' ';
%     tempsubtitle = 'Principal Axis (\theta_p) =';
%     tempsubtitle = strcat(tempsubtitle,...
%
num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
%     tempheader = strcat({temptitle;tempsubtitle});
    tempheader = {temptitle,tempsubtitle};
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    j = 0;
    k = 1;
    for i = 1:size(ImpactEnergy,2)
        if(~isempty(ImpactEnergy(i).ImpactEnergy))
        % Convert energy from (lb*ft^2/s^2) to (J)
        E_impact = ImpactEnergy(i).ImpactEnergy*0.0421401101;
        % Plot dimensionless impact energy parameter
        E_dimensionless = ImpactEnergy(i).E_dimensionless;
        % Plot penetration depth (in) -> (m)
%         tempZa = ImpactEnergy(i).PenetrationVolume * 0.0254;
%         tempZe = ImpactEnergy(i).PenetrationEst * 0.0254;
        tempZa = ImpactEnergy(i).PenetrationVolume;
        tempZe = ImpactEnergy(i).PenetrationEst;
        if(abs(k-1) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'o');
            hold on
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Derived');
            pl2 = scatter(E_dimensionless,tempZe,'o');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Model');
            pl3 = plot([E_dimensionless;E_dimensionless;],...
                [tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-2) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'s');
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Derived');
            pl2 = scatter(E_dimensionless,tempZe,'s');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Model');
            pl3 = plot([E_dimensionless;E_dimensionless;],...
                [tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-3) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'d');
            j = j + 1;
```

```
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Derived');
            pl2 = scatter(E_dimensionless,tempZe,'d');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Model');
            pl3 = plot([E_dimensionless;E_dimensionless;],...
                [tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-4) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'^');
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Derived');
            pl2 = scatter(E_dimensionless,tempZe,'^');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Model');
            pl3 = plot([E_dimensionless;E_dimensionless;],...
                [tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
        elseif(abs(k-5) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'*');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Derived');
            pl2 = scatter(E_dimensionless,tempZe,'*');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case,' Model');
            pl3 = plot([E_dimensionless;E_dimensionless;],...
                [tempZe;tempZa;],':k');
            set(get(get(pl3,'Annotation'),'LegendInformation'),...
                'IconDisplayStyle','off');
            % Reset k index
            k = 0;
        end
%         set(get(get(pl2,'Annotation'),'LegendInformation'),...
%             'IconDisplayStyle','off');
%         tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
%         j = j + 1;
%         tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
        hold on
        end
        k = k + 1;
    end
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
%     ylabel('\epsilon','FontName','Times New Roman')
    ylabel('Penetration (m)','FontName','Times New Roman')
    xupper = -Inf;
    xupper2 = 0;
    for j = 1:size(ImpactEnergy,2)
        xupper2 = max(ImpactEnergy(j).E_dimensionless);
        xupper = max(xupper2,xupper);
    end
    xupper = 1.5*xupper;
    xlim([0 xupper])
%     tempXlabel = strcat('Dimensionless Impact Energy (',...
%         'm_p*v^2*\rho_t*d_p^3/(m_p*C_{t,0}^2))');
```

```
    tempXlabel = strcat('Modified Weber Number, \xi');
    xlabel(tempXlabel,'FontName','Times New Roman');
    fl = legend(tempLegend,'Location','eastoutside');
    set(fl,'FontName','Times New Roman');
%     tl = title(tempheader,'FontName','Times New Roman');
    tl = title(tempheader,'FontName','Times New Roman');
    grid on
    hold off
    % Save figure
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    % Zoom in on clustered data
    xlim([0 xupper/2])
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = strcat(temptitle,'(2)');

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tempLegend;
clearvars E_impact E_dimensionless;
clearvars xupper2 xupper tempXlabel cS1 tl yl fl i j k;
% ------------------------------------------------------------------------
---
% Plot output impact energy and penetration estimate
if(bPlotImpactEnergy)
    temptitle = 'Dimensionless Impact Energy vs Penetration';
    tempsubtitle = ' ';
%     tempsubtitle = 'Principal Axis (\theta_p) =';
%     tempsubtitle = strcat(tempsubtitle,...
%
num2str(max(PrincipalStress(i).pTheta(:,1)),'%.1f'),char(176));
%     tempheader = strcat({temptitle;tempsubtitle});
    tempheader = {temptitle,tempsubtitle};
    ID_figure = ID_figure + 1;
    imgFigure = figure(ID_figure);
    j = 0;
    k = 1;
    for i = 1:size(ImpactEnergy,2)
        if(~isempty(ImpactEnergy(i).ImpactEnergy))
        E_impact = ImpactEnergy(i).ImpactEnergy;
        % Plot dimensionless impact energy parameter
        E_dimensionless = ImpactEnergy(i).E_dimensionless;
        % Plot penetration depth (in) -> (m)
%         tempZa = ImpactEnergy(i).PenetrationActual * 0.0254;
        tempZa = ImpactEnergy(i).PenetrationActual;
```

```
        if(abs(k-1) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'o');
            hold on
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case);
        elseif(abs(k-2) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'s');
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case);
        elseif(abs(k-3) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'d');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case);
        elseif(abs(k-4) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'^');
            j = j+1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case);
        elseif(abs(k-5) < 1e-6)
            pl = scatter(E_dimensionless,tempZa,'*');
            j = j + 1;
            tempLegend{j} = strcat(ImpactEnergy(i).Case);
            % Reset k index
            k = 0;
        end
%         set(get(get(pl2,'Annotation'),'LegendInformation'),...
%             'IconDisplayStyle','off');
%         tempLegend{j} = strcat(ImpactEnergy(i).Case,' Actual');
%         j = j + 1;
%         tempLegend{j} = strcat(ImpactEnergy(i).Case,' Estimate');
        hold on
        end
        k = k + 1;
    end
%     ylim([-max(PrincipalStress(i).sigma_1(:,1))
max(PrincipalStress(i).sigma_2(:,1))])
%     ylabel('\epsilon','FontName','Times New Roman')
    ylabel('Penetration (m)','FontName','Times New Roman')
    xupper = -Inf;
    xupper2 = 0;
    for j = 1:size(ImpactEnergy,2)
        xupper2 = max(ImpactEnergy(j).E_dimensionless);
        xupper = max(xupper2,xupper);
    end
    xupper = 1.5*xupper;
    xlim([0 xupper])
%     tempXlabel = strcat('Dimensionless Impact Energy (',...
%         'm_p*v^2*\rho_t*d_p^3/(m_p*C_{t,0}^2))');
    tempXlabel = strcat('Modified Weber Number, \xi');
    xlabel(tempXlabel,'FontName','Times New Roman');
    fl = legend(tempLegend,'Location','eastoutside');
    set(fl,'FontName','Times New Roman');
%     tl = title(tempheader,'FontName','Times New Roman');
    tl = title(tempheader,'FontName','Times New Roman');
    grid on
    hold off
    % Save figure
```

```
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = temptitle;

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    % Zoom in on clustered data
    xlim([0 xupper/2])
    if(bSaveOutputPlots)
%         FileFigureOUT = strcat('Impact Energy Dimensionless');
        FileFigureOUT = strcat(temptitle,'(2)');

saveas(gcf,strcat(FileFigDirOUT_final,FileFigureOUT),FileFigureExtensio
nOUT);
    end
    if(bAutoMode)
        close(imgFigure)
    end
end
clearvars temptitle tempsubtitle temptitle2 tempheader tempLegend;
clearvars E_impact E_dimensionless;
clearvars xupper2 xupper tempXlabel cS1 tl yl fl i j k;
```

---

mStrainCalcs.m

```
%
=========================================================================
=
% Isolate strain sensor channels from the data and convert from voltage
to
% strain, isolate relevant data
%
=========================================================================
=

% Convert Voltage signal to strain - ignore non-strain sensor data
SensorSensitivity = zeros(size(CHData,2),1);
for i = 1:size(SensorSensitivity,1)
    if(isnan(CHData(i).Serial))
    else
        for j = 1:size(dataStrainSensorCal,1)
            if(abs(dataStrainSensorCal(j,1)-CHData(i).Serial)<1e-6)
                SensorSensitivity(i) = dataStrainSensorCal(j,2);
                break;
            end
        end
    end
end
clearvars i j;
% Clear non-strain sensor sensitivities from the array
```

```
tempNaNSensorSerial = isnan(ID_SensorSerial);
SensorSensitivity(tempNaNSensorSerial) = [];
% Convert Sensitivity from mV/microstrain to V/strain
SensorSensitivity = SensorSensitivity * 1000;
CHDataStrain = CHData;
CHDataStrain(tempNaNSensorSerial) = [];
clearvars tempNaNSensorSerial;
% Initialize channel and sensor array
CHSerial = [];
for i = 1:size(CHDataStrain,2)
    CHSerial = [CHSerial;
        CHDataStrain(i).Channel, string(CHDataStrain(i).Serial)];
end
clear i;
% Eliminate rosette sensors
CHSerial2 = CHSerial;
for i = 1:size(CHSerial2,1)
    if(abs(Serial_A - str2num(char(CHSerial2(i,2)))) < 1e-6)
        CHSerial2(i,2) = 'Serial_A';
    elseif(abs(Serial_B - str2num(char(CHSerial2(i,2)))) < 1e-6)
        CHSerial2(i,2) = 'Serial_B';
    elseif(abs(Serial_C - str2num(char(CHSerial2(i,2)))) < 1e-6)
        CHSerial2(i,2) = 'Serial_C';
    end
end
CHSerial2(strcmp(CHSerial2(:,2),'Serial_A'),:) = [];
CHSerial2(strcmp(CHSerial2(:,2),'Serial_B'),:) = [];
CHSerial2(strcmp(CHSerial2(:,2),'Serial_C'),:) = [];
% ------------------------------------------------------------------------
---
% Isolate shock/dynamic response
V_unstrained = zeros(size(CHDataStrain,2),1);
V_stddev = zeros(size(CHDataStrain,2),1);
for i = 1:size(CHDataStrain,2)
    % Determine the mean unstrained voltage and std dev for each
channel
    V_unstrained(i,1) = mean(CHDataStrain(i).Data(1:ID_trigger,2));
    tempDev = CHDataStrain(i).Data(:,2) - V_unstrained(i,1);
    V_stddev(i,1) =
sqrt(sum(tempDev.^2)/(size(CHDataStrain(i).Data,1)));
end
clearvars i tempDev;
% Find initial shock response
ID_Shock = zeros(size(CHDataStrain,2),3);
% Col1 locates front end of impulse, Col2 locates the rear end, C3=C2-
C1
tempsigma_1 = 6;            % stddev's used in identifying impulse
location
tempsigma_2 = 12;
for i = 1:size(CHDataStrain,2)
    for j = 1:size(CHDataStrain(1).Data,1)
        if(V_unstrained(i) + tempsigma_1 * V_stddev(i) - ...
                abs(CHDataStrain(i).Data(j,2)) < 1e-6)
            break
        end
    end
```

```
    ID_Shock(i,1) = j;
    for j = size(CHDataStrain(1).Data,1):-1:1
        if(V_unstrained(i) + tempsigma_2 * V_stddev(i) - ...
                abs(CHDataStrain(i).Data(j,2)) < 1e-6)
            break
        end
    end
    ID_Shock(i,2) = j;
    ID_Shock(i,3) = ID_Shock(i,2) - ID_Shock(i,1);
end
clearvars i j tempsigma_1 tempsigma_2;
% Clip to time = 0
for i = 1:size(CHDataStrain(1).Data,1)
    % Find where time changes from (-) to (+)
    if(-CHDataStrain(1).Data(i,1) < 1e-6)
        break
    end
end
ID_tzero = i;
clear i;
% Add strain sensor angle to struct
[CHDataStrain(:).SensorAngle] = deal(zeros(size(CHDataStrain,2),1));
for i = 1:size(CHDataStrain,2)
    for j = 1:size(dimStrainSensorLoc,1)
        if(abs(dimStrainSensorLoc(j,1) - CHDataStrain(i).Serial) < 1e-
6)
            CHDataStrain(i).SensorAngle = ...
                dimStrainSensorLoc(j,size(dimStrainSensorLoc,2));
            break
        end
    end
end
clearvars i j;
% Reshape and isolate shock/dynamic response data arrays
StrainCrop = CHDataStrain;
vShockLim = roundn(max(ID_Shock(:,3)),2)+iStrainArraySize;
vShock = zeros(vShockLim,2);
k_buffer = 50;                      % shifts values to be copied to
left
% Ensure vShock does not attempt to reference in excess of CHData array
if(max(ID_Shock(:,1))+size(vShock,1) - size(CHDataStrain(1).Data,1) >
1e-6)
    k_buffer = int64(max(ID_Shock(:,1))+size(vShock,1)...
        - size(CHDataStrain(1).Data,1));
    fprintf('NOTE>> Shock impulse detection failure during analysis,
possible array reference error\n');
    fprintf('NOTE>> Cropping strain response plots to time zero\n');
    bCropToTimeZero = true;
end
if(bCropToTimeZero)
%     for i =1:size(CHDataStrain,2)
%         StrainCrop(i).Data(vShockLim+1:end,:) = [];
%         for j = 1:vShockLim
%             k = ID_tzero -1 + j;
%             StrainCrop(i).Data(j,1) = CHDataStrain(i).Data(k,1);
%             StrainCrop(i).Data(j,2) = CHDataStrain(i).Data(k,2);
```

```
%         end
%     end
    for i = 1:size(StrainCrop,2)
        StrainCrop(i).Data(1:ID_tzero,:) = [];
        StrainCrop(i).Data(vShockLim+1:end,:) = [];
    end
else
    for i = 1:size(CHDataStrain,2)
        for j = 1:size(vShock,1)
            k = ID_Shock(i,1)-k_buffer + j;
            vShock(j,1) = CHDataStrain(i).Data(k,1);
            vShock(j,2) = CHDataStrain(i).Data(k,2);
        end
        StrainCrop(i).Data = vShock;
    end
end
clearvars i j k k_buffer;
% Convert sensor measurands to strain - ignore non-strain sensor data
% Pre-allocate strain matrices
% Strain = zeros(size(CHDataStrain,2),size(CHDataStrain(1).Data,1));
Strain = CHDataStrain;
StrainMax = zeros(size(CHDataStrain,2),2);
StrainRate = Strain;
StrainRateMax = StrainMax;
% Loop through strain sensor data
for i = 1:size(CHDataStrain,2)
    % Determine the unstrained voltage for each channel
    V_unstrained(i,1) = mean(CHDataStrain(i).Data(1:ID_trigger,2));
    % Convert voltage to strain (eps) --> strain = (V)/Sensitivity
(V/eps)
    Strain(i).Data(:,2) = (CHDataStrain(i).Data(:,2) - ...
        V_unstrained(i,1)) * 1/SensorSensitivity(i);
    StrainCrop(i).Data(:,2) = (StrainCrop(i).Data(:,2) - ...
        V_unstrained(i,1)) * 1/SensorSensitivity(i);
    % Find location of maximum strain StrainMax(ID, Value)
    [StrainMax(i,2), StrainMax(i,1)] = max(Strain(i).Data(:,2));
end
% Determine strain rate (eps/ms) --> convert to (eps/s)
for i = 1:size(Strain,2)
    for j = 2:size(Strain(i).Data,1)
        StrainRate(i).Data(j,2) = 1000.0 * ((Strain(i).Data(j,2) - ...
            Strain(i).Data(j-1,2))/...
            (Strain(i).Data(j,1) - Strain(i).Data(j-1,1)));
    end
    % Correct for strain rate unstrained bias
    StrainRate(i).Data(:,2) = StrainRate(i).Data(:,2) - ...
        mean(StrainRate(i).Data(1:ID_trigger-1,2));
%     [StrainRateMax(i,2),StrainRateMax(i,1)] =
max(StrainRate(i).Data(:,2));
    [StrainRateMax(i,2),StrainRateMax(i,1)] =
max(abs(StrainRate(i).Data(:,2)));
end
% Initialize strain-rate crop
StrainRateCrop = StrainRate;
if(bCropToTimeZero)
    for i = 1:size(StrainCrop,2)
```

```
        StrainRateCrop(i).Data(1:ID_tzero,:) = [];
        StrainRateCrop(i).Data(size(StrainCrop(1).Data,1)+1:end,:) =
[];
    end
else
    for i = 1:size(StrainCrop,2)
        StrainRateCrop(i).Data(1:ID_Shock(1,1),:) = [];
        StrainRateCrop(i).Data(size(StrainCrop(1).Data,1)+1:end,:) =
[];
    end
end
clearvars i j;
% Convert sensor orientation to radians
% % dimStrainSensorLoc(:,size(dimStrainSensorLoc,2)) = ...
% %     dimStrainSensorLoc(:,size(dimStrainSensorLoc,2)) * pi()/180.0;
% ------------------------------------------------------------------------
---
% Transform Strain to determine Principal Axes
% eps_xp = eps_x * cos^2(th) + eps_y * sin^2(th) + gamma_xy * sin(th) *
cos(th)
% eps_yp = eps_x * sin^2(th) + eps_y * cos^2(th) - gamma_xy * sin(th) *
cos(th)
% gamma_xyp = 2*(eps_y - eps_x) * sin(th) * cos(th) +
gamma_xy(cos^2(th) - sin^2(th))
% For a Delta Rosette (sensor A is 0 deg, B is 60 deg, C is 120 deg)
% eps_1,2 = 1/3 * (eps_A + eps_B + eps_C +/- sqrt(2 * ((eps_A -
eps_B)^2 +
% (eps_B - eps_C)^2 + (eps_C - eps_A)^2)
% Principal angle 'theta' is acute angle from axis of sensor
% Sensor A to nearest principal axis
% PrincipalAxes = zeros(size(Strain(1).Data,1),3);
PrimaryStrain = zeros(size(Strain(1).Data,1),2);
PrimaryStrainRate = PrimaryStrain;
% PrincipalTheta = zeros(size(Strain(1).Data,1),1);
StrainShear = PrimaryStrain(:,1);
% Create time array for plotting principal strains
vTime = Strain(1).Data(:,1);
for i = 1:size(Strain,2)
    if(abs(Strain(i).Serial - Serial_A) < 1e-6)
        eps_A = Strain(i).Data(:,2);        % Oriented 0 deg
    elseif(abs(Strain(i).Serial - Serial_B) < 1e-6)
        eps_B = Strain(i).Data(:,2);        % Oriented 60 or 240 deg to
A
        tempThetaRosette = Strain(i).SensorAngle;
    elseif(abs(Strain(i).Serial - Serial_C) < 1e-6)
        eps_C = Strain(i).Data(:,2);        % Oriented 120 deg to A
    end
end
% Convert strain data to primary axes
tempThetaRosette = tempThetaRosette * pi()/180;
if(bStrainRosetteConversion)
    eps_xx = eps_A;
    eps_yy = ((eps_A - 2*eps_B)*sin(4*tempThetaRosette) + ...
        2*eps_C*sin(2*tempThetaRosette))/...
        ((4*(sin(tempThetaRosette))^2)*(sin(2*tempThetaRosette)));
    eps_xy = (2*eps_A*((((sin(tempThetaRosette))^2)*...
```

```
        ((cos(2*tempThetaRosette))^2)-((sin(2*tempThetaRosette))^2)*...

((cos(tempThetaRosette))^2)))+2*(eps_B*(sin(2*tempThetaRosette)^2)...
        -eps_C*(sin(tempThetaRosette)^2)))/...
        (4*(sin(tempThetaRosette)^2)*(sin(2*tempThetaRosette)));
    PrimaryStrain(:,1) = eps_xx;
    PrimaryStrain(:,2) = eps_yy;
    StrainShear(:,1) = eps_xy;
else
    PrimaryStrain(:,1) = (1/3) * (eps_A + eps_B + eps_C + sqrt(2 * (...
        (eps_A - eps_B).^2 + (eps_B - eps_C).^2 + (eps_C -
eps_A).^2)));
    PrimaryStrain(:,2) = (1/3) * (eps_A + eps_B + eps_C - sqrt(2 * (...
        (eps_A - eps_B).^2 + (eps_B - eps_C).^2 + (eps_C -
eps_A).^2)));
%     PrincipalTheta(:,1) = (1/2) * atan((sqrt(3)* eps_B .* eps_C)./...
%         (2 * eps_A - eps_B - eps_C));
end
% Determine primary strain-rate (eps/ms) --> convert to (eps/s)
for i = 2:size(PrimaryStrain,1)
    PrimaryStrainRate(i,1) = 1000 * ((PrimaryStrain(i,1) - ...
        PrimaryStrain(i-1,1))/...
        (vTime(i) - vTime(i-1)));
    PrimaryStrainRate(i,2) = 1000 * ((PrimaryStrain(i,2) - ...
        PrimaryStrain(i-1,2))/...
        (vTime(i) - vTime(i-1)));
end
% Convert theta to deg
% PrincipalTheta = PrincipalTheta * 180.0/pi();
% % Resize array to reduce processing memory and increase chart
readability
% iResizeLimit = int64(1000);
% iResizeIndex = size(PrincipalTheta,1)/iResizeLimit;
% PrincipalThetaResize = zeros(iResizeLimit,1);
% j1 = 1;
% for i = 1:iResizeLimit
%     j2 = j1 + iResizeIndex - 1;
%     if(j1 - size(PrincipalTheta,1) > 1e-6)
%         j1 = size(PrincipalTheta,1) - 1;
%     elseif(j2 - size(PrincipalTheta,1) > 1e-6)
%         j2 = size(PrincipalTheta,1);
%         PrincipalThetaResize(i) = mean(PrincipalTheta(j1:j2));
%         PrincipalThetaResize(i+1:size(PrincipalThetaResize,1)) = ...
%             mean(PrincipalTheta(j1:j2));
%         break
%     end
%     PrincipalThetaResize(i) = mean(PrincipalTheta(j1:j2));
%     j1 = j2 + 1;
% end
% clearvars j1 j2;
if(bCropToTimeZero)
    vTime(1:ID_tzero,:) = [];
    vTime(size(StrainCrop(1).Data,1)+1:end,:) = [];
    PrimaryStrainCrop = PrimaryStrain;
    StrainShearCrop = StrainShear;
    PrimaryStrainRateCrop = PrimaryStrainRate;
```

```
%     PrincipalThetaCrop = PrincipalTheta;
    PrimaryStrainCrop(1:ID_tzero,:) = [];
    PrimaryStrainCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
    StrainShearCrop(1:ID_tzero,:) = [];
    StrainShearCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
    PrimaryStrainRateCrop(1:ID_tzero,:) = [];
    PrimaryStrainRateCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
%     PrincipalThetaCrop(1:ID_tzero,:) = [];
%     PrincipalThetaCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
else
    vTime(1:ID_Shock(1,1),:) = [];
    vTime(size(StrainCrop(1).Data,1)+1:end,:) = [];
    PrimaryStrainCrop = PrimaryStrain;
    StrainShearCrop = StrainShear;
    PrimaryStrainRateCrop = PrimaryStrainRate;
%     PrincipalThetaCrop = PrincipalTheta;
    PrimaryStrainCrop(1:ID_Shock(1,1),:) = [];
    PrimaryStrainCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
    StrainShearCrop(1:ID_Shock(1,1),:) = [];
    StrainShearCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
    PrimaryStrainRateCrop(1:ID_Shock(1,1),:) = [];
    PrimaryStrainRateCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
%     PrincipalThetaCrop(1:ID_Shock(1,1),:) = [];
%     PrincipalThetaCrop(size(StrainCrop(1).Data,1)+1:end,:) = [];
end
```

---

mVelocityEstimate.m

```
%
=========================================================================
=
% Analyze data from the chronograph channels - estimate velocity by
% determining the trigger time
%
=========================================================================
=
% -------------------------------------------------------------------
---
% Acquire test case projectile information
Projectile = [];
for i = 1:size(ProjectileLibrary,2)
    if(strcmp(FileTestCaseFolder,ProjectileLibrary(i).Case))
        % Check for repeat tests
        if(i < iCounter)
            % Possible repeat test on a later test date
            % Store projectile assignment
            Projectile = ProjectileLibrary(i);
        else
            Projectile = ProjectileLibrary(i);
            break
        end
```

```
    end
end
if(isempty(Projectile))
    error('WARNING>> Projectile not identified by Test Case');
end
clear i;
% Include wad impacts?
if(strcmp(FileTestDateFolder,'20190731'))
    nWads = 2;
else
    nWads = 1;
end
% !!!!! HARD CODE WAD MATERIAL FOR WOOD !!!!!
Wad = MediaLibrary(5);
% ------------------------------------------------------------------------
---
% Estimate impact velocity from data
% Attempt to read from CH1 and CH2 the time projectile or wad will pass
by
% chronograph, else estimate from cannon calibration curve data
if(strcmp(CHData(1).Channel,'CH1')&&strcmp(CHData(2).Channel,'CH2'))
%     Find indices for detection times for laser sensor 1 and 2
    for i = 1:size(CHData(1).Data,1)
        if(abs(CHData(1).Data(i,2)-CHData(1).Data(1,2)) - ...
                abs(0.75*CHData(1).Data(1,2))> 1e-6)
            break
        end
    end
    for j = 1:size(CHData(2).Data,1)
        if(abs(CHData(2).Data(j,2)-CHData(2).Data(1,2)) - ...
                abs(0.75*CHData(1).Data(2,2))> 1e-6)
            break
        end
    end
    ID_trigger = i;
%     Determine dt between laser sensors --> then velocity
    t_trigger = CHData(2).Data(j,1) - CHData(2).Data(i,1);
    velocity_impact = dimSetupDimensions.Chronograph_dz / t_trigger;
%     Convert velocity_impact (in/ms) to (ft/s)
    velocity_impact = velocity_impact * 1000.0/12.0;
    % Store measured velocity for reference (ft/s)
    velocity_measured = velocity_impact;
    fprintf('\n-----------------------------------------------------');
    fprintf('\nMeasured velocity (ft/s):\t%.2f',velocity_impact);
    fprintf('\n-----------------------------------------------------');
    if(velocity_impact - velocity_impact_limit < 1e-6 || ...
        velocity_impact > 5e3)
        % Estimate using secondary approach -> attempt to catch wad
        tempVelocity = CHData(1).Data;
        tempdV = CHData(1).Data(:,2) > 9.125;
        tempVelocity(tempdV,:) = [];
        if(~isempty(tempVelocity))
        tempdV = tempVelocity(:,2) < 2.5;
        tempVelocity(tempdV,:) = [];
%         tempdV = CHData(1).Data(:,2) > 0.5*max(CHData(1).Data(:,2));
%         tempVelocity(tempdV,:) = [];
```

```
        tempdt = zeros(size(CHData(1).Data,1),1);
        tempVold = tempVelocity(1,2);
        for i = 2:size(CHData(1).Data,1)
            tempVnew = tempVelocity(i,2);
            if(tempVnew > tempVold)
                % Laser is open after sensing object
                break
            end
            tempVold = tempVnew;
        end
        dt_obj = (tempVelocity(i,1) - tempVelocity(1,1))/1000;  % [s]
        else
            % Chronograph did not trip -> drive velocity -> zero
            dt_obj = 1e3;
        end
        if(strcmp(FileTestDateFolder,'20190731'))
            dz_obj = dimSetupDimensions.Wad_dz;                 % [in]
        else
            dz_obj = dimSetupDimensions.Wad2_dz;
        end
        velocity_impact_new = (dz_obj/12) / dt_obj;             %
[ft/s]
        % Check if calculated velocity is probable
        if(velocity_impact_new - 2e3 > 1e-6)
            % Improbable velocity -> estimate using cannon calibration
data
            velocity_impact = fVelocityInterpolate(Projectile.Mass,...
                velocity_impact_estimate);
            fprintf('\n-------------------------------------------------
---');
            fprintf('\nEstimating impact velocity using calibration
data');
            fprintf('\n-------------------------------------------------
---');
        elseif(velocity_impact_new < velocity_impact_limit)
            % Improbable velocity -> estimate using cannon calibration
data
            velocity_impact = fVelocityInterpolate(Projectile.Mass,...
                velocity_impact_estimate);
            fprintf('\n-------------------------------------------------
---');
            fprintf('\nEstimating impact velocity using calibration
data');
            fprintf('\n-------------------------------------------------
---');
        else
            % Check measured wad velocity for large deviations from cal
            velocity_measured = velocity_impact_new;
            velocity_calibrated =
fVelocityInterpolate(Projectile.Mass,...
                velocity_impact_estimate);
            fprintf('\nEstimating impact velocity with captured wad');
            fprintf('\nMeasured wad velocity
(ft/s):\t%.2f',velocity_measured);
            fprintf('\nCalibrated projectile velocity
(ft/s):\t%.2f',velocity_calibrated);
```

```
            tempVerror = (velocity_measured - velocity_calibrated)/...
                velocity_calibrated;
            if(tempVerror > 0.45)
                velocity_impact = velocity_calibrated;
                fprintf('\nNOTE>> Using calibrated velocity.');
            else
                velocity_impact = velocity_measured;
                fprintf('\nNOTE>> Using captured wad velocity.');
            end
        end
        clearvars tempVelocity tempdV tempVerror tempdt tempVold
tempVnew;
        clearvars dt_obj dz_obj;
    end
    fprintf('\n-----------------------------------------------------');
    if(velocity_impact - velocity_impact_limit < 1e-6)
        fprintf('\nImpact velocity is below expected value [%.2f]
(ft/s).',...
            velocity_impact_limit);
    end
    fprintf('\nVelocity (ft/s): %.2f',velocity_impact);
    fprintf('\nTrigger time [CH1 CH2] (ms):\t%.2f\t%.2f',...
        CHData(1).Data(i,1),CHData(2).Data(j,1));
    fprintf('\nSensor spacing (in):\t%.2f',...
        dimSetupDimensions.Chronograph_dz);
    fprintf('\n-----------------------------------------------------');
    fprintf('\n');
    clearvars i j;
else
    % CH1 and CH2 data not found estimate velocity by calibration curve
    velocity_measured = Inf;
    velocity_calibrated = fVelocityInterpolate(Projectile.Mass,...
                velocity_impact_estimate);
    velocity_impact = velocity_calibrated;
    % Set ID_trigger for reference in mStrainCalcs
    ID_trigger = floor(size(CHData(1).Data,1)/5);
end
```

---

mWriteOutputData.m

```
%
=========================================================================
=
% % Write data to text file
%
=========================================================================
=
if(bWriteOutputFile)
%     FileNameOUT = FileTestCaseFolder;
    FileDataDirOUT =
strcat(FileParentFolder,'\',FileDataDirectoryOUT,...
```

```
        '\',FileTestDateFolder,'\',FileTestCaseFolder,'\');
    FilePathOUT_final = strcat(FileParentFolder,'\',...
        FileDataDirectoryOUT,'\');
    FileNameOUT = strcat(FileDataDirOUT,FileTestCaseFolder);
    if(~exist(FileDataDirOUT,'dir'))
        mkdir(FileDataDirOUT);
    end
    if(~exist(FilePathOUT_final,'dir'))
        mkdir(FilePathOUT_final);
    end
end
% Write data to comma separated value
if(bWriteOutputFile)
    if(bWriteInputFile)
    FileDirOUT = strcat(FileNameOUT,'_INPUT',FileExtension);
    ID_FileOUT = fopen(FileDirOUT,'w');
    DataOUT = zeros(size(CHDataStrain(1).Data,1),2 *
size(CHDataStrain,2));
    DataOUTheader = zeros(1,size(CHDataStrain,2));
    DataOUTheader = string(DataOUTheader);
    formatstring1 = '';
    formatstring2 = '';
%     Format data output
    for i = 1:size(CHDataStrain,2)
        formatstring1 = strcat(formatstring1,'%s\t\t');
        formatstring2 = strcat(formatstring2,'%f\t%f\t');
        DataOUTheader{i} = CHDataStrain(i).Channel;
    end
    formatstring2 = strcat(formatstring2,'\n');
    formatstring1 = strcat(formatstring1,'\n');
    fprintf(ID_FileOUT,formatstring1,DataOUTheader{:});
%     Write data to file
    for i = 1:size(CHDataStrain,2)
        DataOUT(:,2*i-1) = CHDataStrain(i).Data(:,1);
        DataOUT(:,2*i) = CHDataStrain(i).Data(:,2);
    end
    for i = 1:size(DataOUT,1)
        fprintf(ID_FileOUT,formatstring2,DataOUT(i,:));
    end
    fclose(ID_FileOUT);
    clearvars formatstring1 formatstring2 ID_FileOUT DataOUT i;
    end
%     ------------------------------------------------------------------
---
%     Output Channel Strain Data
    if(bWriteCHStrain)
    FileDirOUT = strcat(FileNameOUT,'_Strain',FileExtension);
    ID_FileOUT = fopen(FileDirOUT,'w');
    DataOUT = zeros(size(Strain(1).Data,1),2 * size(Strain,2));
    formatstring1 = '';
    formatstring2 = '';
%     Format data output
    for i = 1:size(Strain,2)
        formatstring1 = strcat(formatstring1,'%s\t\t');
        formatstring2 = strcat(formatstring2,'%f\t%f\t');
        DataOUTheader{i} = Strain(i).Channel;
```

```
    end
    formatstring2 = strcat(formatstring2,'\n');
    formatstring1 = strcat(formatstring1,'\n');
    fprintf(ID_FileOUT,formatstring1,DataOUTheader{:});
%     Write data to file
    for i = 1:size(Strain,2)
        DataOUT(:,2*i-1) = Strain(i).Data(:,1);
        DataOUT(:,2*i) = Strain(i).Data(:,2);
    end
    for i = 1:size(DataOUT,1)
        fprintf(ID_FileOUT,formatstring2,DataOUT(i,:));
    end
    fclose(ID_FileOUT);
    clearvars formatstring1 formatstring2 ID_FileOUT DataOUT i;
    end
%     ------------------------------------------------------------------
---
    %     Output Principal Strain and Principal Axis and Strain-rate
    if(bWritePrincipalStrain)
    FileDirOUT = strcat(FileNameOUT,...
        '_PrimaryStrains',FileExtension);
    ID_FileOUT = fopen(FileDirOUT,'w');
    DataOUT = zeros(size(PrimaryStrain,1),6);
    formatstring1 = '%s\t%s\t%s\t%s\t%s\t%s\n';
    formatstring2 = '%f\t%f\t%f\t%f\t%f\t%f\n';
%     Format data output
    DataOUTheader = {"Time (s)","Strain_xx","Strain_yy","Strain_xy",...
        "Strain-rate_1","Strain-rate_2"};
    fprintf(ID_FileOUT,formatstring1,DataOUTheader{:});
%     Write data to file
    DataOUT(:,1) = Strain(1).Data(:,1);
    DataOUT(:,2) = PrimaryStrain(:,1);
    DataOUT(:,3) = PrimaryStrain(:,2);
%     DataOUT(:,4) = PrincipalTheta;
    DataOUT(:,4) = StrainShear(:,1);
    DataOUT(:,5) = PrimaryStrainRate(:,1);
    DataOUT(:,6) = PrimaryStrainRate(:,2);
    for i = 1:size(DataOUT,1)
        fprintf(ID_FileOUT,formatstring2,DataOUT(i,:));
    end
    fclose(ID_FileOUT);
    clearvars formatstring1 formatstring2 ID_FileOUT DataOUT i;
    end
    % ------------------------------------------------------------------
---
    % Output consolidated data
    if(bWriteImpactEnergy)
        FileDirOUT = strcat('ImpactData',...
            '_OUTPUT',FileExtension);
        FileDirOUT = strcat(FilePathOUT_final,FileDirOUT);
        ID_FileOUT = fopen(FileDirOUT,'w');
        % Print data header
        % Test Case
        % Initialize header
        HeaderOUT = ["Case"];
        % Initialize impact analysis arrays
```

```
DataOUThdr(1,:) = ["Number of projectiles"];
DataOUThdr(2,:) = ["Impact energy (J)"];
DataOUThdr(3,:) = ["Dimensionless impact energy"];
DataOUThdr(4,:) = ["Measured velocity (m/s)"];
DataOUThdr(5,:) = ["Calculated velocity (m/s)"];
DataOUThdr(6,:) = ["Projectile mass (kg)"];
DataOUThdr(7,:) = ["Cumulative projectile mass (kg)"];
DataOUThdr(8,:) = ["Penetration Estimate (m)"];
DataOUThdr(9,:) = ["Penetration Derived (m)"];
DataOUThdr(10,:) = ["Penetration Actual (m)"];
DataOUThdr(11,:) = ["Volume loss (10^-3 m^3)"];
DataOUThdr(12,:) = ["Peak rear-face stress (Pa)"];
DataOUThdr(13,:) = ["Peak rear-face strain (1e-6)"];
DataOUThdr(14,:) = ["Peak rear-face strain-rate (s^-1)"];
DataOUThdr(15,:) = ["Safety Factor - Penetration"];
DataOUThdr(16,:) = ["Safety Factor - Rear-face cracking"];
DataOUThdr(17,:) = ["Safety Factor - Rear-face spalling"];
DataOUThdr(18,:) = ["Z_p/Z_t"];
DataOUT = zeros(size(DataOUThdr,1),1);
% Initialize material property reference variables
Target_HEL = Medium.HugoniotElasticLimit;   % (Pa)
Target_Ftdy = Medium.TensileYieldDynamic;   % (Pa)
% Initialize format strings
tempformatstr1 = '%s';
tempformatstr2 = '%s';
for i = 1:size(ImpactEnergy,2)
    % Define header
    tempformatstr1 = strcat(tempformatstr1,'\t%s');
    HeaderOUT = [HeaderOUT ImpactEnergy(i).Case];
    % Define output data
    tempformatstr2 = strcat(tempformatstr2,'\t%.3f');
    DataOUT(1,i) = ImpactEnergy(i).nProjectiles;
    DataOUT(2,i) = ImpactEnergy(i).ImpactEnergy;
    DataOUT(3,i) = ImpactEnergy(i).E_dimensionless;
    DataOUT(4,i) = ImpactEnergy(i).V_measured;
    DataOUT(5,i) = ImpactEnergy(i).V_actual;
    DataOUT(6,i) = (2*ImpactEnergy(i).ImpactEnergy/...
        ImpactEnergy(i).V_actual^2);
    DataOUT(7,i) = ImpactEnergy(i).nProjectiles * ...
        (2*ImpactEnergy(i).ImpactEnergy/...
        ImpactEnergy(i).V_actual^2);
    DataOUT(8,i) = ImpactEnergy(i).PenetrationEst;
    DataOUT(9,i) = ImpactEnergy(i).PenetrationVolume;
    DataOUT(10,i) = ImpactEnergy(i).PenetrationActual;
    DataOUT(11,i) = ImpactEnergy(i).VolumeLoss*1e3;
    DataOUT(12,i) = ImpactEnergy(i).StressMax;
    DataOUT(13,i) = ImpactEnergy(i).StrainMax*1e6;
    DataOUT(14,i) = ImpactEnergy(i).StrainRateMax;
    DataOUT(15,i) = ImpactEnergy(i).SF_Penetration;
    DataOUT(16,i) = ImpactEnergy(i).SF_Cracking;
    DataOUT(17,i) = ImpactEnergy(i).SF_Spalling;
    DataOUT(18,i) = ImpactEnergy(i).SF_Zt;
end
clear i;
tempformatstr1 = strcat(tempformatstr1,'\n');
tempformatstr2 = strcat(tempformatstr2,'\n');
```

```
        formatstring = cell(2,1);
        formatstring{1} = tempformatstr1;
        formatstring{2} = tempformatstr2;
        clearvars tempformatstr1 tempformatstr2 i;
        fprintf(ID_FileOUT,formatstring{1},HeaderOUT);
        % Write data to file
        for i = 1:size(DataOUT,1)
            fprintf(ID_FileOUT,formatstring{2},...
                DataOUThdr(i,1),DataOUT(i,:));
        end
        % Close file
        fclose(ID_FileOUT);
        clearvars formatstring ID_FileOUT DataOUThdr DataOUT i;
    end
end
```

BIBLIOGRAPHY

30 C.F.R. §75.335. (2019). Seal strengths, design applications, and installation. Retrieved from https://www.ecfr.gov/cgi-bin/text-idx?SID=c4e7ec6ea4f053b5a15d1335b70d41aa&mc=true&node=se30.1.75_1335&rgn=div8

Atou, T., Sano, Y., Katayama, M., & Hayashi, S. (2014). Damage Evaluation of Reinforced Concrete Columns by Hypervelocity Impact. *Procedia Engineering*, 348-354.

Boresi, A. P., & Schmidt, R. J. (2003). *Advanced Mechanics of Materials.* John Wiley & Sons.

Cadoni, E., Albertini, C., Labibes, K., & Solomos, G. (2001). Behavior of plain concrete subjected to tensile loading at high strain-rate. *Fracture Mechanics of Concrete Structures*.

Carlucci, D. E., & Jacobson, S. S. (2008). *Ballistics: Theory and Design of Guns and Ammunition.* CRC Press.

Cengel, Y. A., & Cimbala, J. M. (2006). *Fluid Mechanics, Fundamentals and Applications.* McGraw Hill.

Clifton, J. R. (1982). *Penetration Resistance of Concrete - A Review.* Washington D.C.: National Bureau of Standards.

Cook, R. D., Malkus, D. S., Plesha, M. E., & Witt, R. J. (1974). *Concepts and Applications of Finite Element Analysis.* Madison: John Wiley and Sons.

Cooper, P. W. (1996). *Explosives Engineering.* Wiley-VCH.

Department of Defense. (2008). *Unified Facilities Criteria 3-340-02, Structures to Resist the Effects of Accidental Explosions.* United States Department of Defense.

Dowding, C. H. (1985). *Blast Vibration Monitoring and Control.* Prentice Hall.

Gamezo, V. N., Zipf, R. K., Kessler, D. A., & Oran, E. S. (2013). DDT in Natural Gas-Air Mixtures on Large Scales: Experiments and Simulations. *24th ICDERS.* Taipei.

Gillies, A. D., & Jackson, S. (1998). Some investigations into the explosibility of mine dust laden atmospheres. *Coal Operators' Conference.*

Goldsmith, W. (1960). *Impact: The Theory and Physical Behaviour of Colliding Solids.* London: Dover.

Grady, D. (1996). *Shock Equation of State Properties of Concrete.* Sandia National Laboratories.

Graff, K. F. (1975). *Wave Motion in Elastic Solids.* Dover.

Grassl, P., Xenos, D., Nyström, U., Rempling, R., & Gylltoft, K. (2013). CDPM2: A damage-plasticity approach to modelling the failure of concrete. *International Journal of Solids and Structures*.

Grassl, P., Xenos, D., Nyström, U., Rempling, R., & Gylltoft, K. (2013). CDPM2: A damage-plasticity approach to modelling the failure of concrete. *International Journal of Solids and Structures*.

Humphrey, H. B. (1960). *Historical Summary of Coal-Mine Explosions in the United States, 1810-1958.* United States Bureau of Mines.

Kennedy, R. P. (1976). A Review of Procedures for the Analysis and Design of Concrete Structures to Resist Missile Impact Effects. *Nuclear Engineering Design, 37*, 183-203.

Kinney, G. F., & Graham, K. J. (1985). *Explosive Shocks in Air.* Springer.

Kolsky, H. (1963). *Stress Waves in Solids.* Dover.

Kong, X. Z., Wu, H., Fang, Q., Zhang, W., & Xiao, Y. K. (2017). Projectile penetration into mortar targets with a broad range of striking velocities: Test and analyses. *International Journal of Impact Engineering*, 18-29.

Krauthammer, T. (2008). *Modern Protective Structures.* CRC Press.

Linz, P. D., Fan, S. C., & Lee, C. K. (2016). Modeling of Combined Impact and Blast Loading on Reinforced Concrete Slabs. *Latin American Journal of Solids and Structures*.

Marchand, K. A., Vargas, M. M., & Nixon, J. D. (1992). *The Synergistic Effects of Combined Blast and Fragment Loadings.* Air Force Engineering and Services Center.

Marsh, S. P. (1999). *LASL Shock Hugoniot Data.* University of California.

Meyers, M. A. (1994). *Dynamic Behavior of Materials.* John Wiley & Sons.

Murthy, A., Palani, G. S., & Iyer, N. R. (2010). Impact Analysis of Concrete Structural Components. *Defence Science Journal*, 307-319.

NDRC. (1946). *Effects of Impact and Explosion.* Washington D.C.

PCB Piezotronics. (n.d.). *Model 740B02 Installation and Operating Manual.*

Rao, S. S. (2011). *Mechanical Vibrations.* Prentice Hall.

Richmond, J. K., Price, G. C., Sapko, M. J., & Kawenski, E. M. (1983). *Historical Summary of Coal Mine Explosions in the United States, 1959-81*. United States Bureau of Mines.

Rouquand, A., Pontiroli, C., & Canton, E. (1996). An explicit damage model for dynamic concrete behaviour. Numerical simulations and comparisons with experimental results on reinforced concrete plates under blast loading. *Transactions on the Built Environment*.

Zabetakis, M. G. (1965). *Flammability Characteristics of Combustible Gases and Vapors.* United States Bureau of Mines.

Zipf, R. K., Gamezo, V. N., Sapko, M. J., Marchewka, W. P., Mohamed, K. M., Oran, E. S., . . . Sellers, D. D. (2010). Methane-Air Detonation Experiments at NIOSH Lake Lynn Laboratory. *Eigth International Symposium on Hazards, Prevention, and Mitigation of Industrial Explosions.* Yokohama: NIOSH.

Zipf, R. K., Gamezo, V. N., Sapko, M. J., Marchewka, W. P., Mohamed, K. M., Oran, E. S., . . . Sellers, D. D. (2011). Preliminary Large-scale DDT Experiments at NIOSH Lake Lynn Laboratory. *International Colloquium on the Dynamics of Explosions and Reactive Systems.*

Zipf, R. K., Sapko, M. J., & Brune, J. F. (2007). *Explosion Pressure Design Criteria for New Seals in U.S. Coal Mines, Information Circular 9500.* NIOSH.

Zukas, J. A. (1980). *Impact Dynamics: Theory and Experiment.* US Army Armament Research and Development.

Zukas, J. A. (2004). *Introduction to Hydrocode.* Elsevier.

VITA

Bruce Albert von Niederhäusern graduated from Palm Bay High School in Melbourne, FL in 2006. He then was enrolled at Utah State University in Logan, UT and in 2007 he took a 2 year hiatus to serve an evangelical service mission in the United Kingdom. After his return, he completed his B.S. in Mechanical Engineering at Utah State University in 2013. In 2016, Bruce enrolled at Missouri University of Science and Technology at Rolla, MO to pursue his Master's degree. In December 2019, Bruce received his M.S. in Explosives Engineering from Missouri University of Science and Technology.