A Parametric Investigation and Optimization of a Cylindrical Explosive Charge

Logan Ellsworth Beaver
Marquette University

A PARAMETRIC INVESTIGATION AND OPTIMIZATION OF A CYLINDRICAL EXPLOSIVE CHARGE

by

Logan Ellsworth Beaver

A Thesis Submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Mechanical Engineering

Milwaukee, Wisconsin

August 2017

ABSTRACT
A PARAMETRIC INVESTIGATION AND OPTIMIZATION OF
A CYLINDRICAL EXPLOSIVE CHARGE

Logan Ellsworth Beaver

Marquette University, 2017

Explosive device design has a wide impact in the space, manufacturing, military, and mining industries. As a step toward computer assisted design of explosives, an optimization framework was developed using the Design Analysis Kit for Optimization and Terrascale Applications (Dakota). This software was coupled with the hydrocode CTH. This framework was applied to three exploding cylinder models, two in 1D and one in 2D.

Gradient descent, dividing rectangles, and a genetic algorithm were each applied to the one-dimensional models. Parametric studies were performed as a basis for comparison with the optimization algorithms, as well as qualifying the 1D model's accuracy. The gradient descent algorithm performed the best, when it converged on the optimum. Dividing rectangles took approximately twice as many iterations to converge as gradient descent, and the genetic algorithm performed marginally better than a full parametric study.

ACKNOWLEDGMENTS

Logan Ellsworth Beaver

I would like to acknowledge and thank the many people that have helped me along the way toward earning my degree. Especially my advisor, Dr. John Borg, for his patience and guidance. Without him none of this would have been possible.

A huge thank you to my friends in the Shock Physics lab, and the rest of the Mechanical Engineering department, for creating lifelong friendships and memories over these past two years.

Thank you Geremy Kleiser and the Department of Defense Air Force Research Laboratory for their invaluable support with this project.

And of course, a special thank you to all of my friends and family outside of Marquette. Your support has been, and will continue to be, invaluable on this journey through life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

1. INTRODUCTION

1.1 Motivation and Scope

The design of explosive devices has a wide reaching impact in mining, demolition, space, manufacturing, and military applications. Traditionally the design of devices using explosives is an expensive and time consuming process involving: extensive machining, building experimental setups, and going to great lengths to ensure the safety of everyone involved. These material and labor costs can be significantly reduced by using computational tools to assist in the design of an explosive.

By coupling a physics based simulation package with an optimization toolkit it is possible to automate most of the design process for explosive devices. In this thesis an automated optimization framework is developed by coupling the Design Analysis Kit for Optimization and Terascale Applications[1] (DAKOTA) with CTH[2], a state of the art hydrocode.

As a step toward computer assisted explosives design, the classic exploding cylinder problem was analyzed and run through several optimization algorithms. The software framework was deployed to perform optimizations and parametric studies across the design space. The framework's extensibility was also demonstrated by applying it to a related design problem.

The framework was applied to maximize the kinetic energy output of an exploding cylinder. Specifically, the kinetic energy of the metal case was maximized by optimizing the cylinder geometry. A dimensional analysis was also performed to visualize the kinetic energy density of the explosive.

1.2 Literature Review

Explosives have a rich history, reaching back from the alchemical origin of gunpowder in the late middle ages[3] to the first modern explosives developed in the late 1800s[4]. Nitroglycerin, the first practical explosive, was synthesized in 1847 by the Italian chemist Ascanio Sobrero[5]. Unlike previous explosives at the time Nytroglycerin detonates rather than deflagrating. When an explosive substance ignites there is an oxidation reaction which produces hot gaseous byproducts. If the heat generated from this reaction cannot be conducted away faster than it is generated a runaway burn ensues. For deflagrating materials, such as gunpowder, this burn front advances slower than the speed of sound in the material. In the case of a high explosive, such as TNT, this reaction will accelerate past the speed of sound in the material and form a steady shock wave[6]. This detonation shock wave releases energy at an extremely fast rate over a very short period of time, resulting in a much higher intensity explosion than a deflagrating material[7]. Most high explosives detonate at a speed near 8 kilometers per second[6], whereas deflagrating wavefronts generally move slower than 1.3 kilometers per second[8].

The standard explosive Trinitrotoluene (TNT) was first synthesized by the German chemist Julius Wilbrand in 1863[9], and it would be another 28 years before another German chemist, Carl Häussermann, fully understood the usefulness of TNT as an industrial scale explosive[10]. At the same time Alfred Nobel was ushering in the age of modern explosives with the invention of modern blasting caps in 1863[11], dynamite[12], and gelignite, the first plastic explosive, in 1875[13]. Plastic explosives are manufactured by mixing a detonating compound with binding agents and filler to produce a significantly more stable compound. Some plastic explosives are so stable that they can be machined, extruded, and even re-cast into different shapes without detonating[6].

The first large scale use of modern explosives in military applications occurred during World War 1. The modern hand grenade was patented by Leon Roland in 1915[14] and improved upon by William Millis in 1916[15]. Additionally, the practice of strategic bombing began when a German zeppelin dropped eight shrapnel bombs on the Belgian city of Antwerp in 1914[16]. By the end of the war both sides understood the devastating potential of explosives in warfare, and a host of sophisticated bombs and explosives were developed by both sides. This ballistic arms race continued on into the 1940s.

At the height of the second world war bombs with interesting mechanical designs were being developed to maximize the damage to personnel and infrastructure. Many novel designs were developed with unique aerodynamic properties, such as the British bouncing bomb and the German butterfly bomb[17]. This was also the time when G.I. Taylor, R. Gurney, and N.F. Mott published their seminal works analyzing explosives.

In Taylor's 1941 paper[18] an analytical model for the transient shape of an exploding cylinder was developed. Taylor used a Lagrangian reference frame traveling with the detonation wave to simplify the analysis. By relating the radial displacement of the case to its position behind the detonation front Taylor was able to calculate the gas pressure and velocity acting on the case along its entire length. This resulted in a set of differential equations describing the shape and velocity of the cylinder during detonation.

Taylor also developed a fracture model for cylindrical explosives in a 1944 paper[19]. Based in part on earlier experimental findings, Taylor noted that as a cylindrical bomb expanded surface cracks would form along the length of the cylinder. At first these shallow cracks would cause a compressive circumferential stress on the inner surface of the shell. As the cylinder continued to expand the cracks deepened as the internal compressive stress decreased. Eventually the cracks

would fully penetrate the outer shell, releasing the high pressure gas. From his analysis, Taylor found that the ratio of the inner radius at failure to the initial inner radius was nearly constant and independent of wall thickness. This ratio ranged from 1.7 for steels to 3.2 for aluminum. This phenomenon was called the "Expansion Ratio" in a 1977 BRL report by Predebon et al[20], which seemed to confirm Taylor's findings.

Meanwhile, in 1943 Gurney authored a report[21] which soon became the fundamental model for the velocity of the exploding cylinder. Gurney noted that the radial velocity of the shell fragments was much greater than their longitudinal velocity. This fact, coupled with the relatively high speed of the detonation wave, meant that the explosion could be effectively modeled as a one dimensional gas expansion. This resulted in a radial energy balance relating the potential energy of the explosive with the kinetic energy of the reactant gases and the kinetic energy of the shell. Solving this energy balance resulted in an expression for the velocity of the shell wall at the point of failure as a function of: initial explosive mass, shell mass, and an empirical energy constant. This constant, sometimes referred to as the Gurney constant, has been experimentally measured for many explosives. The Gurney constant has also been shown to correlate well with the steady state detonation wave speed in many high explosives[6].

Significant advances were made with respect to the fragmenting and fracture of cylindrical explosives by Mott in his 1945 paper[22], where he describes the process of fragmenting as "the tearing apart of a rapidly expanding tube when the material of the tube reaches the limit of its ductility." Due to imperfections in the explosive's outer shell, the strength of the casing will vary along its circumference. At the moment of fracture the weaker points will fail, and a stress release wave will travel outward from that point. The area between the fracture and the relief wave creates a region of low stress where further fracture cannot occur. Using this

approach Mott also derived a stochastic model for fragment length based on the properties of the explosive.

Further analytical models for gas dynamics and fragmentation have been developed based on the works of Taylor, Gurney, and Mott. Several studies[23][24][25] have matched experimental data to the models developed by Gurney fairly well when using cylinders with a length to diameter ratio above 2-3[26]. It has also been observed that the velocity at the ends of the cylinder are significantly lower than the equation derived by Gurney predicts[26] due to edge effects. Several studies have added empirical correction factors to match these edge effects, including equivalent mass equations[26] and velocity correction factors[23][27]. A two dimensional formulation of the Gurney equation has also been proposed[28]. Work[29] has been done to modify the Gurney equation to more accurately match experiments with small shell thicknesses. Attempts have also been made to model the effects of the blast wave on the shell's velocity[30], and functions have been empirically formulated to account for the shock interaction and edge effects of the explosive[31].

Many modern models for the fracture of exploding cylinders have also applied elastic and plastic strain models to the foundational work of Mott and Taylor. Physics-based models[32] have included the effects of plastic work hardening, rate-dependent plastic flow, thermal softening, and void nucleation on the fracture process. Two dimensional material flow in the radial and tangential directions of the cylinder have also been considered in the same work.

With the advent of terrascale computing, and massively parallel processing becoming more widespread, a significant amount of effort has gone towards developing high fidelity computational models of the exploding cylinder problem. A gas leakage model was initially added to the finite difference code HEMP in the mid 70s[33], and as computers have gotten faster more realistic models have been

developed, such as a high fidelity three dimensional model of an exploding cylinder which included air gaps between material layers and a feedback controlled cookoff system[34]. A standardized test setup for exploding cylinders, the Scaled Thermal Explosion Experiment (STEX), was developed at Lawrence Livermore National Laboratories[35] in 2002, leading to a plethora of new experimental data which can be used to verify computational models.

There is not a significant body of work in explosives design with respect to optimization which has relied on experience and intuition in the past. Many analytical models are not general enough to cover the wide design space required by optimization algorithms. The most significant work in explosives optimization actually appears in computer graphics literature. It is desirable to have a very specific shape and motion for explosions in movies and video games, and papers have been published[36] which seek to optimize an initial explosive charge with respect to the shape of the explosion. In general these explosive models are neither rigorous nor realistic, and as such are not applicable to engineering. However, they do give a useful insight to the process of optimizing an explosives system.

2. ANALYTICAL MODELS

2.1 Physical Phenomena

When analyzing the exploding cylinder, many analytical models set up the geometry in such a way that the shock effects are negligible[37], or neglect the effect of the blast wave entirely by considering just the effects of the expanding gas[38]. In general the of energy contribution of the detonation wave is insignificant compared to the gas expansion, and these gas dynamics models are applicable to approximately solve a wide range of problems[30] .

During an explosion the blast wave propagates through the material significantly faster than the speed of sound in the explosive. This results in the shell being affected by the blast wave before it interacts with the expanding gas. This effectively results in two regimes of energy transfer between the explosive and the outer shell[30]. Usually the high pressure gas has a much greater impact on the final velocity than the detonation wave, but thin-walled explosives are significantly affected by the shock dynamics[30].

Two of these fundamental gas-based analytical models for cylinder expansion come from G.I. Taylor and R.W. Gurney. To calculate the profile of an exploding cylinder Taylor used a Lagrangian approach with the conservation equations. Gurney calculated the case velocity at the moment of fracture using a conservation of energy approach.

2.2 Taylor Model

To analyze the shape of an exploding cylinder Taylor made several key assumptions to reduce the problem down to a two dimensional steady-state analysis. He assumed that the chemical reaction in the flame front was instantaneous, the mass of the shell was significantly higher than the mass of the explosive charge, and the cylinder was infinitely long and axisymmetric.

To simplify the gas dynamics calculations a Lagrangian reference frame was fixed to the detonation wave as it traveled down an infinitely long cylinder. Taylor utilized a combination of Bernoulli's equation, Newton's laws, and continuity to derive the final shape and velocity of the case along its length. The velocity of the case is given by

$$V = 2\,U \sin\left(\frac{1}{2}\tan\left(\frac{\mathrm{d}r}{\mathrm{d}x}\right)^{-1}\right) \tag{1}$$

where V is the velocity of the case, U is the velocity of the detonation wave, $\mathrm{d}r$ is the radial displacement of the case, and $\mathrm{d}x$ is the distance behind the flame front. Equation 1 must be solved numerically, and the solutions can be found in [39].

2.3 Gurney Model

The Gurney model of an exploding cylinder uses a conservation of energy approach to relate the chemical potential energy of an explosive to the output kinetic energy[21]. For simplicity Gurney also assumed a linear velocity profile, such that $V_r = V_0 \frac{r}{a}$, where V_r is the radial velocity of the expanding gas, a is the inner radius of the explosive and V_0 is the velocity of the case. Applying conservation of

energy results in the equation

$$EC = \frac{1}{2}MV_0^2 + \frac{1}{2}V_0^2 \int_0^a 2\pi r\rho \frac{r^2}{a^2}\,\mathrm{d}r \tag{2}$$

where E is the initial chemical energy density of the explosive, C is the mass of the charge, M is the case mass, V_0 is the case velocity at the moment of fractures, ρ is the density of the explosive gas, and a is the inner radius of the case.

By using a constant density assumption Equation 2 simplifies to the form

$$\frac{V}{\sqrt{2E}} = \left(\frac{M}{C} + \frac{1}{2}\right)^{-\frac{1}{2}} \tag{3}$$

where C is the initial mass of the explosive charge. The term $\sqrt{2E}$ is sometimes referred to as the Gurney constant and has units of velocity. Equation 3 is convenient for analytical calculations of kinetic energy as it only has three parameters; the charge and shell mass, which can easily be calculated from material properties, and the Gurney constant, which is empirically determined for each explosive.

It has been shown that the Gurney equation overestimates fragment velocity when the aspect ratio, $\frac{L}{D}$, is less than 2 or when the mass ratio, $\frac{M}{C}$, is less than 0.1[26][29]. Several attempts have been made to correct the Gurney equations, such as adding an artificial solid core to the explosive[29], empirically fitting a correction function to velocity measurements[23][31]. Even with these drawbacks the Gurney equation is still a practical tool for developing an initial estimate of the fragment velocity of an explosive[33].

2.4 Expansion Ratios

When modeling the response of an exploding cylinder it is useful to know at what state the cylinder walls will fracture. As a cylindrical explosive expands long cracks develop on the surface in the axial direction[39]. At some points these cracks will penetrate the outer wall, resulting in gas leakage. This leakage reduces the energy available to accelerate the fragments, significantly reducing the radial acceleration of the case and limiting the terminal velocity[33]. Therefore, by increasing the maximum expansion of the case before fracture it is possible to increase the terminal velocity of the fragments[20].

Work has been done by Taylor[39] and Predeborn[20] to help determine the expansion ratios for several metals. The expansion ratio is the radius where gas leakage first occurs divided by the initial radius of the cylinder. The expansion ratios for steel, copper, and aluminum are approximately 1.9, 2.4, and 3.2 respectively. Expansion ratios also appear to be independent of wall thickness in the limited experiments that have been performed.

2.5 Dimensional Analysis

A dimensional analysis approach can be taken to the model exploding cylinder problem using the variables summarized in Table 1.

Table 1: Variables and their dimension used for nondimensionalization of the exploding cylinder problem

Name	Variable	Dimension
Shell Mass	M	Mass
Charge Mass	C	Mass
Outer Radius	R_o	Length
Thickness	t	Length
Final Velocity	V	$\frac{\text{Length}}{\text{Time}}$
Gurney Constant	g	$\frac{\text{Length}}{\text{Time}}$

By combining like variables in a Bukingham Pi analysis it is possible to reproduce the dimensionless groups in Equation 3. It is also possible to group terms based on the kinetic energy of the case and potential energy of the explosive, resulting in two nondimensional groups. The kinetic energy of the case is $\frac{1}{2}MV^2$, leaving the terms C, g, t, and R_o. An analogue to potential energy can be created with the term $\frac{1}{2}Cg^2$, and the remaining terms can be grouped into the geometric variable $\frac{t}{R_o}$. These nondimensional groups results in the function

$$\frac{MV^2}{Cg^2} = f\Big(\frac{t}{R_o}\Big) \tag{4}$$

with each variable coming from Table 1. Although Equation 4 does not give the explicit functionality of each term, it may give a useful insight on how to flatten higher dimension simulation results, experimental data, and Equation 3 to two variables. This flattening can be seen in Figure 1 using the gurney equation with several geometries for an aluminum case filled with TNT.

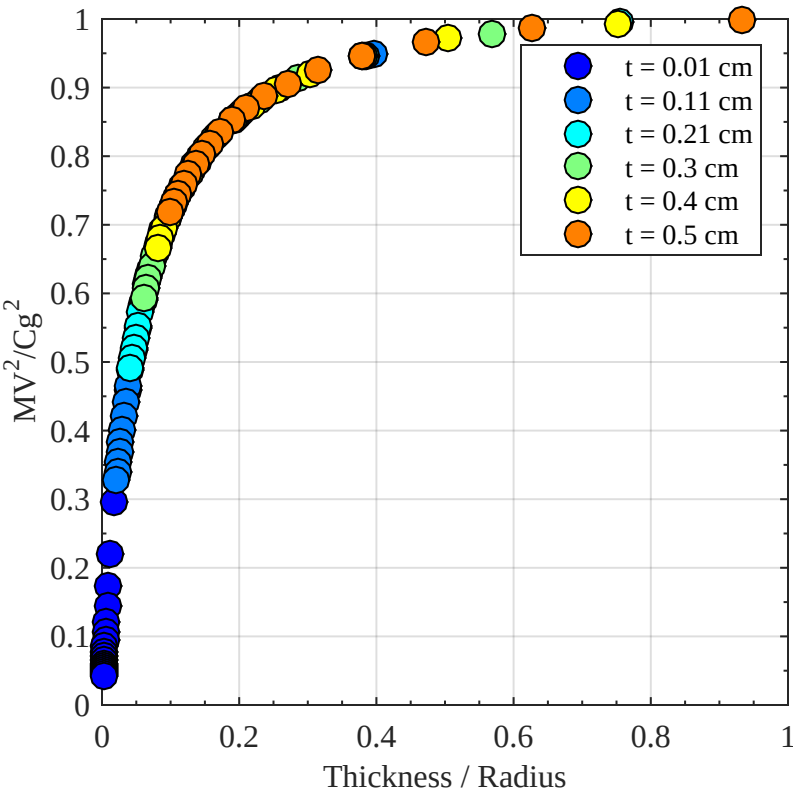

Figure 1: The Gurney equation nondimensionalized to fit the form of Equation 4.

As the wall thickness of the cylinder approaches zero the wall mass and kinetic energy also go to zero, as seen in Figure 1. It is also reasonable that the wall thickness approaches the outer radius, the mass of explosive should go to zero. Intuitively, a very thick cylinder should have a negligible kinetic energy and a nonzero potential energy. It would be expected then that as $\frac{t}{R_o} \to 1$, $\frac{MV^2}{Cg^2} \to 0$. However, applying this limit to Equation 3 using L'hopital's rule results in a value of 1, as seen in Figure 1. This nondimensionalization scheme applied to the CTH parametric study results in Chapter 4.

2.6 Blast Wave and Shock Effects

Modern approaches for modeling shock and detonation waves use sophisticated equation of state (EOS) tables and analytical models, such as Jones-Wilkins-Lee or Mie-Grüneissen. These approaches are based on models developed by W.J.M. Rankine and P.H. Hugoniot in the late 1800s[40] of how shock waves travel through materials. By applying the fundamental conservation equations of mass, momentum, and energy the Rankine-Hugoniot jump condition was derived.

From experimental data, the relationship between the velocity of a shock wave, U_s, and the velocity of the particles behind the shock, U_p, is approximately linear in many materials[41] with the form

$$U_s = sU_p + c_0 \tag{5}$$

where c_0 is the bulk longitudinal sound speed of the material and s is the empirical Hugoniot constant. Equation 5 can be combined with the conservation equations to obtain a relationship between pressure and volume with the form[6]

$$p_2 = p_1 + \frac{c_0^2(v_1 - v_2)}{(v_1 - s(v_1 - v_2))^2} \tag{6}$$

where 1 is the state of the unshocked material and 2 is the state of the material just behind the shock wave, p_i is the pressure at each state, and v_i is the specific volume at each state. It should be noted that the Hugoniot of a material represents the discontinuous jump between the initial and final state of the material. The thermodynamic path between states can be modeled as a straight line, known as the Rayleigh Line, or an isentrope[6].

In a detonation the blast wave induces a particle velocity in the case as it moves along the cylinder. This wave can ring between the inner and outer surfaces

several times; this behavior is based on the shock impedance of each material. Eventually the hot expanding gas catches up to the case, supplying it with additional kinetic energy until the case bursts and the gas can escape[30]. By analyzing the detonation wave it is possible to include the energy from the shock wave into existing models, as well as allowing for a physics-based derivation for the state of the expanding gas as it drives the outer shell.

An idealized detonation consists of unburned explosive, a flame front, and high pressure gas products[6][7]. As the explosion proceeds the flame front reaches a steady shock speed, known as the detonation velocity. This detonation velocity is relatively constant, but it can decrease when the outer radius of an explosive is very small[31]. By applying conservation of mass and momentum to the flame front it is possible to derive the Chapman–Jouguet (C-J) Condition which predicts the initial state of the reactants behind the detonation wave[7].

The chemical transition between the unshocked solid explosive and the reactant gases can not be easily modeled with the Hugoniot equations. To account for this the Jones-Wilkins-Lee (JWL) EOS was developed to model the shock response of the reactant gases[42]. The JWL EOS is an empirical model which can be combined with the Hugoniot jump condition to approximately model the detonation. Two interesting relationships can be observed by plotting the JWL EOS fo the product gases and the Hugoniot of a solid explosive in P-v space.

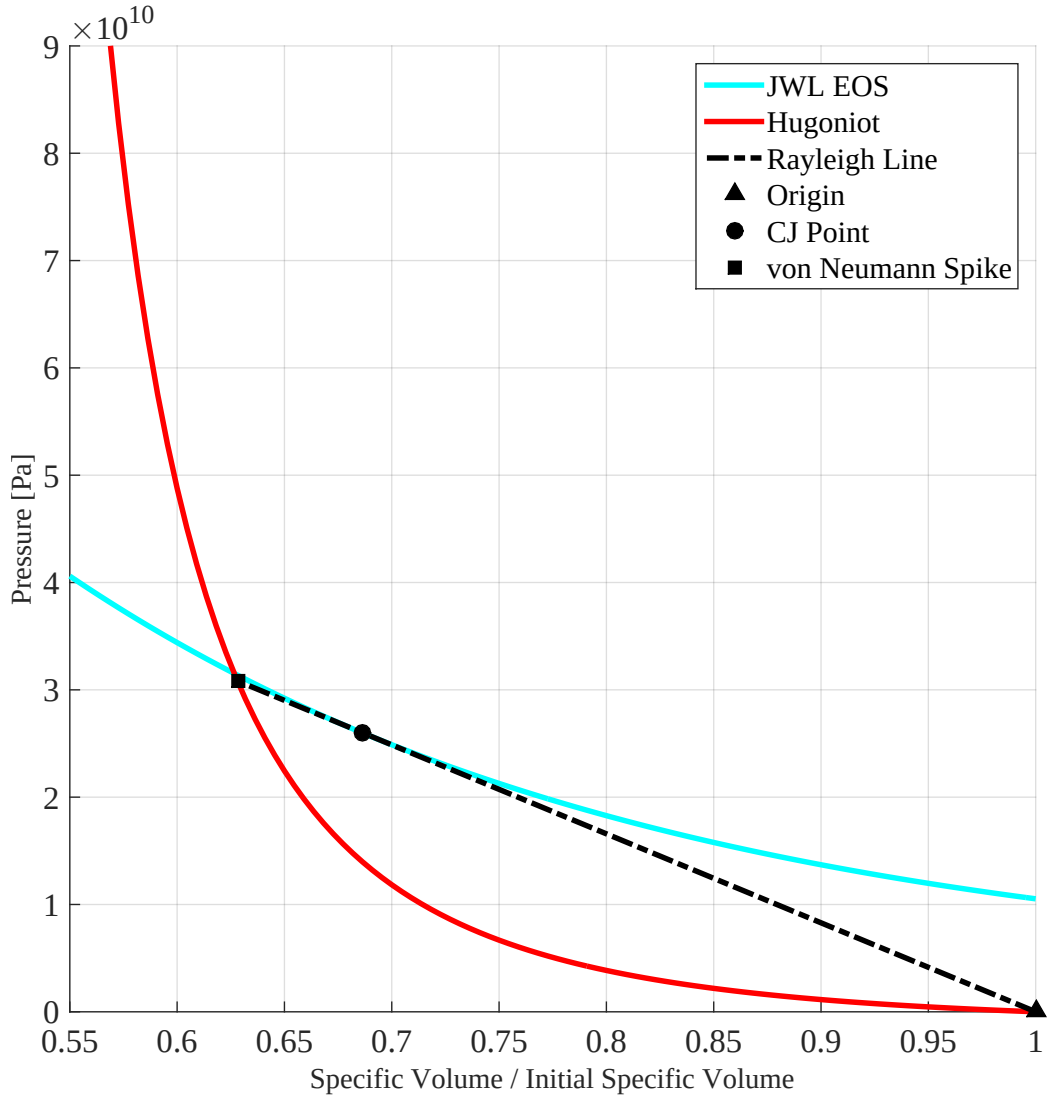

Figure 2: The Hugoniot of TNT plotted alongside the JWL equation of state for TNT. The Rayleigh line, initial state, C-J state, and von Neumann spike are also shown.

The Rayleigh line in Figure 2 starts at the initial state of the solid, is tangent to the JWL EOS curve, and passes through the Hugoniot of the solid TNT at a higher pressure. The tangent point between the Rayleigh line and the JWL EOS coincides with the C-J state of the explosive while the upper intersection of the Rayleigh line with the Hugoniot represents the peak pressure (known as the von Neumann Spike) in the flame front[7]. It is possible for the Rayleigh line to pass through the JWL curve, resulting in either a strong shock, when the pressure is

higher than the C-J pressure, or a weak shock, when the pressure is lower than the C-J pressure. However, strong and weak shocks tend to be unstable, and are usually transients that converge to the C-J point[7]. A qualitative pressure versus length diagram is given in Figure 3 for the values in Figure 2 to show the pressure profile in the axial direction.

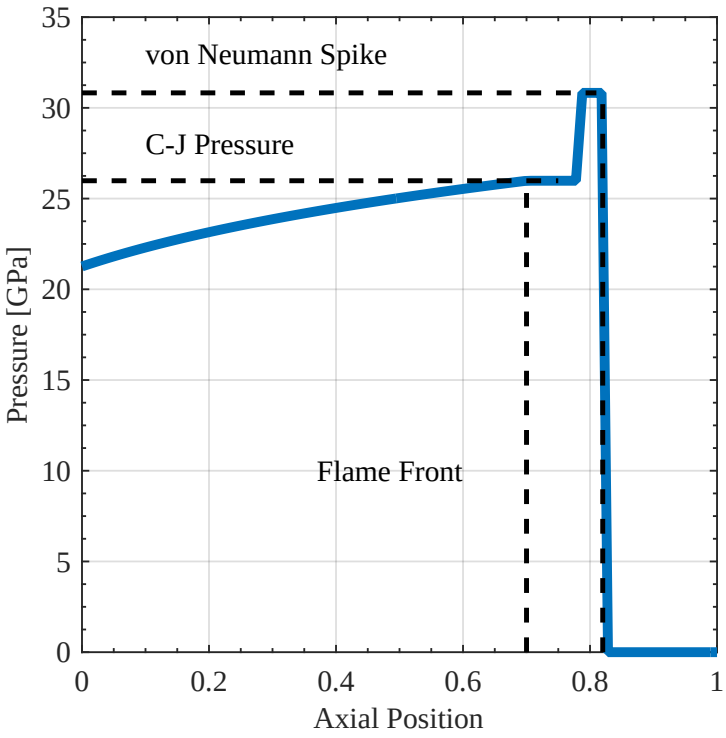

Figure 3: An example of a pressure versus position graph for the explosive in Figure 2. From right to left, the pressure starts at 1 atmosphere, reaches to the von Neumann spike around 0.8 units, reaches the C-J pressure at 0.7 units, and then decays due to Taylor expansion of the case.

The properties used to generate Figures 2 and 3 are given in Table 2.

Table 2: TNT constants used to generate the JWL EOS and Hugoniot curves in Figure 2[6].

EOS	v_0 [m^3/kg]	p_0 [Pa]	c_0 [m/s]	s	Chemical Energy [GPa]
JWL	6.105 10^{-4}	101300	2140	1.88	6.00
EOS	A $[GPa]$	B $[GPa]$	$R1$	$R2$	w
Hugoniot	373.8	3.747	4.15	0.90	0.35

By understanding the underlying physical mechanisms of detonation waves it is possible to build more accurate computational and analytical models of explosives. In current state of the art hydrocodes the flame front is handled as an interpolation between the initial and final states of the explosive. This simplifies the detonation processing cost while producing an accurate initial condition for the reactant gases[43]. These approaches are used by CTH to implement accurate detonations, and this approach will be compared to the analytical models developed by Taylor and Gurney.

2.7 Rarefaction Waves and Edge Effects

Many cylindrical explosives are detonated at a point on one end, resulting in a wave that travels along its length. Initially this wave expands spherically until the inner radius of the cylinder is reached. At this point part of the detonation wave pressure is reflected back in to the product gases, and part of it is transmitted into the wall of the cylinder. The wave in the cylinder wall continues outward, eventually reaching the air-case interface. Air's impedance is significantly lower

than any metal, so a majority of the energy in the wave is transmitted into the air as a shock wave. The wave energy is further reduced when the metal casing expands, resulting in a net outward velocity in the case. This process releases most of the pressure in the case, resulting in a rarefaction, or release, wave.

A rarefaction wave reduces the pressure of any material it passes through. Due to their low pressure rarefaction waves travel at the local sound speed of the material, and they are reflected and transmitted through interfaces like any other wave. This can result in a highly complex wave interactions in heterogeneous materials, with several reflected high pressure and release waves interacting at different points. These interactions are the main cause of edge and end effects in an explosive, and can be seen in Figure 4.

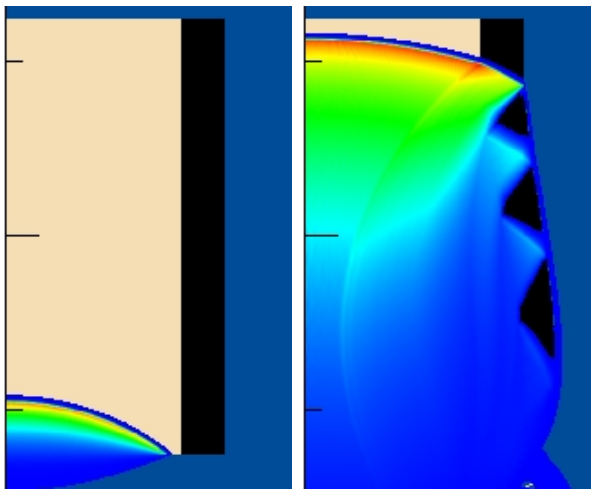

Figure 4: A velocity contour plot of the exploding cylinder before and after the detonation wave passes through the material. Blue is air, black is aluminum, and tan is TNT.

The release waves can be seen in the right image of Figure 4. The sawtooth-like pattern is caused by the interaction of release waves and the reflected detonation front. This ringing leads to oscillations in the case velocity.

The same release waves are generated when the detonation wave reaches either end of the cylinder. In this case the release waves are reflected back in to the

product gases, reducing the pressure and energy available to accelerate the case. This results in a significantly lower velocity near the edges of the explosive. The end of the explosive which was initially detonated generates a rarefaction wave almost immediately. This results in a significantly lower fragment velocity at the detonated end compared to the far end. This asymmetry is seen in experimental data[23] and is shown in Figure 5.

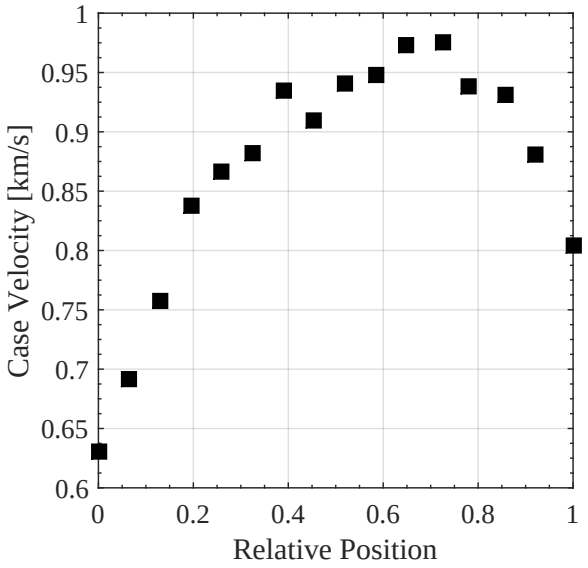

Figure 5: Experimental data from [23] showing the variation of shell velocity with a length to diameter ratio of approximately 1.3.

The velocity near the middle of the explosive approaches the Gurney velocity for sufficiently large values of $\frac{L}{D}$[44]. For this reason the optimizations were performed with respect to an ideal explosive which has no variation in fragment velocity along its length. This can then be transformed into a real velocity profile using empirical shape functions or a physics-based model based on the length to diameter ratio.

2.8 Parameterization and Computational Efficiency

Three parameterizations of the exploding cylinder geometry can be used to completely describe the system: inner and outer radii ($r_o - r_i$ space), outer radius and thickness ($r_o - t$ space), or charge mass and case mass ($C - M$ space). For design purposes, optimizing the kinetic energy of the fragments in terms of case and charge mass isn't especially useful. To determine the geometry of the explosive it is first necessary to convert the charge mass into an equivalent inner radius, then calculate an outer radius using the density of the outer shell. For this reason only the $r_o - r_i$ and $r_o - t$ parameterizations are explored further.

It should be apparent that whenever the mass or velocity of the shell is equal to zero the kinetic energy will also be zero. There are two cases where this can happen; when the case thickness is equal to the outer radius or when the case thickness is zero. Additionally, the velocity term is second order while mass is first order, and the velocity of the shell increases with the mass of the explosive. For this reason the peak kinetic energy should occur as the mass of the case approaches zero. This relationship is shown in Figure 6 using the Gurney equation.

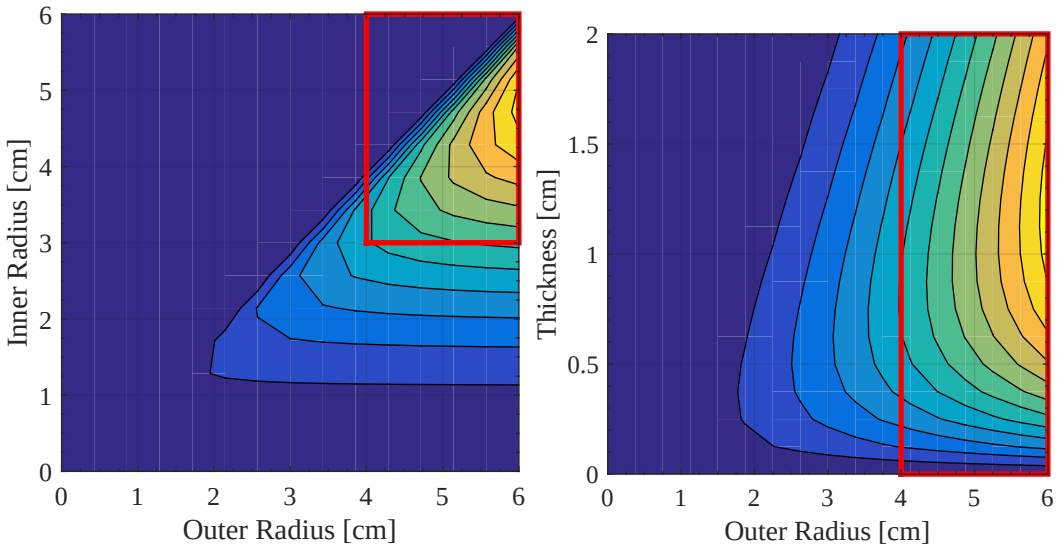

Figure 6: The kinetic energy output of an explosive cylinder problem in terms of inner and outer radius, and outer radius and thickness. Areas of high and low kinetic energy are yellow and red, respectively.

In Figure 6 the regions of high and low kinetic energy are represented by yellow and red respectively. The design space of each variable is bounded by solid lines. To perform an optimization an upper and lower bound must be set on each variable, which generally results in a square domain. When the geometry is described in terms of $r_o - r_i$ the area of interest lies diagonally across the design space, whereas the area of high kinetic energy is parallel to the thickness axis in $r_o - t$ space.

It could be stated that the $r_o - t$ representation is more computationally efficient, that is it takes more function evaluations in $r_o - r_i$ space to get the exact same information. Alternatively, the number of near-zero kinetic energy function evaluations required to explore the entire $r_o - r_i$ space is much higher. For this reason the optimizations were all performed in terms of wall thickness and outer radius rather than inner and outer radius.

3. SIMULATION ENVIRONMENT

3.1 HPC Hardware

The optimization task was run on high performance computers (HPCs) through the Army Research Lab (ARL) and Department of Defense (DoD), specifically on the Excalibur[45] machine. Excalibur has a Cray XC operating system with 3098 compute nodes, each containing 32 Intel Xenon E5-2698 processors and 128 GB of memory. By taking advantage of the massive parallelization of these HPCs it is possible to run high resolution simulations in a reasonable amount of time.

The simulation and input deck files were stored in a working drive. After submitting the job to the PBS queue all of the required files were copied to a 50 petabyte scratch drive to run and post-process each evaluation. The output files were then transferred to a local machine for visualization and analysis.

3.2 Software Configuration

The basic structure for running Dakota with CTH utilizes seven files split between two folders. The main level folder contains items that only need to be used once, namely the Dakota input deck and the queue submission script. There is a template folder inside the main folder which contains the files required to run CTH for each evaluation and extract the kinetic energy. This structure is shown in Figure 7.

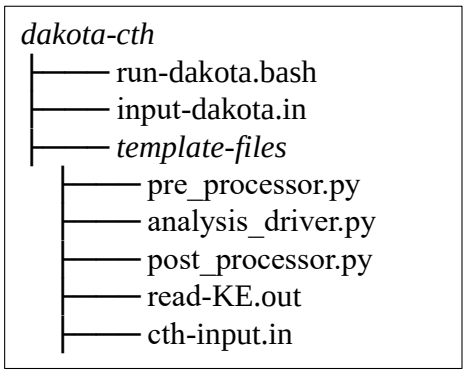

Figure 7: The tree file structure for an optimization run with Dakota coupled to CTH.

For each evaluation of CTH, Dakota copies the template files to a unique folder (such as workdir.10 for the 10th evaluation), and creates a parameter file listing values for each variable. Dakota then runs the preprocessor, analysis driver, and post-processor before writing a return file with the objective function value. When all evaluations have been completed Dakota writes a data file to the top level directory with the results of each evaluation. This process is summarized in Figure 8.

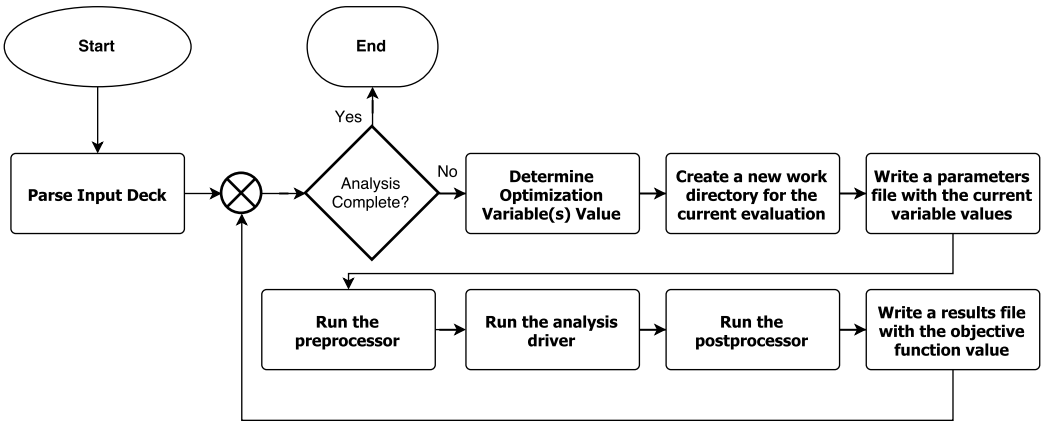

Figure 8: The process flowchart for a Dakota optimization.

It is not necessary for Dakota to run CTH directly. It is possible for Dakota to instead submit jobs asynchronously to the queue and monitor each work directory for the results file. This allows the total processing footprint of the

optimization to vary over time rather than reserving a set number of nodes for the entire optimization task.

After selecting an optimization method, Dakota selects the value of each variable for every iteration until a convergence criteria is met. Some methods, such as gradient descent, require a strict operating order and cannot be easily parallelized. Other methods, such as genetic algorithms, can have many parallel evaluations occurring at once, decreasing the total wall time at the cost of increased processor load.

To run the function evaluations Dakota requires the name of the optimization variables, the number of arguments that will be returned for each objective function evaluation, the command used to run the simulation, and any preprocessing or post-processing commands. The python code and C++ processing code can be found in Appendix III.

In the Dakota's input deck the variables for wall thickness and outer radius are defined, along with their maximum and minimum values. It is convenient to define the geometry in terms of inner and outer radius when building a mesh, so the optimization variables are passed to a preprocessor and the inner radius is calculated by subtracting the thickness from the outer radius. This information is then passed to the analysis driver to build the geometry and run the simulation.

Finally the post-processor parses the kinetic energy from CTH output files and calculates the maximum. This maximum value can then be scaled and returned to Dakota as the objective function value. All Dakota optimizations are minimizations, so it is necessary to negate the kinetic energy returned by CTH. It may also be useful to scale the objective function to a magnitude near 1.

3.3 Computational Model Setup

The simplest possible analysis of an exploding cylinder is a one dimensional model of the system's velocity in the radial direction. By neglecting the axial and circumferential directions it is possible to model the motion of the outer shell up to the point of failure. Additionally, it can be argued that the velocity of the wall at failure is very close to the terminal fragment velocity. With these assumptions it is possible to construct a fairly low cost simulation with reasonable accuracy.

A diagram of the full three dimensional cylindrical and simplified one dimensional geometry is annotated in Figure 9.

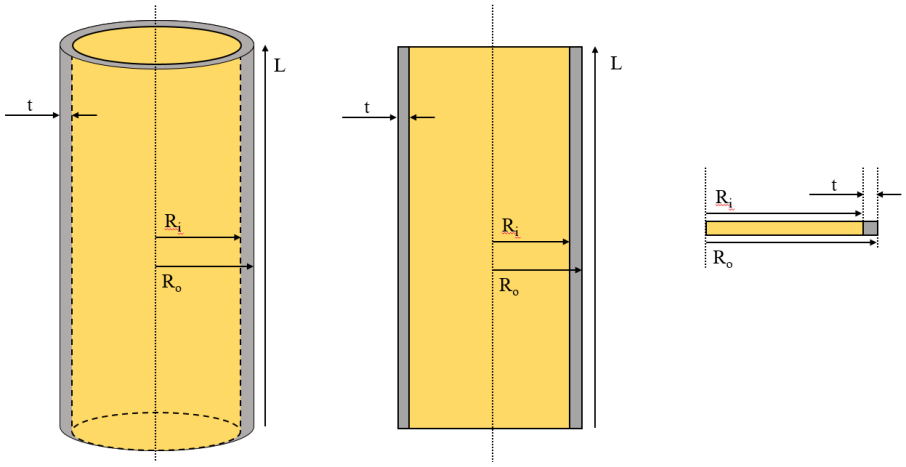

Figure 9: Three dimensional explosive model with two and one dimensional simplifications.

In Figure 9 the measurements are all taken relative to the center axis of the explosive. R_i is the inner radius of the case, R_o is the outer radius of the case, L is the axial length of the explosive, and t is the wall thickness of the case. This configuration was realized in CTH using a 20 cm mesh with a resolution of 0.01 cm/node. The explosive material and wall segment were then placed into the domain based on the values of R_i and R_O.

Three material models are required to run this simulation, one for the explosive, one for the shell, and one for the ambient air. Table 3 shows the material properties and EOS used in the simulation.

Table 3: Initial state for each material in the simulation at STP.

Material	State Model	Density [g/cc]	Sound Speed [km/s]
Aluminum	Sesame	2.6993	5.2097
Air	Sesame	1.218e-3	0.3388
TNT	JWL	1.630	6.934

The purpose of including air in the simulation domain is twofold. First, having air at standard temperature and pressure (STP) outside of the explosive casing equalizes the internal and external pressure. This allows for more reasonable initial pressure and temperature conditions within the explosive device. The second reason is due to an inaccuracy in the numerical solver. CTH is unable to accurately track gas expanding into a vacuum[43], so higher dimensional models require something for the products gases to expand into. Gas was included in the one dimensional model to maintain consistency.

C-J properties for TNT can be found in Table 4.

Table 4: C-J properties used with the JWL TNT model

Material	Detonation Velocity[cm/s]	C-J Pressure [GPa]	C-J Temperature [K]
TNT	693,000	21.19	3801

To propagate the detonation wave CTH uses a conservation of energy equation to interpolate between the initial unburnt explosive and the expanding product gases[43]. This transition is modeled with the equation

$$S(\rho, T, \lambda) = (1 - \lambda)S_i(\rho, T) + \lambda S_f(\rho, T) \tag{7}$$

where S is the state of the material, λ is a factor between 0 and 1 representing the extent of the reaction, ρ is the material density, and T is the material temperature. This allows the explosive's state to transition smoothly between the unexploded material, S_i, and the product gases, S_f. In the case of a pure explosive, such as TNT, the reaction extent, λ, can be calculated using a basic reaction rate model. Equation 7 is easy to calculate, but it does not include some features of the flame front, such as the von Neumann pressure spike or complex multi-step chemical reactions.

The one dimensional CTH domain was set up with the spatial and temporal parameters shown in Table 5.

Table 5: Three environmental variables used to set up the CTH simulation.

Final Time [s]	Min. Time Step [s]	Left Boundary	Right Boundary
$100.0\,10^{-8}$	1.010^{-11}	Reflective	Transmissive

The left boundary was reflective to enforce the axisymmetric behavior of the explosive. The right boundary was set to allow any pressurized ambient air to flow out of the system. In higher dimensional systems the top and bottom boundaries were set to transmissive for the same reason. The full input decks for each CTH simulation can be found in Appendix II.

3.4 Boundary Conditions

There are many boundary conditions (BCs) available in CTH. The most common BCs are reflective, periodic, transmissive, in/outflow, and constant value. A general overview of each boundary condition is given below[43].

The reflective boundary condition causes material to reflect back into the domain at the boundaries. This boundary condition is based on the idea that the current simulation is reflected over each boundary, so any material attempting to flow out comes in to contact with an identical copy of itself flowing in the opposite direction. This reflective interaction is useful for keeping material within the domain while conserving energy and momentum.

A periodic boundary condition must be applied to two opposite boundaries, such as the left and right edges of a square in 2D. Any material that exits over a periodic boundary enters on the corresponding side on the other edge of the domain. This condition conserves energy and momentum of the system, and is useful for modeling a strip of a material with a repeating structure, such as a crystal lattice.

Transmissive boundaries allow material to exit the domain. This can lead to an overall loss of mass, momentum, and energy within the region. Transmission can be useful for simulating only an important sub-region of a full sized problem, for example it is used to allow the product gases to expand out of the domain in the exploding cylinder problem. Transmission is also useful as outlet conditions computational fluid dynamics problems.

The inflow and outflow boundary conditions are achieved by setting the gradient at a boundary to a constant value. The amount of material entering or exiting the domain is dependent on the dot product of the gradient and the normal vectors at the boundary. If this product is positive material is flowing out of the

domain, and if the sign is negative material is flowing in. A gradient of zero is equivalent to the transmissive boundary condition.

Constant value is the simplest boundary condition. After each time step the fixed boundaries are set back to their initial state. This approach can violate conservation if the energy and momentum transferred to the boundary is not redistributed back to the domain at each time step.

3.5 Overview of Optimization Methods

In general terms an optimization problem can be fit into the form [46]

$$\begin{aligned} &\text{minimize: } f_0(x) \\ &\text{subject to: } f_i(x_i) < b_i,\ i = 1, 2, ..., n \end{aligned}$$

where x_i are the optimization variables of the problem, the function $f_0(x)$ is the objective function that maps the vector $x = [x_1, x_2, ..., x_n]$ to a scalar value, and $f_i(x_i)$ are the constraint functions on x_i with respect to the limits b_i. The exploding cylinder problem could be stated as:

$$\begin{aligned} &\text{minimize: } -KE \\ &\text{subject to: } R_o = 5 \text{ cm}, 0 < t < 5 \text{ cm} \end{aligned}$$

where t is the wall thickness, r is the outer radius, and KE is the maximum kinetic energy of the outer shell. The kinetic energy could be calculated with an approximate analytical model, such as those proposed by Gurney[38] and Taylor[37], or the kinetic energy could be derived from simulations.

In Dakota the objective function can be treated as a black box with a fixed number of optimization variables and objective functions. After prescribing an optimization algorithm, the number of inputs and outputs, and the limits for each

variable, it will systematically adjust the variables to reach an optimal solution. It is also possible to perform a parametric study of the objective function through the entire variable domain.

Calculating the variables which result in the maximum kinetic energy output of an explosive is a nonlinear global optimization problem. The problem is nonlinear because the kinetic energy can not be written as a linear combination of the optimization variables; it is a global problem because there may be several combinations of input variables that result in a local kinetic energy maxima. These constraints significantly limit the available optimizers within Dakota's libraries. The three most applicable algorithms within Dakota for this problem are: multi-start gradient descent, dividing rectangles (direct), and genetic algorithms.

Gradient descent is one of the most common and straightforward optimization methods. In general gradient descent is two steps, calculate the direction with the most negative gradient and take a step in that direction[46]. The algorithm will trace the steepest path "downward" and find the most significant local minima. To apply this method to a global optimization problem Dakota uses a multi-start approach[1]. N points are randomly selected within the design space and a gradient descent is performed until a convergence threshold is reached. This method has the benefit of a straightforward implementation, but convergence on the global minimum is not guaranteed and the surface must be smooth and continuous to calculate gradient data.

The direct algorithm is a global gradient-free method with the much weaker requirement of Lipschitz continuity[47]. That is, the function being optimized must have a bounded derivative in each direction, but that derivative does not need to be evaluated. In general any continuous function that doesn't tend to infinity is Lipschitz continuous.

One useful function for testing the robustness of an optimization algorithm is the Rosenbrock Function[48]. The Rosenbrock function is designed in such a way that there is a single global minima, and the gradient around that point is very close to zero. A two dimensional form of the equation can be written as

$$u(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{8}$$

where x_1 and x_2 are optimization variables, and u is the output of the Rosenbrock function. The direct algorithm is used to optimize this function in Figure 10.

The direct algorithm starts by normalizing the optimization variables onto the unit hypercube, i.e. it linearly transforms the range of the variables to [0, 1]. The space is then divided into thirds for each optimization variable, x_i, such that the dimension with the best (smallest valued) evaluation are split first[47]. This step is shown in Figure 10 for a two dimensional form of the Rosenbrock equation.

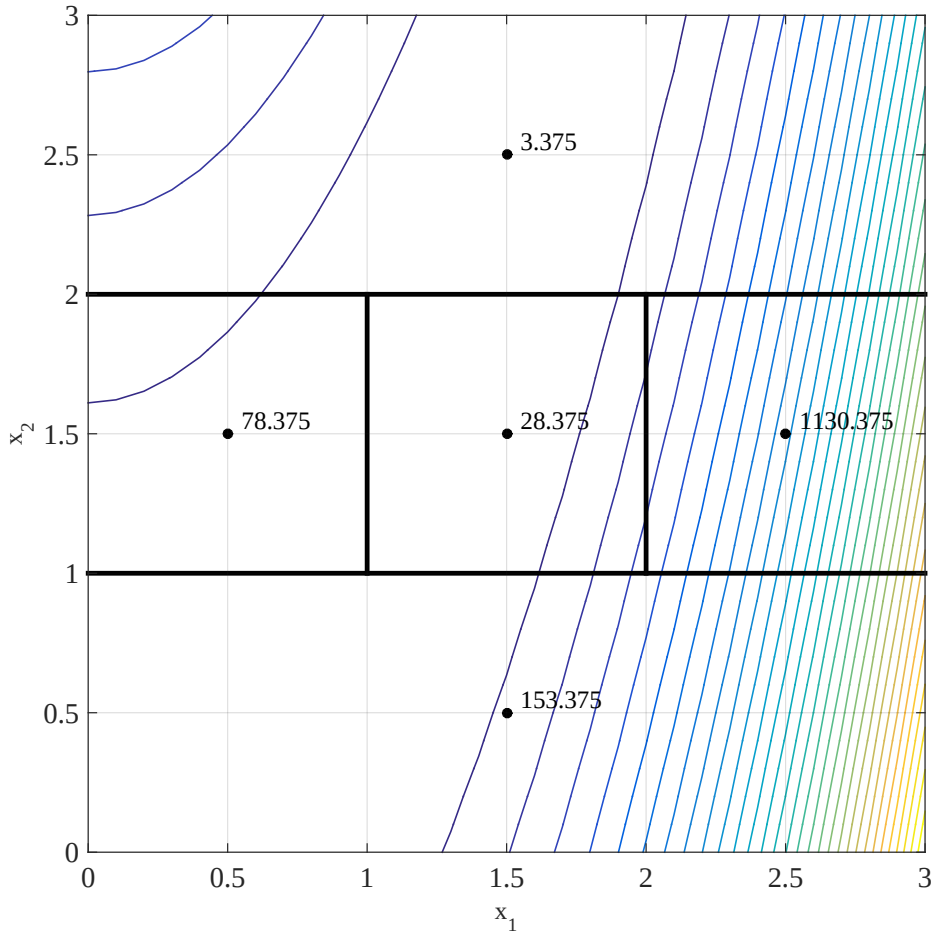

Figure 10: The initial two dimensional setup for a direct optimization. The labeled points show objective function evaluations.

The best function evaluation in Figure 10 is 3.375 in the x_2 direction, so that subspace is divided first. The only remaining subspace is x_1, so it is divided second into smaller partitions. After this initial setup the direct algorithm will systematically identify and subdivide potentially optimal partitions. A partition is considered potentially optimal if it meets the conditions[47]:

$$f(c_j) - \hat{K}d_j <= f(c_i) - \hat{K}d_i \text{ and}$$
$$f(c_j) - \hat{K}d)j <= f_min - \epsilon|f_min|$$

where j references the current partition, i refers to every other partition, c is the center of a partition, $f(c)$ is the evaluation function at point c, $\hat{K}$ is some positive

constant, d is the size of a partition, f_{min} is the current best evaluation, and ϵ is a small convergence constant.

The constant $\hat{K}$ could be thought of as a slope, and the conditions could be conceptually restated as: "Is there a slope $\hat{K}$ such that the edge of partition j could be lower than the edge of partition i?" and "Would the edge of partition j be significantly lower than the current best evaluation?" 'Significantly better' is determined by the constant ϵ which can generally be set on the order of[47] 1x10^{-4}, or 0.01%.

The direct algorithm allows for a global gradient-free optimization with the only condition being Lipschitz continuity. Due to the global nature of the search there is a potential for significantly more evaluations as the number of local minima in the optimization space increases.

Genetic algorithms are a family of biologically-inspired evolutionary computation methods that can be successfully applied to optimization problems. General genetic algorithms have four common elements: populations of chromosomes; random chromosomal mutation; crossover between individual chromosomes; and a fitness function to evaluate each individual in the population[49][1].

As an example, again consider the Rosenbrock function. The two inputs to the function, X_1 and X_2 can be represented as genetic units (or genes) in the array

$$\text{solution} = [x_1, x_2]$$

where hundreds of different solutions could be generated and evaluated. The solutions with the highest fitness value (closest to zero in this case) are allowed to reproduce and generate the next set of solutions. These successful populations could be further improved by implementing mutation and crossover effects.

Like in nature, mutation in genetic algorithms modifies the current population and tends to cause a divergence of solutions[50]. The mutation operator

can take on several forms, including[51]: switching the values of two genes, reversing the order of genes, and adding a small amount of noise to the values of the genes. Each of these operators tend to spread out the population over the solution space.

Crossover is the genetic algorithm equivalent of reproduction between organisms. Two parent solutions are selected, and their genes are combined to generate a new solution. The benefit of crossover is based on the idea that short strings of genetic information can provide high fitness, and thus crossover should eventually combine these building blocks to produce an optimal organism. The mixing of existing genes tends to move the population toward a local optima, and so crossover tends to converge the population[50].

Genetic algorithms have several benefits over other optimization techniques. The genetic units can be any type of variable, such as continuous, discrete, or piecewise. The evaluation of solutions can be easily parallelized by increasing the size of the population, as each individual's fitness is independent of the total population. Finally, the objective function does not need to meet any strict criteria; there can be discontinuities and holes and the genetic algorithm will still be able to find a solution[51].

A comparative summary of the optimization methods presented is presented in Table 6.

Table 6: A comparison of the viable optimizers in Dakota

	Multi - Gradient	Direct	Genetic
Doesn't require a smooth objective function		X	X
Doesn't require a continuous objective function			X
Doesn't requires gradient information		X	X
Guaranteed to find global optima		X	
Allows independent evaluations	X		X

To predict the behavior of each optimizer, a parametric study was performed on each of the 1D models. In general the shape of the objective function is not known a priori, so it is very important to understand the assumptions and requirements built into each algorithm.

4. ONE DIMENSIONAL OPTIMIZATION

4.1 Analysis Overview

The one dimensional analysis of the exploding cylinder was performed in four stages: first the Dakota optimization framework was developed and tested on KO, a pure Lagrangian hydrocode based on HEMP[52]. Then a parameter study was performed on the one dimensional CTH model using Dakota, and the results were compared with the Gurney equation. Third, optimizations were performed on the CTH model with the methods in Table 6. Finally a simple design problem was analyzed where the outer radius was fixed and a second layer of wall material was introduced.

4.2 Dakota Structure for 1D

To run the one dimensional simulations on the HPCs a queue submission script was developed to reserve a single node for the duration of the parametric study or optimization. The HPC Excalibur has a total of 32 multi-threaded cores per node, allowing 64 concurrent processes, or 63 CTH evaluations alongside Dakota. The queue submission scripts and input decks can be found in Appendix II.

For the one dimensional simulations a templated input deck was developed for KO and CTH with the variables for inner and outer radius in curly braces as {r_o} and {r_i}. These values were calculated by Dakota between each iteration, and then they were passed to the preprocessor where they were substituted into the input deck. In the case of KO, the number of nodes per material was also calculated

and substituted into the input deck. KO is compiled with a fixed number of nodes, so the this nodes per material was calculated to minimize the change in aspect ratio. It is possible for significant numerical error to occur at the interface for a large changes in aspect ratio.

4.3 KO Analysis

KO is a Lagrangian hydrocode developed at Marquette University based on the equations of continuity and the conservation of momentum and energy[52]. KO supports several equations of state, including Mie-Grüneissen, snowplow, and ideal gas. To approximate the explosion within KO the bulk pressure, temperature, and velocity of the expanding gas was taken from STEX data[35] and set as the initial condition for the interior gas. The properties used in the KO simulation are given in Table 7.

Table 7: Material properties used in the KO simulation. Initial gas states from [35], Mie-Grüneissen constants from [52], thermodynamic variables from [53].

Material	P_o [MBar]	ρ_0 [g/cc]	U_o [km/s]	S	γ	σ_{yield} [MBar]
Product Gas	$2.0\,10^{-2}$	0.532	0.8	0.00	1.4	0.0
Casing	0.0	8.930	0.00	1.49	1.99	0.477
Ambient Air	$1.0\,10^{-6}$	0.002	0.00	0.00	1.4	0.0

The resulting parametric study is shown in Figure 11 along side the analytical gurney solution.

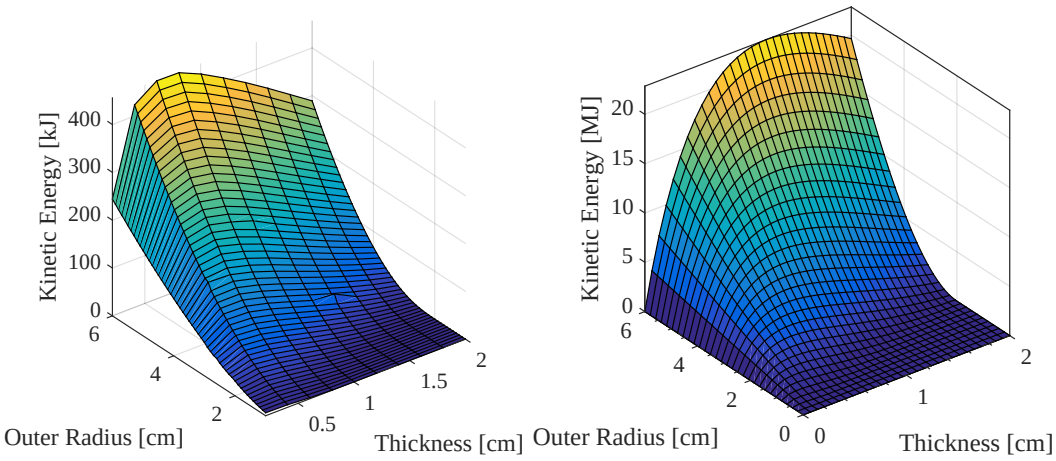

Figure 11: Parametric study results for the one dimensional KO simulation (left) and the analytical Gurney equation (right).

Considering all the assumptions built into the KO simulation, the shape of the two solutions is qualitatively consistent. The difference in magnitude is likely caused by the lack of an explosive model in KO, which would continue pressurizing the gas within the cylinder over time. The difference in surface shape can be explained by nondimensionalizing the results in Figure 11 with Equation 4, as shown in Figure 12.

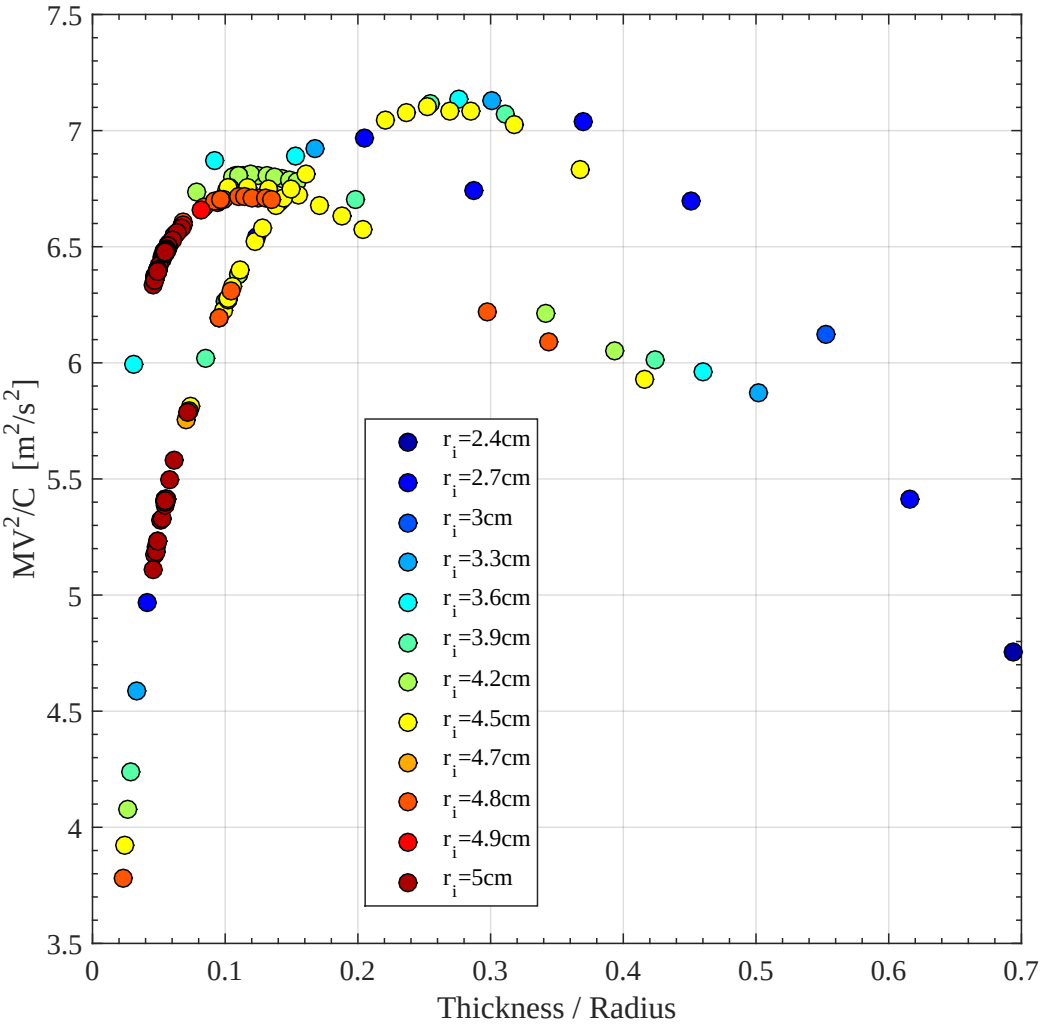

Figure 12: The nondimensional curve for a single parametric study in KO.

From Equation 4 the Gurney constant in Figure 12 is either 2.7 or 2.6 m/s, depending on which peak is chosen. This was likely caused by the initial gas pressure and velocity remaining constant while the case geometry varied. Each KO simulation effectively used a different explosive to achieve identical gas pressures and velocities independent of geometry.

Figure 12 also appears to approach zero as the ratio of thickness to outer radius approaches one. This trend is more intuitive and is compared with CTH in a later section.

4.4 CTH Analysis

CTH is an Eulerian shock physics code developed at Sandia National Laboratories. It contains several EOS and material models, and it is able to simulate one, two, and three dimensional meshes with second-order accurate numerical methods[54].

The one dimensional domain was 50 centimeters wide with 5000 cells, resulting in 0.1 mm resolution. As a rule of thumb, the smallest feature in CTH should be 10-20 cells across[43], allowing a minimum resolvable wall thickness of 1 mm.

A parametric study was performed with the aluminum and TNT values in Table 7 and compared to the Gurney equation. The resulting surface is shown in Figure 13.

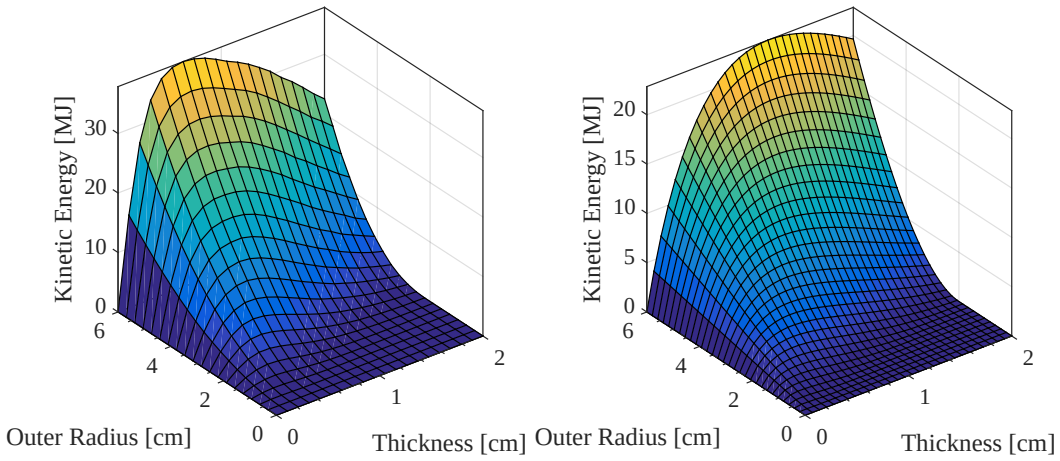

Figure 13: Parametric study results for the one dimensional CTH simulation (left) and the analytical Gurney equation (right).

The surfaces in Figure 13 are much closer than the KO results in Figure 11. As the wall thickness increases the kinetic energy of both surfaces approaches 200 MJ, but the maximum kinetic energy in the CTH simulations is much higher. This difference in shape is explored further in the section on 1D model drawbacks.

A gradient descent, direct, and genetic algorithm optimization were each performed on the one dimensional model. The surface in Figure 13 is smooth, continuous, finite, and contains a single maximum point, so each optimization algorithm is likely to converge. Analytically the optimal thickness and outer radius can be derived from the Gurney equation

$$\left(\frac{r_o}{r_i}\right)^2 = 1 + \sqrt{\frac{\rho_e}{2\rho_c}} \tag{9}$$

where ρ_e is the density of the explosive and ρ_c is the density of the case. Equation 9 is derived in Appendix I. The CTH and Gurney parametric studies are compared with a straight line drawn between the origin and the optimal point in Figure 14.

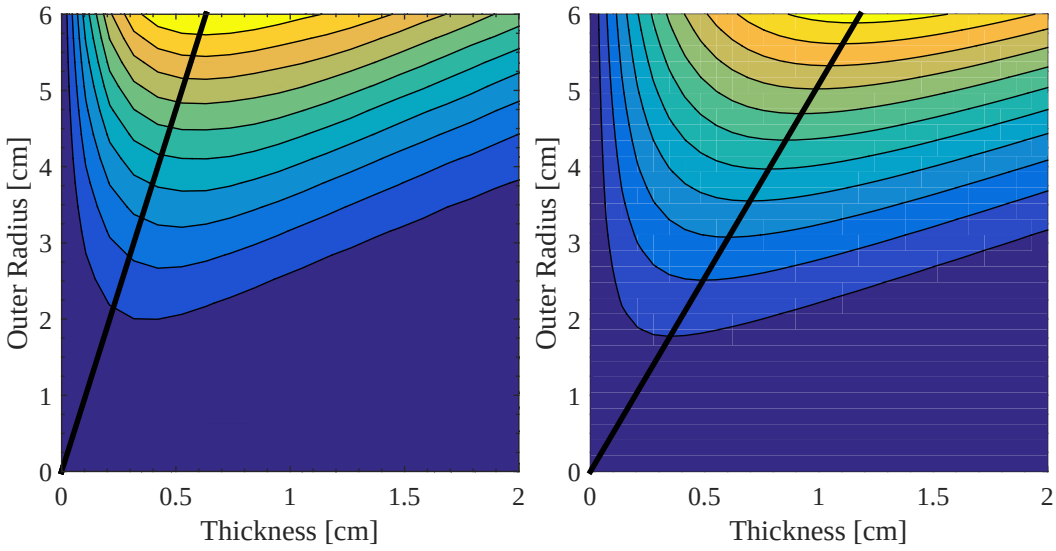

Figure 14: Contour plot for the one dimensional CTH simulation (left) and the analytical Gurney equation (right) with the optimal thickness line in black for each case. The CTH optimum line is calculated from the surface, while the Gurney line uses Equation 9

Figure 14 shows two different optimum lines. From the Gurney equation with the materials in Table 3 the slope of the line is 5.0853, whereas the CTH line has a slope of 9.5. This is a 61% difference between the simulation and analytical solution. To decouple any modeling error the optimization results are presented

with the parametric study rather than Equation 9. The evaluations for each algorithm are shown in Figures 15 and 20, and summarized in Table 8. The optimization visualizations are colored by evaluation number, with the initial guesses black and the final evaluation white for each case.

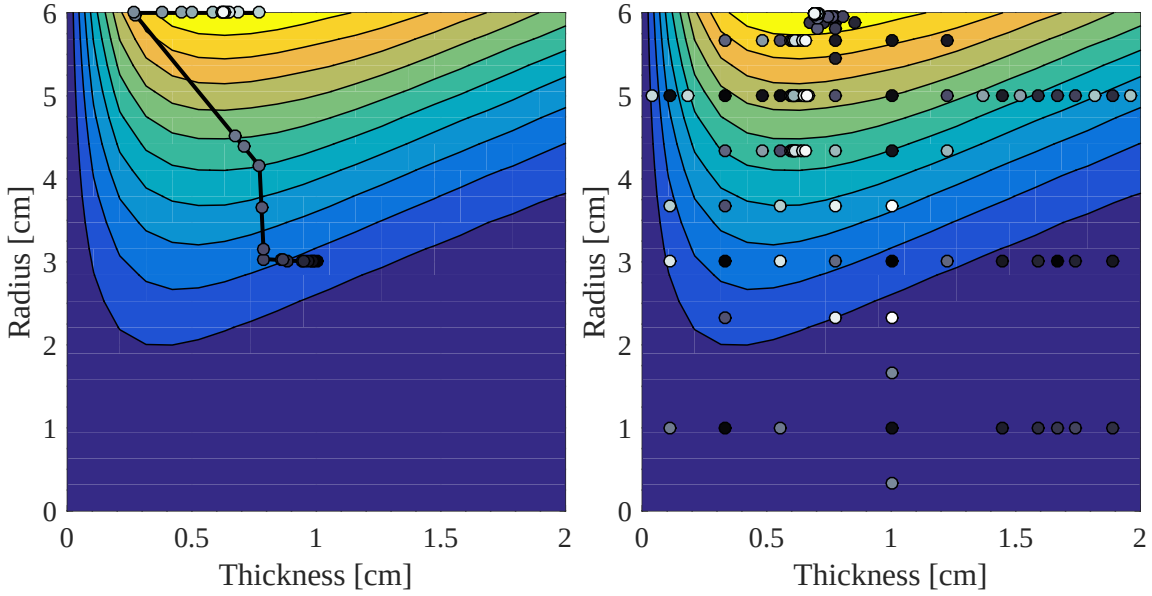

Figure 15: CTH Optimization trials plotted over the parametric study with gradient descent (left) and direct (right).

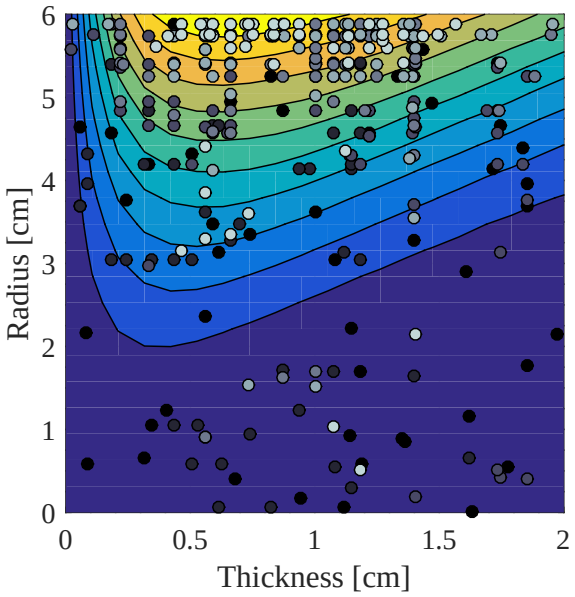

Figure 16: Results of the genetic algorithm optimization overlaid on the 1D CTH parametric study

Due to the smooth, continuous, and locally convex nature of this problem it is possible for the gradient descent method to efficiently reach the global maximum kinetic energy. The one caveat is the unphysical region where the wall thickness is greater than or equal to the outer radius of the cylinder. Kinetic energy in this region is zero everywhere, and so the gradient descent method would immediately converge. This can be avoided by intelligently picking initial parameters that correspond to a physical system.

The direct algorithm divides the solution space into rectangles, and continues subdividing areas that are potentially optimal. This results in more objective function evaluations than gradient descent, as the nonphysical region must still be sampled several times. It's interesting to note that in Figure 13 the gradient in the thickness direction is significantly steeper than in the radial direction. This resulted in the algorithm sampling more objective functions parallel to the thickness axis from the implicit gradient assumption built into the direct algorithm.

Dakota's Single Objective Genetic Algorithm (SOGA) successfully moved a majority of the genetic population to the high energy region at $r_o = 6$ cm in three generations. For this specific setup there were only two pieces of genetic information, the outer radius and thickness of the cylinder. This effectively negated the convergent properties of crossover as discussed in Chapter 3, which negatively impacted overall performance. The final results of each simulation are given in Table 8.

Table 8: One dimensional optimizer results compared to the parametric study. Thickness in the parametric study used a resolution of $\pm 0.07cm$. *The genetic algorithm used a population of 50, corresponding to 10 generations of evaluations.

Method [cm]	Radius [cm]	Thickness [cm]	Max KE [MJ]	Evaluations
Parametric Study	6.00	0.63	37.88	400
Gradient	6.00	0.63	37.90	90
Direct	6.00	0.69	37.81	133
Genetic	5.94	0.58	37.09	500$*$

4.5 1D CTH Nondimensionalization

Equation 4 relates the energy density of the exploding cylinder to the ratio of wall thickness and outer radius. The 1D CTH parametric study in Figure 13 was nondimensionalized with this equation, and the results are shown in Figure 17.

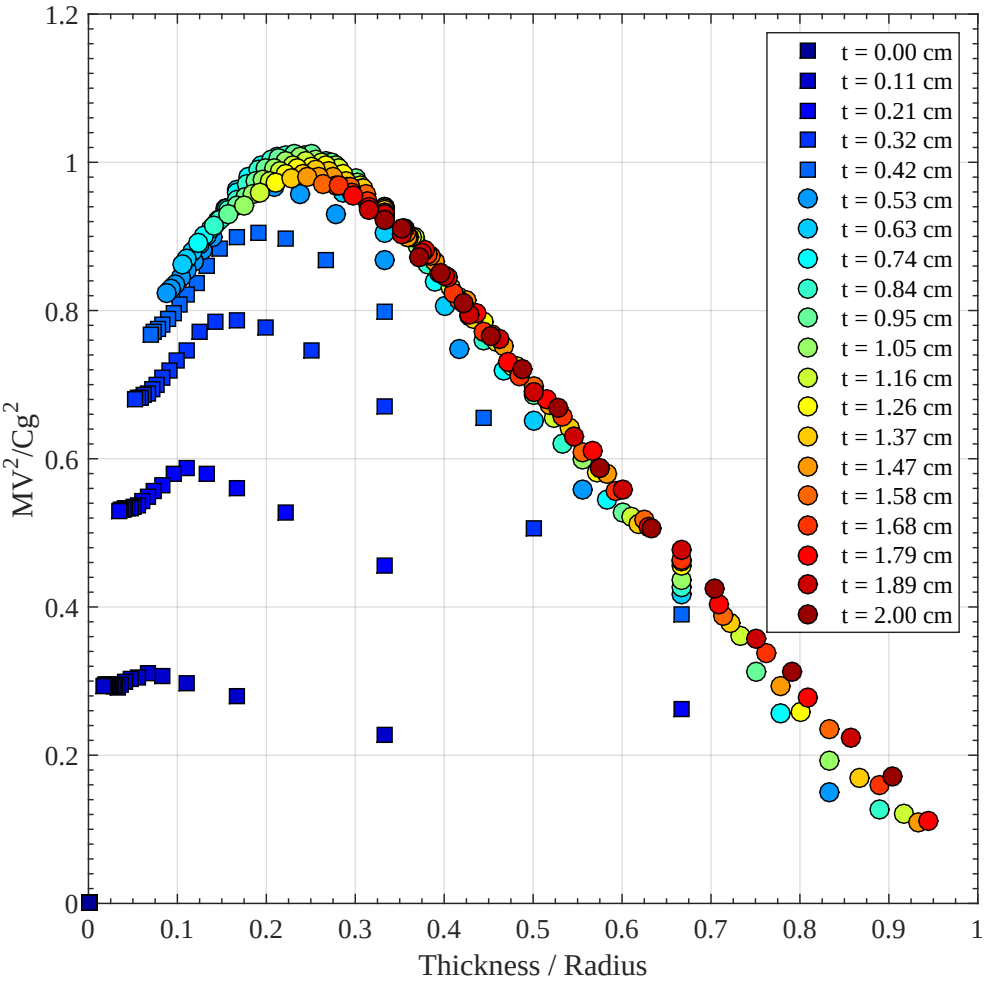

Figure 17: The nondimensional results of the parametric study for the one dimensional model.

Figure 17 was generated by dividing the kinetic energy output from the parametric study by the charge mass, as calculated from the geometry and Table 3, and the Gurney constant squared[6].

Each thickness above 1.5 centimeters the solution follows a single continuous curve. The curve in Figure 17 peaks much sooner than the Gurney solution in Figure 1, and quickly tends toward 0 rather than asymptoting to a value of 1.

The anomalous values (squares in Figure 17) only occur for very thin wall thicknesses. This inconsistency could stem from the low inertia of thin walls, which

are accelerated away from the explosive faster than the product gases can expand[29].

One adaptation of the Gurney equation inserts an infinitely hard solid core at the explosive's center, effectively removing the material not used to accelerate the case[29]. The size of this core can be calculated with the equation

$$\frac{r_h}{R} = 1 + 3\frac{M}{C} - \sqrt{(1 + 3\frac{M}{C})^2 - 1} \tag{10}$$

where r_h is the core radius, M and C are the case and explosive masses respectively, and R is the outer radius of the explosive. This approach reduces the charge mass of the system, which in turn would increase the energy density for the thin walled cylinders. By only considering the explosive that actually accelerate the case it is possible that the anomalous values in Figure 17 would line up with the other results. Unfortunately Equation 10 is derived from the Gurney equation, and would not necessarily fit Figure 17 to a single curve. The derivation of Equation 10 can be found in Appendix III.

4.6 Design Optimization

A secondary problem was developed to optimize the design of an explosive cylinder with a fixed outer radius and a case made of two materials. The parameters used for this optimization are given in tables 9 and 10.

Table 9: Geometric parameters for the design of a cylinder with two wall materials.

Outer Radius [cm]	Outer Thickness Range [cm]	Inner Thickness Range [cm]
5	$0 - 2$	$0 - 2$

Table 10: Material properties for the design of a cylinder with two wall layers.

Material [cm]	Density [g/cc]	Location [cm]
TNT	2.6	Inside Cylinder
Aluminum	3.2	Inner Case
Steel	7.8	Outer Case

In this case the inner wall material was aluminum, and the outer wall was steel. This structure has high ductility and density. The inner layer of aluminum is able to stretch to a much greater expansion ratio than steel, capturing more kinetic energy from the expanding gases. The dense iron allows the walls to be thinner while still maintaining high mass, and therefore high kinetic energy.

It is also possible to reverse the layering such that the lighter aluminum fragments expand faster than the iron core. Unfortunately it is not possible to confirm or deny either of these models without a gas leakage mechanism. Instead the sandwich configuration was used to generate a more complex kinetic energy profile to optimize.

In Figure 13 the maximum kinetic always occurred at the upper limit of the outer radius. In contrast the design problem is zero at the upper and lower limit of both variables, which should result in a more interesting optimal solution. The resulting parametric study is shown in Figure 18

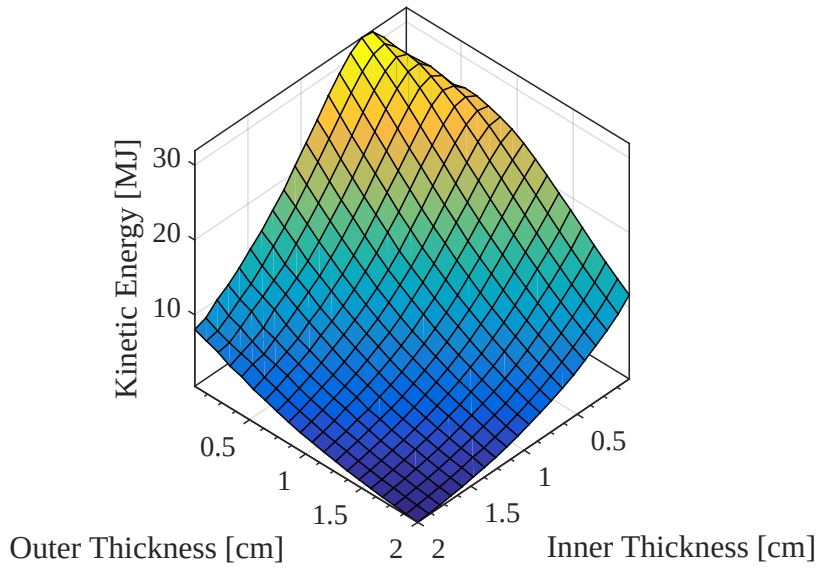

Figure 18: The result of the parametric study of the two material wall design. The energy surface is rotated 180 degrees for easier visualization.

From Figure 18 there is a long peak where both thicknesses are near zero. This shape could be explained by modeling the wall as a homogeneous material with an equivalent thickness and mass. It is possible to reach the optimal wall-to-charge mass ratio in Equation 9 with infinite combinations of thicknesses. The global maximum kinetic energy would occur at the lowest thickness that maintained the optimal case to charge weight ratio.

Additionally, the kinetic energy of the iron case is driven by the motion of the aluminum, rather than the gas directly. This configuration puts extra inertia behind the force acting on the outer iron layer, while also physically constraining the aluminum.

It is likely that the heterogeneous interactions also affects the results in Figure 18. As the product gas expands it acts on the inner aluminum shell, compressing it behind the much heavier iron shell. The expansion ratio of Aluminum is also much higher than steel, which should result in an overall increase in energy transfer between the gas and case. This phenomena is not testable with 1D CTH because there is no gas leakage model available.

The optimization results are given in Figures 19 and 20, and summarized in Table 11.

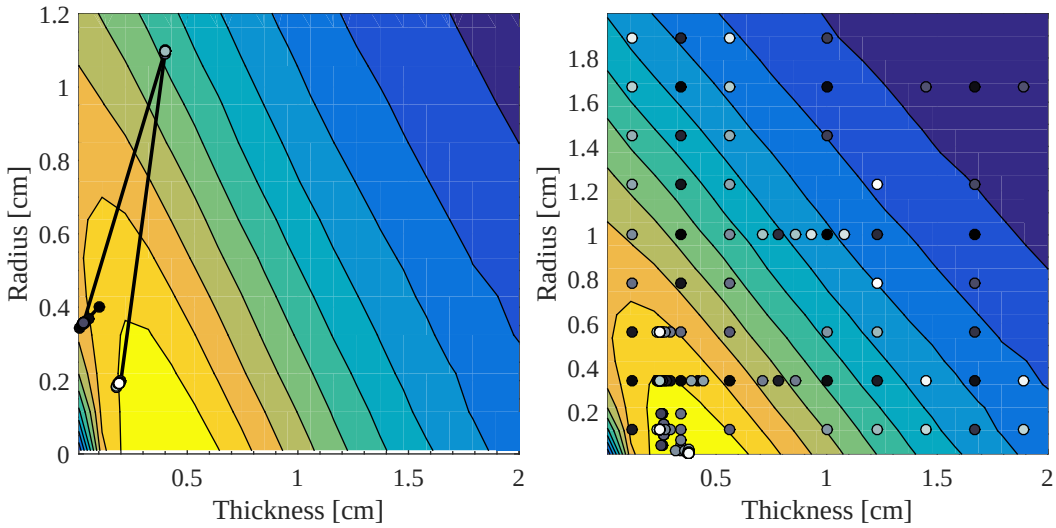

Figure 19: Gradient descent (left) and direct (right) optimizations of the simple design problem in 1D. Dot color represents evaluation index, with the initial guesses in black and the final evaluations in white.

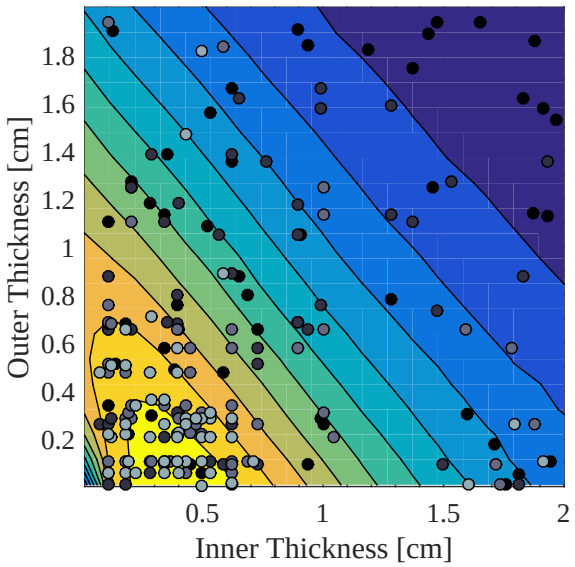

Figure 20: Genetic algorithm optimization of the simple design problem in 1D. Dot color represents evaluation index, with the initial guesses in black and the final evaluations in white.

Table 11: One dimensional design problem optimization results. Each algorithm is compared to the parametric study, which had a thickness resolution of ± 0.1047 cm. *The genetic algorithm contained 50 members which took 5 generations to converge

Method $[cm]$	t_{inner} $[cm]$	t_{outer} $[cm]$	Max KE $[MJ]$	Evaluations
Parametric Study	$0.4289\,cm$	$0.0100\,cm$	31.9	400
Gradient	$0.3780, cm$	$1.40\,cm$	17.73	36
Direct	$0.3778, cm$	$0.0101, cm$	32.0	171
Genetic	$0.4324\,cm$	$0.0097\,cm$	31.91	$250*$

As with the basic 2D model, the results in Table 11 show the gradient descent and direct methods converging relatively quickly, while the genetic algorithm took significantly longer. In this case the gradient descent method got stuck at a local minimum, and was unable to converge at the same solution as the parametric study and gradient descent. In cases like this it would be appropriate to use a multi-start gradient descent method, which involves running the algorithm from several initial points to try and converge at the global optimum.

4.7 Improvements on 1D Model

There are two issues with a one dimensional formulation of the exploding cylinder problem: end effects and gas leakage. Like with the analytical equations, the one dimensional model assumes a uniform velocity profile along an infinite cylinder length. It has been shown experimentally that the ends and edges have a significant effect on fragment velocity. This is most likely caused by low pressure release waves reflecting back into the product gases, reducing their pressure.

Gas leakage can be a significant problem for one, two, and three dimensional simulations. As an example, consider an early state of a one dimensional simulation in Table 12.

Table 12: Peak pressure and corresponding area for an explosive cylinder simulation.

Initial Area [cm]	Gas Pressure [Pa]	Temperature [K]
4.5	$2\ 10^{11}$	3000

The time scale of the gas expansion is so fast, on the order of microseconds, that it can be modeled as adiabatic. Assuming the gas is primarily carbon dioxide at 1000 K, the inner radius of the case at equilibrium can be calculated with the equation

$$\frac{P_m}{P_f}(r_m^2)^\gamma = (r_f^2)^\gamma \tag{11}$$

where γ is approximately 1.18[53], r_m is inner radius of the cylinder at peak pressure, r_f is the radius at equilibrium, P_m is the maximum pressure inside the cylinder, and P_f is the pressure at equilibrium.

Using the values from Table 12 and a final pressure of 1 atmosphere results in a final inner radius on the order of 20 meters. Even as a rough calculation it should be clear that lack of a gas leakage model is significant, and without it the simulation is only accurate up until the point of fracture.

One possible solution is to incorporate the expansion ratios described by Taylor[39] and Predebon[20]. The expansion ratio appears to be relatively independent of thickness, so it could be used as a spatial termination criteria either within CTH or the post-processing script. Thin-walled cylinders are accelerated more quickly than those with thick walls, allowing them to reach a higher expansion

ratio and absorb more kinetic energy from the product gases. This leads to over-prediction of the velocity at low thicknesses, such as seen in Figure 13.

There is an issue with using the expansion ratio in the postprocessing script, as seen in Figure 21.

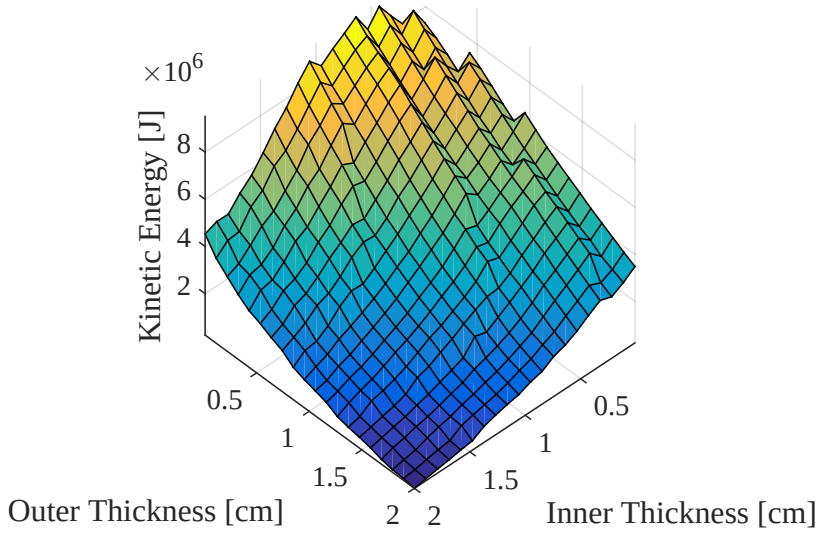

Figure 21: Parametric study results for the two material wall problem when the expansion ratio cutoff is used. Notice the significant noise compared to the same parametric study in Figure 18.

The extra noise on the energy surface in Figure 21 comes from a discretization error between the post-processor and CTH output files. For each simulation CTH outputs a text file containing the state of every cell. These output files are generated at regular intervals, around 5 microseconds in this case, and read by the post processor. This temporal discretization allows the expansion ratio to vary between 2.0 and 2.2, as the case only needs to travel a few millimeters at a velocity on the order of 1 km/s.

As the case thickness gradually decreases it is able to travel further in the same amount of time, and eventually the cylinder reaches an expansion ratio of 2.0 one time step earlier. This results in a sharp drop in kinetic energy, which can be seen in Figure 21. A more realistic solution would involve modifying the CTH

source code to terminate at a specific distance, thereby leveraging the variable time step capabilities of the code to exactly meet the expansion ratio every time.

5. Two Dimensional Optimization

5.1 2D Model Overview

The two dimensional system was modeled as a rectangle in cylindrical space, as seen in Figure 9, with the left edge of the simulation at the centerline. Boundary conditions for this model are given in Table 13.

Table 13: Boundary conditions for the 2D simulation.

Left	Right	Top	Bottom
Reflective	Transmissive	Transmissive	Transmissive

The left edge of the simulation domain is the centerline of the explosive, and a reflective boundary was used to enforce radial symmetry. The remaining boundaries were set to transmissive to allow the hot expanding gases to escape the domain, rather than reflecting back into contact with the case.

As the exploding cylinder expands it forms long thin strips along its length[39]. This axisymmetric fracturing can achieved with a rectangular cross section, which is why it was selected over a circular cross section. This geometry also resulted in a non-uniform velocity profile along the length of the cylinder, with a reduced magnitude at the ends and a maximum near the center. This complication is addressed in the section on updating the Dakota input deck and processing scripts.

Several changes were made to the CTH input deck for the 2D model. Two simulation geometries were developed, one where the ends of the cylinder were left

open, and another where heavy end caps were inserted. These geometries can be seen in Figure 22.

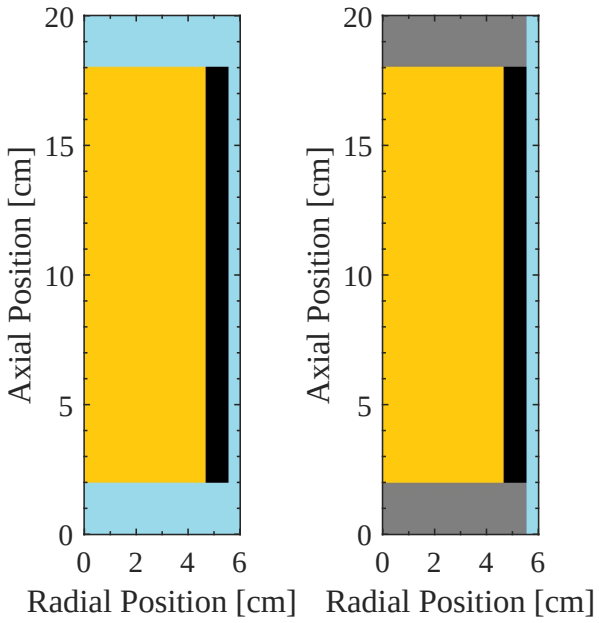

Figure 22: Free end (left) and capped end (right) CTH models for the 2D exploding cylinder. The x-axis of the domain extends out to 20 cm.

Each material is represented by a different color in Figure 22, the explosive is yellow, the aluminum wall is black, the ambient air is light blue, and the fixed end caps are gray. The 2D models both used the same materials in Table 3, and the 2D input decks can be found in Appendix II. The new parameters for the 2D model setup are given in Table 14.

Table 14: Model properties for the 2D simulation.

Length [cm]	Thickness [cm]	Diameter [cm]
16	0-4	0-6

5.2 Dakota Updates

By expanding the domain to two dimensions the number of cells in the simulation was squared. As a rule of thumb, CTH is most efficient when there are $\approx 30,000$ cells per processor[43]. To get a reasonable resolution for small wall thicknesses the 2D model used 4 million cells, which required approximately 130 processors, or just over two Excalibur compute nodes. This effectively required Dakota to launch one CTH job per two nodes to complete each run in a reasonable amount of time.

Unfortunately there is one major drawback on Excalibur, and the Cray XC systems in general, that makes this approach infeasible: any job running on a compute node can not start a separate job on a different compute node. This limitation causes two problems. First, if Dakota is running on a compute node it is unable to directly spawn a child CTH process on a separate reserved node. This was the method used to run the 1D simulations, but it was possible to run one job processor, resulting in 63 parallel evaluations, due to the low cell count in each model. Second, Dakota is unable to submit CTH jobs to the queue while running on a compute node. This effectively stops CTH and Dakota from running simultaneously.

There are two other methods that could be used to overcome this limitation. It is possible to run Dakota in parallel on several nodes and have each instance run its own CTH simulation. Unfortunately running a parallel application from a parallel parent is undefined behavior for most systems[1].

The other solution is to run Dakota interactively by having it submit several CTH jobs to the queue and exit out. After the jobs are finished running Dakota can run the post-processing scripts to get the final objective function values. This method must be repeated for each iterations of jobs. This solution is adequate for

parametric studies, and it could be used to launch 50 parallel genetic algorithm jobs at a time, but it is infeasible to use with the direct or gradient descent algorithms. Updated input decks for this pipeline can be found in Appendix II.

Moving to two dimensions also makes the post processing more challenging. To deal with the extra dimension this step was split into two parts. The first step calculated a case velocity profile by summing the velocity in the radial direction as a function of axial position. The second step then took the maximum point of the velocity profile and returned it to Dakota. This corresponded to the velocity of an ideal explosive, which is what the Gurney equation and 1D simulations calculated. If the average of the velocity profile was used instead then the results between 2D and 1D/Gurney would not be comparable. Updated post processing scripts can also be found in Appendix II.

5.3 CTH Results

Two 2D parametric studies were run to compare with the 1D and Gurney Equation results. One model had free ends, which allowed the gas to quickly leave the domain after the detonation was initiated. This simulation was compared to the Taylor angle formula from Equation 1 as well as the previous results. A second model was simulated with two heavy end caps, which should force the gas to accelerate the case to a higher velocity. This model was also compared to the prior results.

The free-end CTH model was developed first, and the kinetic energy surface was calculated to compare with the 1D and Gurney equation results. For comparison the Gurney equation results and 1D CTH results are given below.

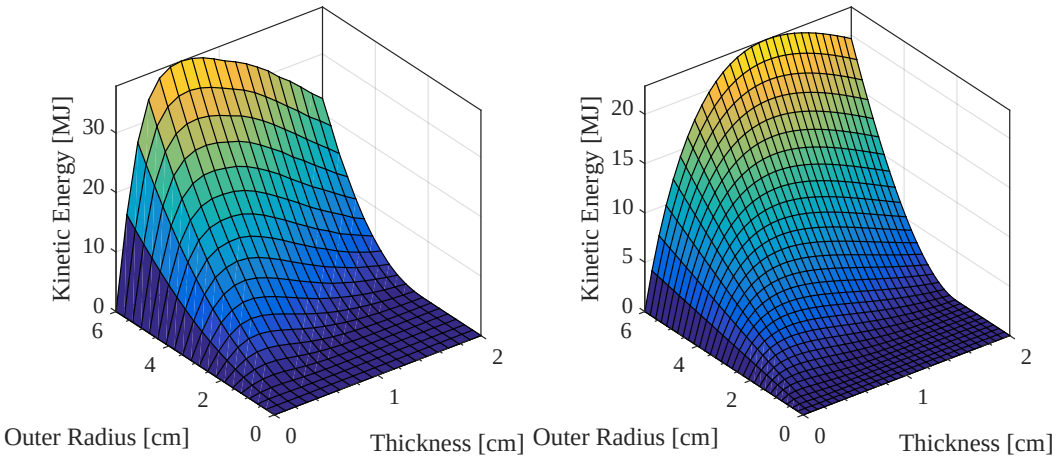

Figure 23: Original results from the 1D CTH simulation (left) and Gurney equation solution (right).

Some kinetic energy values for the 2D free-end model were significantly lower than expected by both surfaces in Figure 23. This was most likely cause by a file IO misalignment between the CTH output and the C++ processing file. These anomalous values are significantly lower than anything predicted by the 1D model and Gurney equation; the values were removed and interpolated over to create the second surface in Figure 24.

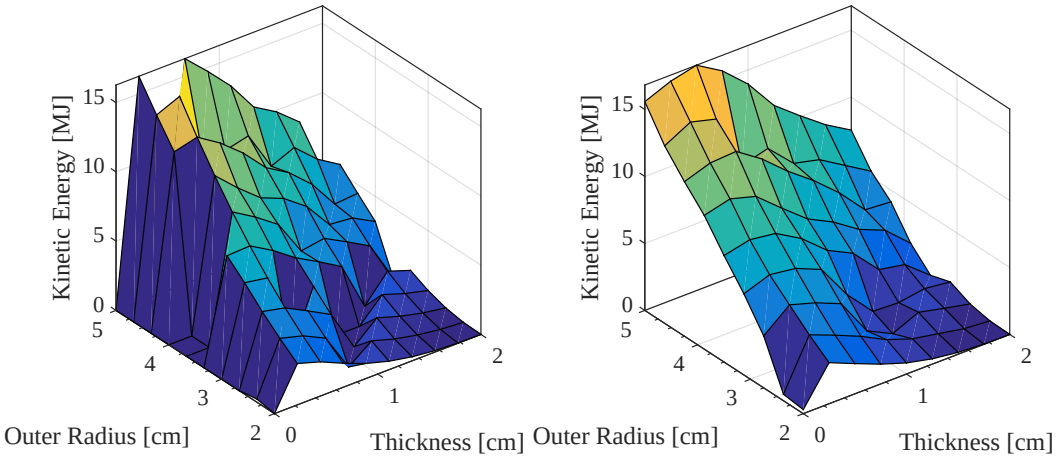

Figure 24: Kinetic energy surface for the 2D model with free ends. The raw results (left) and corrected surface (right) are both shown.

The shape and magnitude of Figures 23 and 24 are similar, and both share the same asymptotes at $t = 0$ and $t = r_o$. The 2D results are close in magnitude to

the Gurney solution with a 15 MJ output around 0.1 and 1.5 cm thicknesses. The 2D free-end model had a lower peak kinetic energy than the 1D model, which was expected from the additional gas leakage.

The same processing was applied to the 2D simulation with end caps, which is shown in Figure 25.

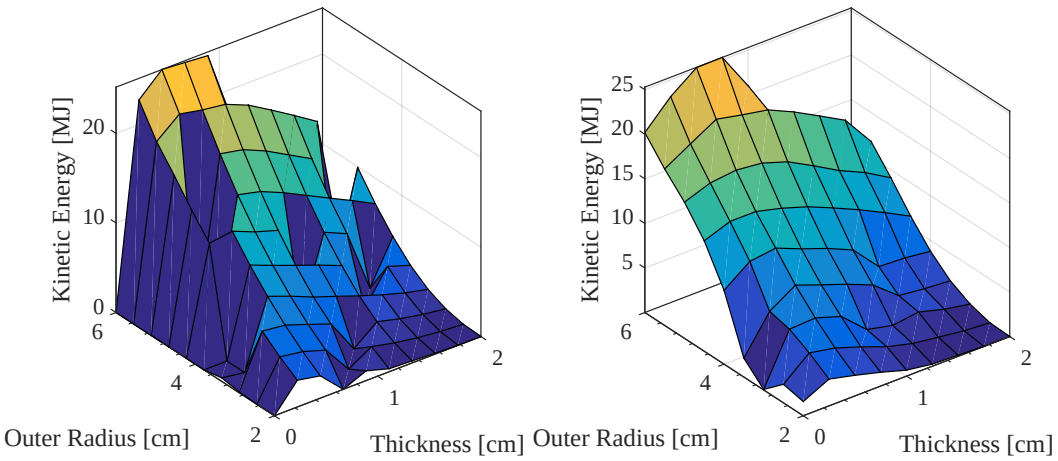

Figure 25: Kinetic energy surface for the 2D model with free ends. The raw results (left) and interpolated surface (right) are both shown.

The 2D results with end caps appear to be somewhere between the 1D CTH simulation and the Gurney equation. The Kinetic energy peak is at 25 MJ, which is lower than 1D, but is at a thickness of 0.4 cm, which is earlier than the results of the Gurney equation.

The 2D results were also nondimensionalized from Figures 24 and 25, and the result is shown in Figure 26.

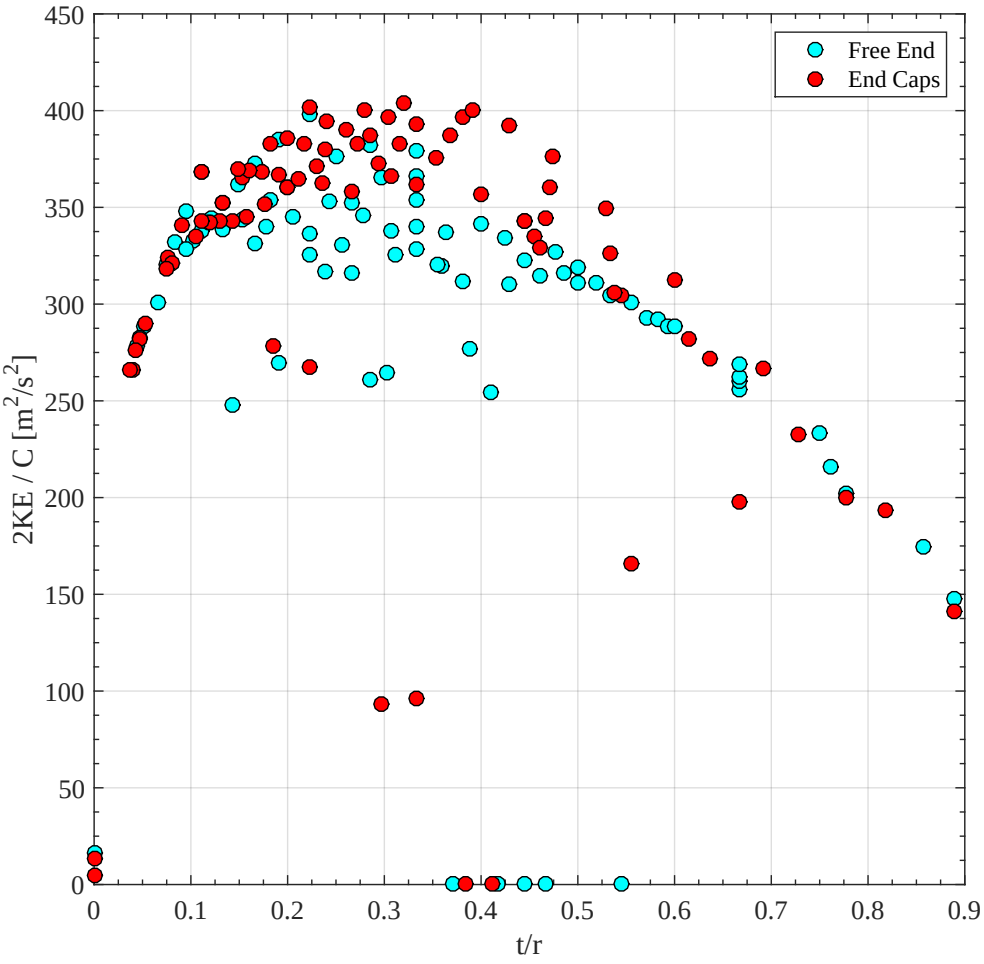

Figure 26: Nondimensional 2D CTH model results.

As with the 1D results, Figure 26 appears to asymptote toward zero as the thickness over radius ratio approaches 1. The free-ended model seems to peak at a ratio near 0.2, while the model with end caps peaks closer to 0.35. For comparison, the 1D results in Figure 17 peak at approximately 0.2.

The peak in Figure 26 occurred at almost exactly 400 m^2/s^2, which would require a Gurney constant of 20 km/s. For TNT this value is around 24 km/s, which is 17% higher.

6. Analysis

6.1 Optimizer performance

Tables 8 and 11 show the performance of the gradient, direct, and genetic algorithms compared to the parametric study with 400 evaluation points. In both cases the gradient and direct algorithms significantly outperformed the parametric study, while the genetic algorithm was only marginally better than a full parametric study.

The kinetic energy surfaces for both problems were smooth and continuous. These special conditions allowed the gradient descent to converge ≈13 times faster than the parametric study. For problems with several peaks or poorly conditioned gradients it is would be necessary to employ a multi-start method and re-run the algorithm several times. This would significantly increase the number of evaluations required, and finding the global maximum would not be guaranteed. A good approach may be to use a simple model, such as the Gurney equation, to characterize the optimization surface before using gradient descent.

The direct method performed significantly better than the parametric study (≈3 times fewer iterations) and genetic algorithm(≈2 times fewer iterations). For a system with multiple peaks it is likely that the direct algorithm would outperform gradient descent, and it would also find each optima during the global search. Direct is well suited to problems where the energy surface can not be easily predicted with a simpler model, such as the Gurney equation.

The genetic algorithm found the global maximum in slightly fewer iterations than the parametric study. This is likely due to the fact that these simple problems only have two optimization variables, which resulted in only two pieces of genetic

information for the algorithm to modify. The population of the genetic algorithm was 50, which was around the same size as the number of iterations for the gradient descent algorithm to converge.

It is likely that as the number of optimization variables increases the performance of the genetic algorithm would also increase relative to a full parametric study. Of course, it is possible that the genetic algorithm could get stuck at a local maximum if not tuned properly. The genetic algorithm is well suited to higher dimensional problems, or those with discontinuities in the optimization variables. For example, a cylindrical explosive with two layers of material, a variable radius, and discrete properties for each layer of material.

6.2 Computational Model Accuracy

The 2D model resulted in a kinetic energy surface similar to the 1D simulation. Overall the 1D model had a higher peak energy output than the 2D and gurney solutions, while also reaching this value at a lower thickness. This inaccuracy is likely caused by the thin-walled simulations reaching a greater expansion ratio than the thicker evaluations.

The shape of the 2D surface is closer to the Gurney solution than the 1D model. A big part of this is the extra gas leakage in both the free-ended and capped model. This gas leakage also significantly reduced the overall kinetic energy output of the cylinder, as the high pressure gas is escaping to the ambient atmosphere rather than pushing on the cylinder.

6.3 Experimental Data

To further verify the CTH model, and as a comparison to the Gurney equation, experimental and simulation results were taken from a variety of sources in the literature. The Lawrence Livermore National Laboratory (LLNL) Explosives Handbook[55] was used to approximate any material properties that were not explicitly given. The data sets consist of copper, aluminum, and steel casings filled with a variety of explosives, including TNT, PBX, and Composition B.

Some data sets calculated the radial velocity as a function of axial position. The experimental data from [23] is shown in Figure 27, where the terminal velocity profile is given for an AISI 1045 steel cylinder filled with Cyclotol, a TNT and RDX derivative.

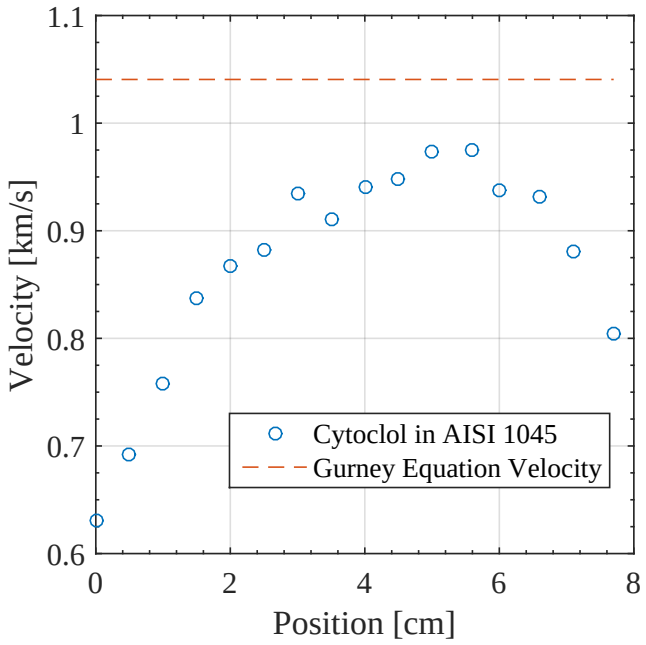

Figure 27: Experimental data of terminal velocity versus axial position from [23], with the Gurney solution shown.

The data in Figure 27 comes from a cylinder with a length to diameter ratio near 1, and the release waves significantly affect the entire velocity profile. In

experiments with a higher length to diameter ratios the velocity profile tends to flatten out at the Gurney velocity[44].

When comparing position dependent data, such as Figure 27, the maximum velocity along the profile was taken as a single data point. The previous optimization were performed to maximize the kinetic energy of an idealized explosive with a constant velocity profile; this corresponds to the point of maximum velocity within the data. Due to the significant release waves in this particular data set, it is likely that the velocity will be lower than what was predicted by CTH and the Gurney equation.

6.4 Dimensional Analysis

The nondimensionalized CTH results are presented again in Figure 28 for parametric study on an aluminum cylinder filled with TNT.

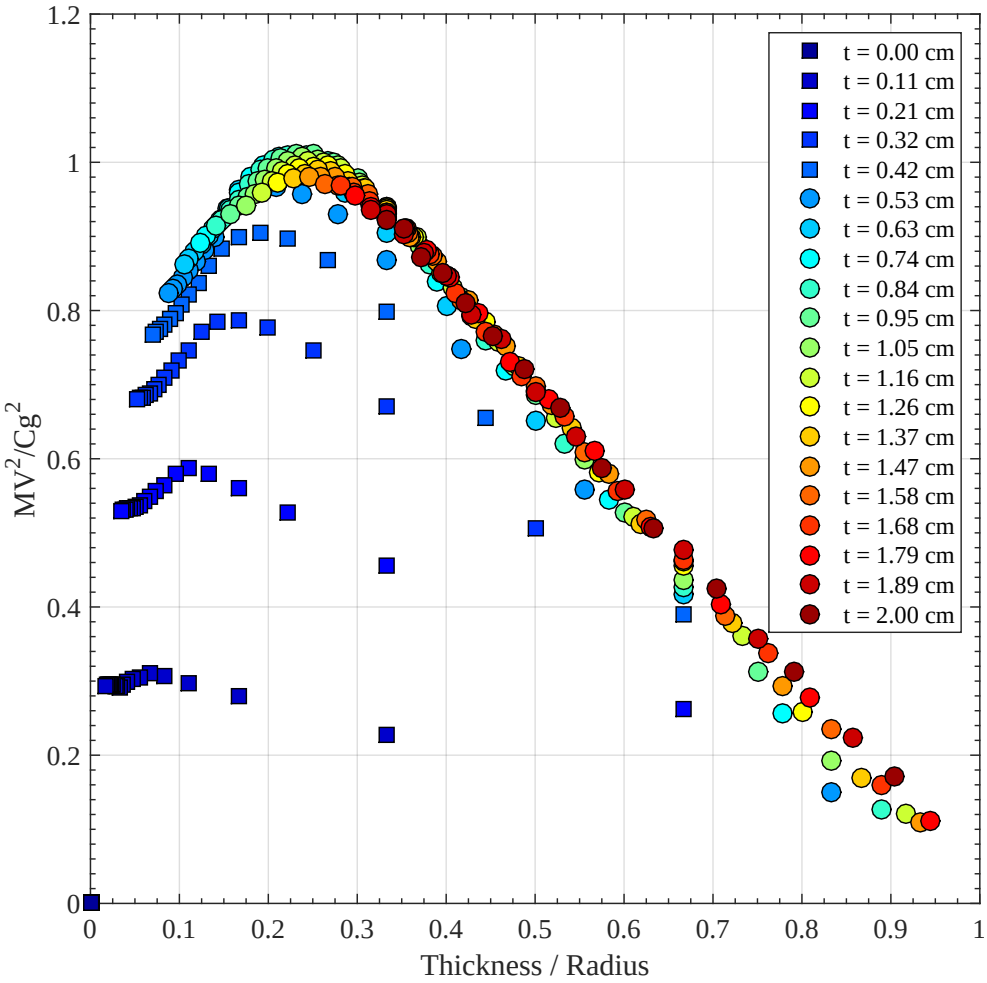

Figure 28: The nondimensional relationship between energy and scale for a TNT filled aluminum shell.

Neglecting the thin walled region in Figure 28 results in a single smooth continuous curve with zeros at $x = 0$ and $x = 1$. The shape of Figure 28 can be approximated by multiplying an exponential decay with a horizontally asymptoting function. The zero at $x = 0$ can also be enforced by fitting the empirically determined equation

$$y = A\, e^{B\,x}\,(1 - e^{C\,x})\,(1 - x)^{D} \tag{12}$$

to the nondimensional data, where x and y are nondimensional scale and energy respectively and A, B, C, and D are tunable parameters. Equation 12 was fit to the curve as shown in Figure 29 using Matlab's cftool command. The case material was

aluminum with a density of 2.70 g/cc, and the explosive was TNT, with a density of 1.63 g/cc and a Gurney constant of 2.44 km/s.

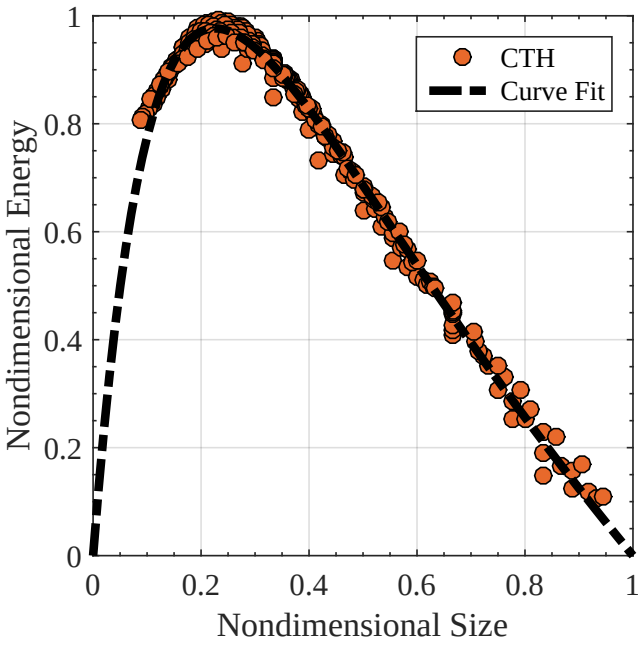

Figure 29: Nondimensional CTH results of a TNT filled aluminum case fit with Equation 12 using the curve fit tool in Matlab.

The constants in Figure 29 are $A = 1.609$ $B = 0.2831$ $C = 8.003$ $D = 1$. This fit had an R^2 value of 0.993 and an RMS Error of 0.019. Equation 12 fits the shape of the data almost exactly. For very thin walls ($\frac{t}{r_o} < 0.1$) CTH appears to diverge from the curve fit, but this cannot be confirmed without a series of high resolution simulations. Equation 12 was also applied to a copper explosive filled with TNT, and is shown in Figure 30.

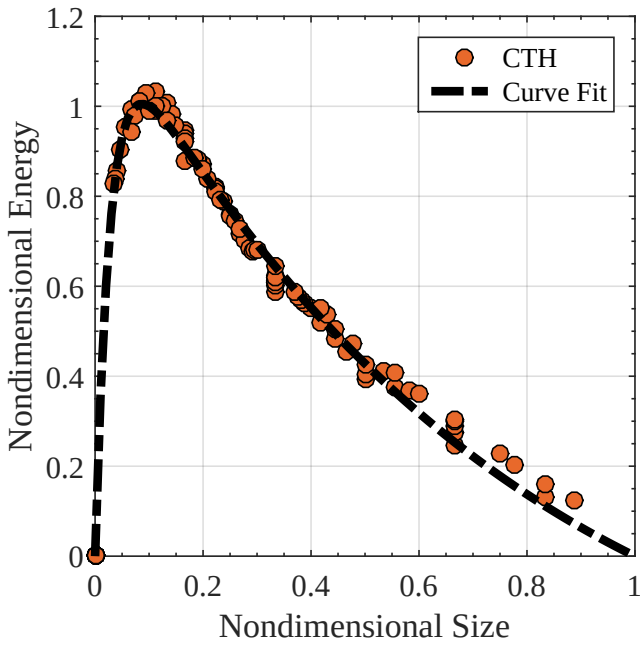

Figure 30: Nondimensional CTH results of a TNT filled copper case fit with Equation 12 using the curve fit tool in Matlab.

The constants in Figure 30 are $A = 1.234$ $B = 0.7316$ $C = 34.70$ $D = 1$. The R^2 value of this fit was 0.994, with an RMS error of 0.025. All of the constants in Equation 12 are nonphysical and can only be calculated by fitting the equation to data. The constants could be tabulated for every explosive and material combination, and kept in a handbook for special design problems.

A more compelling case can be made for Equation 12 being a solution to the differential equation that describes the output of an exploding cylinder. Equation 12 was applied to the Gurney solution in Figure 1 by setting $A = 1$, $B = 0$, $D = 0$, and tuning the parameter $C \approx 5.2$. This fit can be seen in Figure 31 with the same material properties as Figure 29.

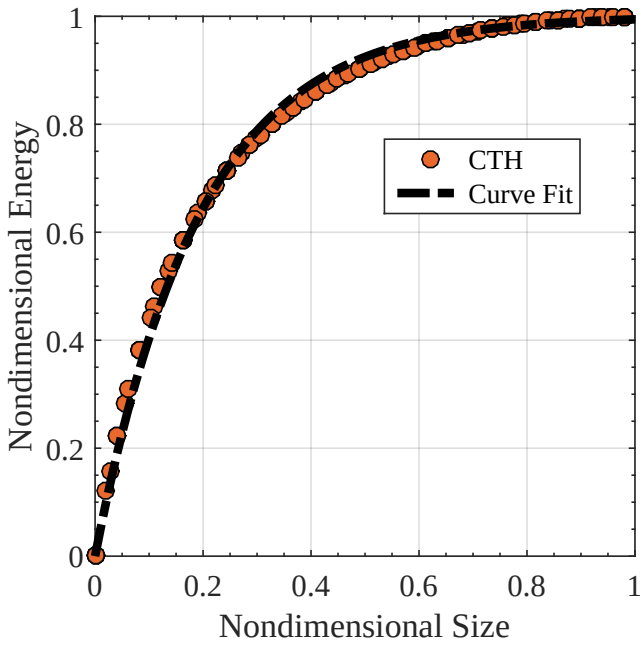

Figure 31: Nondimensional plot of the Gurney equation solution fit with Equation 12.

6.5 Comparison of Gurney, CTH, and Data

To compare the Gurney equation solution and CTH experimental data was taken from the literature[31][56][57][30][58][24][23][20][33]. The data were nondimensionalized by values given in each source, or with the LLNL Explosives Handbook[55] when no parameters were provided. A comparison of the CTH results, Gurney equation, and literature data for a TNT-filled aluminum case is given in Figure 32.

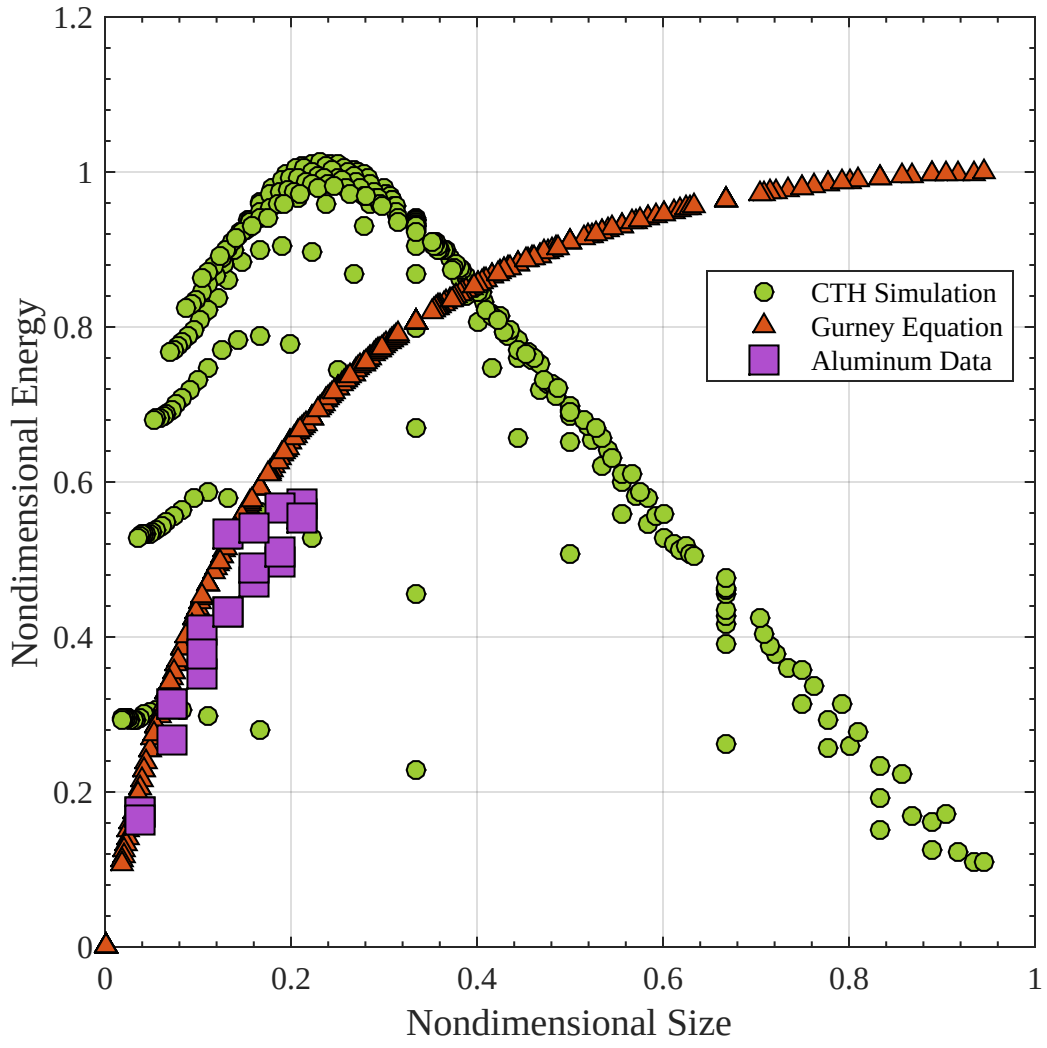

Figure 32: Nondimensional comparison of the Gurney equation, the 1D CTH parametric study, and experimental data for aluminum and TNT.

A second comparison for copper and TNT can be seen in Figure 33.

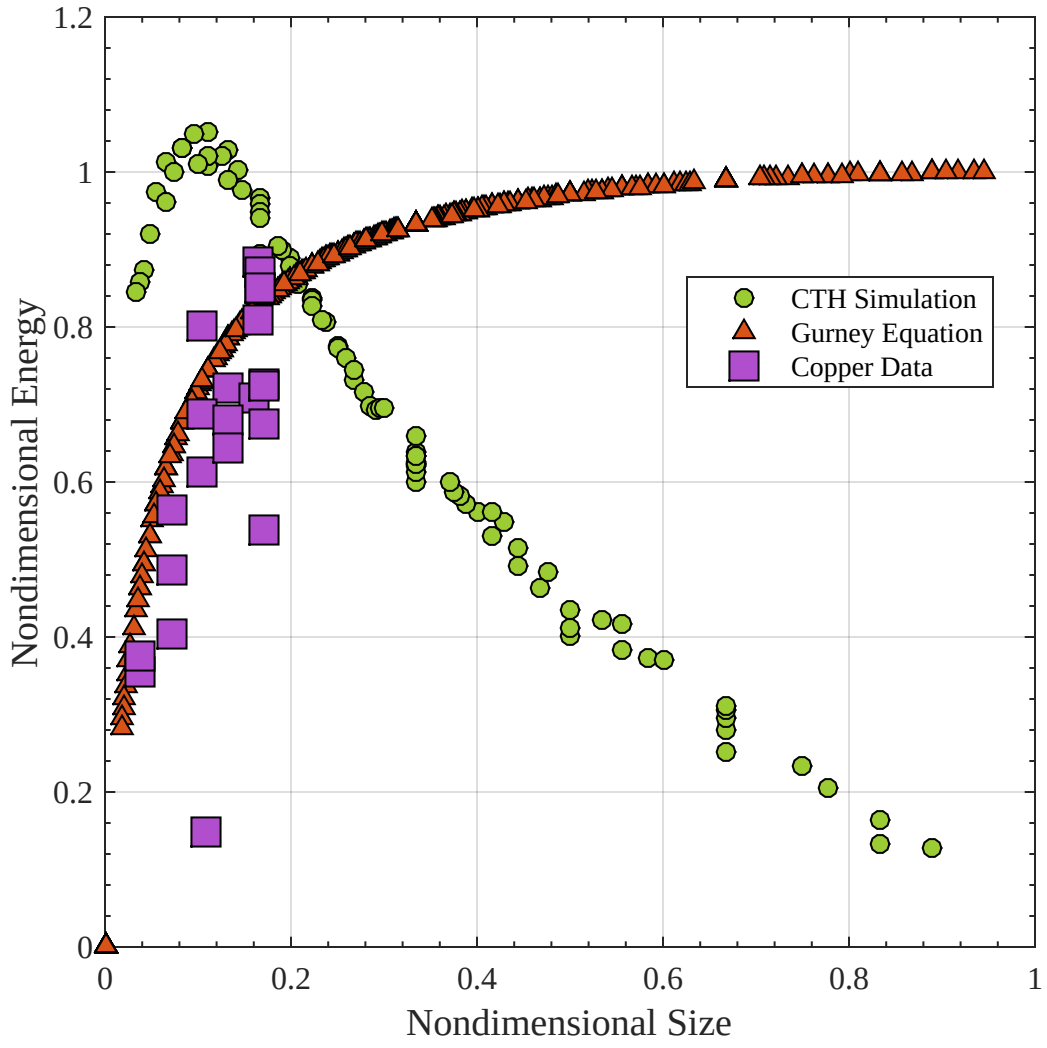

Figure 33: Nondimensional comparison of the Gurney equation, the 1D CTH parametric study, and experimental data for copper and TNT.

The results in Figures 32 and 33 were combined to create Figure 34. This results show a full comparison of the CTH models, Gurney equation results, and data from the literature.

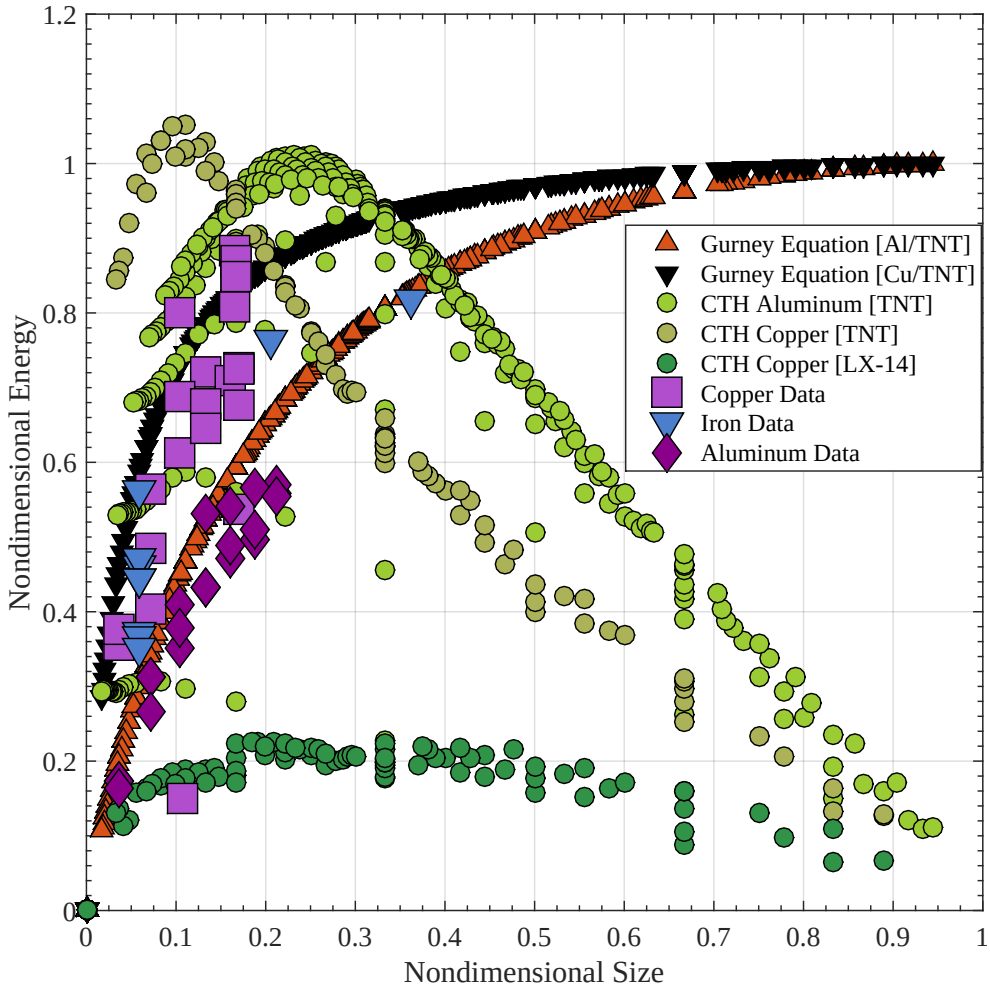

Figure 34: Nondimensional comparison of the Gurney equation, the 1D CTH parametric study, and all experimental data.

As expected the Gurney equation matches the data fairly well. CTH reaches a maximum energy density at a much lower wall thickness than the data, as expected from the earlier 1D analysis. Unfortunately it is impractical to test explosives with significant thickness to diameter ratios, so there is no experimental data to show whether the energy density asymptotes to 1 or falls off to zero on the right side of Figure 34.

There are a couple factors that could be responsible for the difference between CTH and the data. It is possible that the exploding cylinder problem is fundamentally a two or three dimensional problem. This is not a convincing

argument, as the Gurney equation is a one dimensional conservation of energy approach that matches experimental data very well. However, this match is all but guaranteed, as the empirical constant in the formula is based on the same experimental data that it's being compared to.

Another factor is the CTH simulation used a temporal termination criteria rather than an expansion ratio. By running thin and thick walled cylinders for the same amount of time the cylinders with thinner walls ended up with a significantly higher expansion ratio, and therefore a larger velocity. This caused the peak kinetic energy to shift toward lower wall thicknesses in the simulations; a trend which was observed in the kinetic energy surfaces of Figures 13 and 18, as well as in the nondimensionalized plots.

7. Concluding Remarks

7.1 Conclusion

The optimizers tended to outperform the parametric study in both number of iterations and maximum kinetic energy found. This was true for both models, which were smooth and continuous surfaces with a single peak. The gradient descent method had the best performance, with the direct algorithm a close second, and the genetic algorithm performing the worst of the three. The gradient descent method is not guaranteed to converge, and in general a multistart method may be required for more realistic problems.

The one dimensional problem was similar to the kinetic energy predicted by the Gurney equation. The biggest difference resulted from CTH simulating every geometry for the same amount of time. This allowed thin-walled cylinders to expand significantly past their expansion ratios, artificially increasing their peak kinetic energy. This is supported by the fact that as wall thickness increased the parametric kinetic energy surfaces generated by CTH seemed to asymptote toward the Gurney equation solution. Unfortunately, implementing a useful spatial termination criterion would involve modifying the sensitive CTH source code, which is outside the scope of this thesis.

The 2D model appeared to be a more realistic model of the exploding cylinder. Several issues with the 1D model, as discussed earlier, resulted in CTH over-predicting the final kinetic energy of the wall for low thicknesses. It is likely that with a spatial termination criteria the 1D model could be as accurate as a 2D simulation. The exploding cylinder problem appears to be a fundamentally one dimensional problem.

7.2 Future Work

There are two immediately obvious ways the work presented could be extended. The first is to use a more sophisticated model which leverages the full computational power of CTH to optimize a nontrivial design problem. The second option is to run the optimization framework with a separate simulation package, either instead of or in parallel with CTH. Several possibilities are given in the following paragraphs.

One ill-understood area of explosives research is in the design of shaped charges. By using the raw computational power of an HPC it is possible to apply this framework to the task of optimizing the profile of a shaped charge. The objective function could be almost anything related to the system, such as maximizing the velocity of the jet or maximizing the overall energy efficiency.

The framework could also be applied to optimizing the material properties of a composite explosive. The optimization variables could be discrete materials, a continuous range of values for mechanical properties, or a combination of the two. This would be useful for cases when the geometry constraints are exactly known for a problem, but the optimal material is unknown.

Finally, it may be reasonable to develop advanced analytical models of cylindrical explosives by applying the shock relation equations. This could be developing a physics-based equation for the effect of release waves on case velocity, the variation of detonation velocity with diameter, or a relationship for wall velocity at very thin thicknesses. For each case the literature appears to use empirical correlations based off Gurney and other equations. These correlations are useful, but a physics-based model may yield more information about the fundamental phenomena behind explosives.

8. References

[1] B. M. Adams et al. "Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.5 User's Manual". In: (2014).

[2] J. M. McGlaun, S. L. Thompson, and M. G. Elrick. "CTH: A three-dimensional shock wave physics code". In: *International Journal of Impact Engineering* 10.1 (1990), pp. 351–360. ISSN: 0734743X. DOI: 10.1016/0734-743X(90)90071-3.

[3] B. J. Buchanan. *Gunpowder, Explosives, and the State: A Technological History*. Routledge, 2006. ISBN: 0754652599.

[4] T. L. Davis. *The Chemistry of Powder and Explosives*. MIT, 1943. DOI: 10.1021/ed020p414.1.

[5] A. Sobrero. "Sur Plusieur Composés détonants produits avec l'acide nitrique et le sucre, la dextrine, la lactine, la mannite et la glycéine. (French) [On Several Detonating Compounds Produced With Nitric Acid and Sugar, Dextrin, Lactose, Mannitol, and Glycerine]". In: *Comptes Rendus* 24 (1847), pp. 247–248.

[6] P. W. Cooper. *Explosives Engineering*. Wiley-VCH, 1996. ISBN: 9780471186366.

[7] M. A. Meters. *Dynamic Behavior of Materials*. Wiley-VCH, 1994. ISBN: 978-0-471-58262-5.

[8] M. E. Tat et al. "The speed of sound and isentropic bulk modulus of biodiesel at 21C from atmospheric pressure to 35 MPa". In: *Journal of the American Oil Chemists' Society* 77.3 (2000), pp. 285–289. ISSN: 1558-9331. DOI: 10.1007/s11746-000-0047-z.

[9] J. Wilbrand. "Notiz Über Trinitrotoluol. (German) [About Trinitrotoulene]". In: *Annalen der Chemie und Pharmacie* 128.2 (1863), pp. 178–179.

[10] C. Haüssermann. "Über die explosiven Eigenschaften des Trinitrotoluols. (German) [On the Explosive Properties of Trinitrotoluene]". In: *Angewandte Chemie* 4.17 (1891), pp. 505–536.

[11] A. Nobel. *Improvement in exploding nitro-glycerine*. US Patent RE5,800. Mar. 1874. URL: https://www.google.com/patents/USRE5800.

[12] *Nitroglycerine and Dynamite*. http://www.nobelprize.org/alfred_nobel/biographical/articles/life-work/nitrodyn.html Accessed: 2016-12-29.

[13] A. Nobel. *Improvement in gelatinated explosive compounds*. US Patent 175,735. Apr. 1876. URL: https://www.google.com/patents/US175735.

[14] L. Roland. *Grenade*. US Patent 1,126,871. 1915. URL: https://www.google.com/patents/US1126871.

[15] W. Mills. *Grenade and other like apparatus*. US Patent 1,178,092. Apr. 1916. URL: https://www.google.com/patents/US1178092.

[16] S. Spooner. "Aircraft and the War". In: *Flight* 45.1 (1914), pp. 905–906.

[17] P. Brickhill. *The Dam Busters*. PAN Books, 1954. ISBN: 0330236180.

[18] G. I. Taylor. "Analysis of the Explosion of a Long Cylindrical Bomb Detonated at One End". In: Cambridge At The University Press, 1963, pp. 277–286.

[19] G. I. Taylor. "The Fragmentation of Tubular Bombs". In: Cambridge At The University Press, 1963, pp. 387–390.

[20] W. W. Predebon, W. G. Smothers, and C. E. Anderson. *Missile Warhead Modeling: Computations and Experiments*. Tech. rep. Aberdeen Proving Ground, MD: USA Ballistic Research Laboratory, 1977, p. 55.

[21] R. W. Gurney. *The Initial Velocities of Fragments from Bombs, Shell, Grenades*. Tech. rep. BRL-405. Aberdeen Proving Ground, Maryland: Ballistic Research Laboratories, Sept. 1943.

[22] N. F. Mott. "Fragmentation of Shell Cases". In: *Proceedings of the Royal Society of London* 189.1018 (1947), pp. 300–308.

[23] G. Y. Huang, W. Li, and S. S. Feng. "Axial distribution of Fragment Velocities from cylindrical casing under explosive loading". In: *International Journal of Impact Engineering* (2015). ISSN: 0734743X. DOI: 10.1016/j.ijimpeng.2014.08.007.

[24] P. C. Souers et al. "Upgraded analytical model of the cylinder test". In: *Propellants, Explosives, Pyrotechnics* (2013). ISSN: 07213115. DOI: 10.1002/prep.201200192.

[25] J. T. Dehn. *Models of Explosively Driven Material*. Tech. rep. Aberdeen Proving Ground: Army Ballistic Research Lab, 1984, p. 78.

[26] Y. J. Charron. *Estimation of Velocity Distribution of Fragmenting Warheads Using a Modified Gurney Method*. Tech. rep. Air Force Institute of Technology, 1979, p. 113. DOI: ADA074759.

[27] M. D. Hutchinson. "With-Fracture Gurney Model to Estimate both Fragment and Blast Impulses". In: *Central European Journal of Energetic Materials* 7.2 (2010), pp. 175–186. ISSN: 1733-7178.

[28] B. A. Breech. "Extension of the Gurney Equations to Two Dimensions for a Cylindrical Charge". In: (2011).

[29] E. Hirsch. "Improved Gurney Formulas for Exploding Cylinders and Spheres using "Hard Core" Approximation". In: *Propellants, Explosives, and Pyrotechnics* 11.3 (1985), pp. 81–84.

[30] P. Elek, S. Jaramaz, and D. Micković. "Modeling of the Metal Cylinder Acceleration Under Explosive Loading". In: *Scientific Technical Review* 63.2 (2013), pp. 39–46.

[31] S. I. Jackson. *Scaled Cylinder Test Experiments with Insensitive PBX 9502 Explosive*. Tech. rep. Los Alamos, New Mexico: Los Alamos National Laboratory, 2014, p. 10.

[32] J. N. Johnson. "Ductile Fracture of Rapidly Expanding Rings". In: *Journal of Applied Mechanics* 50.3 (1983), pp. 593–600. DOI: 10.1115/1.3167096.

[33] R. R. Karpp and W. W. Predebon. *Calculation of Fragment Velocities from Naturally Fragmenting Munitions*. Tech. rep. Aberdeen Proving Ground: US Army Ballistic Research Laboratories, 1975, p. 31.

[34] J. J. Yoh et al. "Simulating thermal explosion of cyclotrimethylenetrinitramine-based explosives: Model comparison with experiment". In: *Journal of Applied Physics* 97.8 (2005). ISSN: 00218979. DOI: 10.1063/1.1863429.

[35] J. Wardell and J. Maienschein. "The Scaled Thermal Explosion Experiment". In: *Proceedings of the 12th International Detonation Symposium*. (San Diego, CA). July 2002, pp. 384–393.

[36] S. Sato, Y. Dobashi, and T. Yamamoto. "Controlling Simulated Explosions by Optimization and Prediction". In: *12th International Conference on Cumputer Aided Design and Computer Graphics*. 2011, pp. 296–301. DOI: 10.1109/CAD/Graphics.2011.78.

[37] G. I. Taylor. "Analysis of the Explosion of a Long Cylindrical Bomb Detonated at One End". In: *The Scientific Papers of Sir Geoffrey Ingram Taylor: Volume 3: Aerodynamics and the Mechanics of Projectiles and Explosions*. Ed. by G. K. Batchelor. Cambridge at the University Press, 1963. Chap. 30, pp. 277–286.

[38] R. W. Gurney. *The Initial Velocities of Fragments From Bombs, Shell, Grenades*. Tech. rep. Aberdeen Proving Ground, Maryland: Ballistic Research Laboratories, 1943, p. 11.

[39] G. I. Taylor. "The Fragmentation of Tubular Bombs". In: *The Scientific Papers of Sir Geoffrey Ingram Taylor: Volume 3: Aerodynamics and the Mechanics of Projectiles and Explosions*. Ed. by G. K. Batchelor. Cambridge at the University Press, 1963. Chap. 44, pp. 387–390.

[40] M. D. Salas. *The Curious Events Leading to the Theory of Shock Waves*. Tech. rep. Rome, Italy: 17th Shock Interaction Symposium, 2006, p. 20.

[41] G. I. Kerley. *The Linear Us-Up Relation in Shock-Wave Physics*. Tech. rep. Appomattox, VA: Kerley Technical Services, 2006, p. 21.

[42] R. Menikoff. *JWL Equation of State*. Tech. rep. Los Alamos National Laboratory, 2015, p. 18.

[43] *CTH Course Notes*. Tech. rep. Albuquerque, NM: Sandia National Laboratories, 2016.

[44] H. Grisaro and A. N. Dancygier. "Numerical study of velocity distribution of fragments caused by explosion of a cylindrical cased charge". In: *International Journal of Impact Engineering* 86 (2015), pp. 1–12. DOI: 10.1016/j.ijimpeng.2015.06.024.

[45] DoD Supercomputing Resource Center. *Cray XC40 (Excalibur) User Guide*. 2017. URL: https://www.arl.hpc.mil/docs/excaliburUserGuide.html (visited on 03/03/2017).

[46] S. Boyd and L. Vandenberghe. *Convex Optimization*. 1st. Cambridge, MA: Cambridge University Press, 2004, p. 727. ISBN: 0521833787.

[47] D. E. Finkel. "DIRECT Optimization Algorithm User Guide". In: (2003).

[48] H. H. Rosenbrock. "An Automatic Method for finding the Greatest or Least Value of a Function". In: *The Computer Journal* 3 (1960), pp. 175–184.

[49] M. Melanie. *An Introduction to Genetic Algorithms*. 5th. Cambridge, MA: MIT Press, 1998, p. 158. ISBN: 0-262-13316-4.

[50] N. I. Senaratna. *Genetic Algorithms: The Crossover-Mutation Debate*. Tech. rep. Colombo, Sri Lanka: University of Colombo, 2005, p. 22.

[51] O. Abdoun, J. Abouchabaka, and C. Tajani. "Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem". In: (2012).

[52] M. A. Meters. *Computer Simulation of Dynamic Phenomena*. Springer, 1999. ISBN: 3-540-63070-8.

[53] Y. Cengel and M. Boles. *Thermodynamics: An Engineering Approach*. McGraw Hill Education, 2015. ISBN: 978-0-07-339817-4.

[54] E. Harstad. *CTH Shock Physics*. 2017. URL: www.sandia.gov/CTH/ (visited on 03/02/2017).

[55] B. M. Dobratz and P. C. Crawford. *LLNL Explosives Handbook - Properties of Chemical Explosives and Explosive Simulants*. Tech. rep. Livermore, CA: University of California, 1985, p. 510.

[56] K. Xiangshao et al. "A numerical investigation on explosive fragmentation of metal casing using Smoothed Particle Hydrodynamic method". In: *Materials and Design* (2013). ISSN: 18734197. DOI: 10.1016/j.matdes.2013.04.041.

[57] M. S. Bola et al. "Expansion of Metallic Cylinders under Explosive Loading". In: *Defence Science Journal* 42.3 (1992), pp. 157–163.

[58] T. Hiroe et al. "Deformation and fragmentation behaviour of exploded metal cylinders and the effects of wall materials, configuration, explosive energy and

initiated locations". In: *International Journal of Impact Engineering* (2008). ISSN: 0734743X. DOI: 10.1016/j.ijimpeng.2008.07.002.

[59] G. K. Batchelor. *The Scientific Papers of Sir Geoffrey Ingram Taylor: Volume 3: Aerodynamics and the Mechanics of Projectiles and Explosions*. Cambridge At The University Press, 1963.

9. Appendix I - Mathematical Derivations

9.1 Gurney Equation

The Gurney equation is derived from a general energy balance at the moment before gas leakage occurs. This derivation assumes a linear velocity profile and constant density for the reactant gases. The variables of note are E, the specific internal energy of the explosive, C, the mass of the explosive charge, M, the mass of the case, V_0, the velocity of the case, a, the inner radius of the case at the point of fracture, and ρ_g, the density of the product gases.

$$\mathrm{PE} = \text{Wall KE} + \text{Gas KE} \qquad \text{general conservation of energy}$$

$$E\,C = \frac{1}{2}MV_0^2 + \int_0^a \frac{1}{2}\big(2\pi\rho_g r\big)V_{gas}^2\,\mathrm{d}r$$

$$E\,C = \frac{1}{2}MV_0^2 + \int_0^a \pi\rho_g r V_0^2\Big(\frac{r}{a}\Big)^2\,\mathrm{d}r \qquad \text{linear velocity profile assumption}$$

$$E\,C = \frac{1}{2}MV_0^2 + \pi\rho_g V_0^2\int_0^a \frac{r^3}{a^2}\,\mathrm{d}r \qquad \text{constant gas density assumption}$$

$$E\,C = \frac{1}{2}MV_0^2 + \pi\rho_g V_0^2\Big(\frac{a^2}{4}\Big) \qquad \text{resolve the integral}$$

$$2E = V_0^2\Big(\frac{M}{C} + \frac{1}{2}\frac{\pi\rho_g a^2}{C}\Big) \qquad \text{gather velocity terms and multiply by two}$$

Consider conservation of mass between the solid explosive and product gases:

$$C = M_{gas} = \pi a^2\rho_g$$

Substitute C into the conservation of energy equation,

$$\frac{V_0}{\sqrt{2E}} = \left(\frac{M}{C} + \frac{1}{2}\right)^{-\frac{1}{2}}$$

which results in the final equation derived by R.W. Gurney in his 1943 paper.[38].

9.2 Optimal Radius Derivation

The Gurney equation can be used to optimize the geometry of an exploding cylinder. First Equation 3 must be squared and multiplied by $\frac{1}{2}M$ to form kinetic energy. For this optimization the variable $g = \sqrt{2E}$. This results in

$$KE = \frac{M\,C\,g^2}{2M + C}$$

As the outer radius is increased the case mass, M, becomes much greater than the charge mass, C. This eventually results in the asymptote $KE = C\,g^2$ as $r_o \to \infty$, which is seen in the nondimensional Gurney solution plot.

The definitions of M and C can also be substituted into the kinetic energy equation to get in terms of inner and outer radius

$$KE = \pi g^2 \rho_m \rho_c \frac{(r_o^2 - r_i^2) r_i^2}{2\rho_m (r_o^2 - r_i^2) + r_i^2 \rho_c}$$

where ρ_c and ρ_e are the densities of the case and explosive, respectively.

It should be clear that the kinetic energy is zero when $r_i = 0$ and $r_i = r_o$. For any physical system it is also required that $r_o \geq r_i$ and $r_i \geq 0$. To maximize the kinetic energy, this function must be derived and set equal to zero. From the constraints the inner radius is bounded by $0 \leq r_i \leq r_o$ while the outer radius is bounded by $r_i \leq r_o \leq \infty$.

The kinetic energy versus outer radius curve for a constant r_i and variable r_o is positive definite and starts at $KE(r_o = r_i) = 0$. In contrast the kinetic energy

versus inner radius curve is bounded by $KE(r_i = 0) = KE(r_i = r_0) = 0$. For this reason the optimization only requires finding the critical points of the kinetic energy with respect to the inner radius. This is achieved by taking the partial derivative of kinetic energy with respect to the inner radius and setting it equal to zero.

$$KE = \frac{M\,C\,g^2}{2M + C} \qquad \text{Expression for kinetic energy}$$

$$\frac{\partial\, KE}{\partial\, r_i} = \frac{\partial}{\partial\, r_i}\left[\pi g^2 \rho_m \rho_c \frac{(r_o^2 - r_i^2) r_i^2}{2\rho_m (r_o^2 - r_i^2) + r_i^2 \rho_c}\right]$$

$$\frac{\partial}{\partial\, r_i}\left[\frac{(r_o^2 - r_i^2) r_i^2}{2\rho_m (r_o^2 - r_i^2) + r_i^2 \rho_c}\right] = 0 \qquad \text{Multiplying constants}$$

This can be simplified by applying the product rule and gathering like terms.

$$\frac{r_i^5(4\,\rho_c - 2\,\rho_e) + r_i^3(8\,\rho_c\, r_o^2) + r_i(4\,\rho_c\, r_o^4)}{r_i^4(\rho_e^2 - 4\, rho_c\, \rho_e + 4\,\rho_c^2) + r_i^2(-8\rho_c^2 + 4\,\rho_m\,\rho_c) r_o^2 + 4\rho_c^2 r_o^4} = 0$$

The numerator of this equation is the critical points of the kinetic energy, and the denominator gives additional constraints. One zero exists at $r_i = 0$, to solve for the rest define the variables,

$$X = r_i^2$$

$$Y = r_o^2$$

which results in,

$$X^2(4\rho_c - 2\rho_e) + XY(-8\rho_c) + Y^2(4\rho_c) = 0$$

The relationship $Y \geq X$ exists by definition, therefore some function $Y = f(X)$ should exist such that f is never negative. This function can be solved

for by using the quadratic formula.

$$Y = \frac{8\,\rho_c\,X \pm \sqrt{64\,\rho_c^2 - 4X^2\,(4\,\rho_c)(4\,\rho_c - 2\,\rho_e)}}{8\,\rho_c} \qquad \text{Quadratic Formula}$$

$$Y = X \pm \sqrt{X^2 \frac{32\,\rho_c\,\rho_e}{64\,\rho_c^2}} \qquad \text{Simplify}$$

$$\frac{Y}{X} = 1 \pm \sqrt{\frac{\rho_e}{1\,\rho_c}} \qquad \text{Collect like terms}$$

$$\frac{r_o^2}{r_i^2} = 1 \pm \sqrt{\frac{\rho_e}{2\,\rho_c}} \qquad \text{Substitute}$$

which results in the Equation 9 when the positive form is taken. From the physical constraint that $r_o \geq r_i$ the positive equation must be taken except when $r_o = r_i$.

10. Appendix II - Input Decks

10.1 1D Basic Model

For the 1D CTH model a similar input deck was developed to run the parametric studies and each optimization. The input deck for the parametric study is given below, with the changes for each simulation type following.

```
environment,
    tabular_data
            tabular_data_file = 'data_table.dat'

method,
    output silent
    multidim_parameter_study
        partitions = 19 19

variables,
    continuous_design = 2
        cdv_lower_bounds  0.00, 0.00
        cdv_upper_bounds  2.00, 6.00
        cdv_descriptor    't',  'ro'

interface,
    fork,
        asynchronus
        evaluation_concurrency = 31
        analysis_drivers = 'python analysis_driver.py'
        input_filter = 'python pre_processor.py'
        output_filter = 'python post_processor.py'
```

```
23          parameters_file = 'params.in'
24          results_file    = 'results.out'
25          copy_files = 'template_files/*'
26          work_directory named 'workdir'
27          directory_tag
28          directory_save
29          file_save
30
31  responses,
32      num_objective_functions = 1
33      no_hessieans
34      no_gradients
```

Only the method region (lines 5-8) must be changed to perform an optimization with the dividing rectangles or genetic algorithms.

```
1  method,
2      output silent
3      coliny_direct
4          max_iterations = 1000
5          convergence_tolerance = 1e-12
```

And for the genetic algorithm:

```
1  method,
2      output silent
3      soga
4          max_iterations = 1000
5          convergence_tolerance = 1e-12
```

The gradient descent algorithm requires changes to the method, responses, and vairables block block. CTH can not return the derivative of kinetic energy for each evaluation, so it is necessary for Dakota to numerically calculate a gradient. This behavior is defined in the responses section.

```
1  method,
```

```
2      output silent
3      speculative
4      conmin_frcg
5          max_iterations = 1000
6          convergence_tolerance = 1e-12
```

The command speculative on line 3 tells Dakota to calculate a gradient in each direction for every point. In some cases the gradient in each direction may not be required, but this can decrease the wall-clock time for some highly parallelized simulations. For the responses section:

```
1 responses,
2      no_hessians
3      numerical_gradients
4          method_source dakota
5          interval_type forward
6          fd_gradient_step_size = 1.0e-4
```

It is also required to tell Dakota the initial point for the gradient descent algorithm. This is achieved by adding the following line to the variables block between lines 11 and 12.

```
1          cdv_intial_point  1.00, 3.00
```

The CTH input deck is used to generate the geometry and boundary conditions for the problem. A templated input deck was developed, where the variables for each simulation were stored inside {Curly Braces.} These variables were substituted into the input deck by the preprocessor for each evaluation. The processing scripts can be found in Appendix III.

```
**Title Record**
2d explosive

**Control Record**
control
    mmp0
    tstop=35.0e-6
    ntbad=999999
endc

**Mesh Record*
mesh
block=1 geom=1dc type=e
    x=0.0
        x1 n=2000 w=20.0 ratio=1.0
            * one cell every 0.1cm
    endx
    xactive 0.0 20.0
endmesh

*Spyplot and File Output*
spy
    Save("P, PM, M, VOLM, DENS, VOL,
                    T, VX, EK, DENSM");
    SaveTime(0, 0.5e-6);
    PlotTime(0, 0.5e-6);
    ImageFormat(2048, 1536);

    define main() {
    DataOut("allout_", "DENS", "VOLM+2",
            "M+2", "VX", "T", "P", "DENSM+2");

    XLimits(0,20);

    MatColor(1, WHEAT); %HE Color
    MatColor(2, BLACK); %Aluminum Color

    Image("V-",WHITE,BLACK);
      Window(0, 0, 0.85, 1);
      Label(sprintf("Velocity at 0.2e secs.",TIME));
      XBMirror(OFF);
      Plot2DMats();

      ColorMapRange(100, 2.0e5);
```

```
      ColorMapClipping(ON, OFF);
      Plot2D("VX");
      DrawColorMap("VX", 0.8, 0.22, 1, 0.66);
    EndImage;
    }
endspy

**Diatom Record**
diatom
    package 'explosive'
      material=1
      pressure=1.0e6
      temperature=0.02585
      insert box
        p1 0.0
        p2 {r_i}
      endi
    endpackage
*
    package 'wall'
      material=2
      pressure=1.0e6
      temperature=0.2585
      insert box
        p1 0.0
        p2 {r_o}
      endi
    endpackage
*
    package 'air'
      material=3
      pressure=1.0e6
      temperature=0.2585
      insert box
        p1 0.0
        p2 20.0
      endi
      endpackage
enddiatom

** EOS Record **
eos
    mat1 jwl tnt
    mat2 sesame aluminum
```

```
    mat3 sesame air
endeos

** Explosive Record **
heburn
    material=1
      detvel=693.0e3
      dp=0.0
      time=0.0
      radius=20
endhe

** Boundary Record **
boundary
    bhydro
      block=1
        bxbot=0
        bxtop=1
      endb
    endh
endb

** Convection Record **
convct
    convection=1
    interface=smyra
endcon

** Timestep Control **
mindt
    time=0.0 dt=1.0e-12
endmindt

** Edit Record **
edit
*
    shortt
      time=0.0 dt=1e-3
*
    longt
      time=0.0 dt=1.0
*
    restt
      time=0.0 dt=5e-3
```

```
133 *
```

```
134 endedit
```

For more information on the input deck see the CTH web page on the Sandia National Laboratories website[54].

10.2 1D Design Model

To deal with the multi-layer design problem a new variable needed to be introduced to the Dakota input deck, and the outer radius variable was switched to a constant state variable. The only update made from the basic 1D optimization occurred in the variables block, which is given below.

```
1     variables,
2       continuous_design = 2
3         cdv_lower_bounds  0.00, 0.00
4         cdv_upper_bounds  2.00, 2.00
5         cdv_descriptor    't1', 't2'
6       continuous_state = 1
7         csv_initial_state 5.00
8         csv_descriptor    'r_o'
```

A new layer of geometry was added to the CTH input deck by modifying the diatom (line 50) and the EOS records (line 82). The extra diatom package is:

```
1     package 'wall2'
2       material=4
3       pressure=1.0e6
4       temperature=0.2585
5       insert box
6         p1 0.0
7         p2 {r_2}
8       endi
9     endpackage
```

The wall2 package was placed on line 70 between the explosive and wall packages. The templated variable on line 67 (the original wall package) was also updated to

```
1 p2 {r_1}
```

which represents the inner radius plus the inner material thickness.

The extra EOS was added between lines 85 and 86 as

```
1     mat4 sesame iron
```

10.3 1D Job Submission

```
1  #!/bin/bash
2  #PBS -S /bin/bash
3  #PBS -l select=1:ncpus=32:mpiprocs=32
4  #PBS -l walltime=24:00:00
5  #PBS -q standard
6  #PBS -A #############
7  #PBS -N 1d_cth_parm
8  #PBS -j oe
9  #PBS -l plave=scatter:excl
10 ## Optional Directives -------------------------------
11 #PBS -m be
12 #PBS -M email@domain.sfx
13
14 ## Create a job-specific subdirectory to work in ---
15 echo ">> Job start at `date"
16 export RUNDIR=$WORKDIR/cth/1d/parametric-study
17 mkdir -p $RUNDIR
18 cp -r $PBS_O_WORKDIR/* $RUNDIR
19 cd $RUNDIR
```

```
20 ## Load job-specific modules -----------------------
21 module load cth/11.1
22 module load dakota
23 ## Job run commands --------------------------------
24 dakota input_dakota.in
25 echo ">> Job end at ‘date"
```

10.4 2D Free End Model

The 2D free end model used the same optimization variables and objective function as the basic 1D model, so no changes were made to the initial Dakota input deck.

To switch the domain to 2D the mesh record was updated with:

```
1  **Mesh Record**
2  mesh
3      block=1 geom=2dc type=e
4        x=0.0
5          x1 n=2000 w=20.0 ratio=1.0    *one cell every 0.1cm
6        endx
7        y0=0.0
8          y1 n=2000 w=20.0 ratio=1.0    *one cell every 0.1cm
9      endb
10 endmesh
```

As the velocity now has two dimensions the term "VX" was replaced everywhere with "VMAG" in the spyplot section (line 21).

The geometry of the explosive, wall, and air were then updated to two dimensions.

```
1  **Diatom Record**
2  diatom
3      package 'explosive'
4        material=1
```

```
5         pressure=1.0e6
6         temperature=0.02585
7         insert box
8           p1 0.0    2.0
9           p2 {r_1} 18.0
10        endi
11      endpackage
12  *
13      package 'explosive'
14        material=2
15        pressure=1.0e6
16        temperature=0.02585
17        insert box
18          p1 0.0    2.0
19          p2 {r_o} 18.0
20        endi
21      endpackage
22  *
23      package 'air'
24        material=3
25        pressure=1.0e6
26        temperature=0.02585
27        insert box
28          p1 0.0  20.0
29          p2 0.0  20.0
30        endi
31      endpackage
32  enddiatom
```

The detonation point of the explosive record was also updated to be at the bottom edge of the explosive:

```
1  **Explosive Record**
2  heburn
```

```
3      material=1
4        detvel=693.0e3
5        dp=0.0 2.0
6        time=0.0
7        radius=20
8  endhe
```

Finally, a top and bottom transmissive boundary condition were added to let the high pressure gas escape freely.

```
1  **Boundary Record**
2  boundary
3      bhydro
4        block=1
5          bxbot=0
6          bxtop=1
7          bybot=1
8          bytop=1
9        endb
10     endh
11 endb
```

10.5 2D Capped Model

As with the free end model, no changes were made to the Dakota input deck from the basic 1D model.

To add the end caps the control record and diatom records were modified to include the fixed material. The control record was updated to:

```
1  **Control Record**
2  control
3      mmp0
```

```
4      tstop=35.0e-6
5      ntbad=999999
6      rigid
7  endc
```

A new fixed material was also added to the diatom record.

```
1      package 'caps'
2        material=2
3        pressure=1.036
4        temperature=0.02585
5        rigid
6        insert box
7          p1 0.0    0.0
8          p2 {r_o}  2.0
9        endi
10       insert box
11         p1 0.0    0.0
12         p2 {r_o}  18.0
13       endi
14     endpackage
```

Fixed material is always put into the domain first, so it was specifically set to only fill the empty space above and below the free-end cylinder in the previous section.

10.6 2D Job Submission

As discussed earlier, an operating system feature of the Cray XC machines made it effectively impossible to run CTH and Dakota in parallel. The queue submission strategy was modified to deal with this, and the updated input decks are given below.

First, a shell script runs the command:

```
module load dakota
dakota input_dakota.in
```

Then the analysis driver called this bash script for each evaluation:

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -l select=1:ncpus=32:mpiprocs=32
#PBS -l walltime=2:00:00
#PBS -q standard
#PBS -A #############
#PBS -N 2d_eval
#PBS -j oe
#PBS -l plave=scatter:excl
## Optional Directives -------------------------------

## Create a job-specific subdirectory to work in ---
cd $PBS_O_WORKDIR/
## Load job-specific modules -----------------------
module load cth/11.1
## Job run commands --------------------------------
aprun -n 32 mpicth setup id="h"
```

After the submitted jobs have finished for every evaluation, the Dakota deck was rerun with the input filter removed and the analysis driver changed to

```
        analysis_drivers = 'python post_processor.py'
```

which postprocessed every complete evaluation and compiled the results into a data file.

11. Appendix III - Processing Scripts

11.1 Analysis Driver

The analysis driver is a python script called by Dakota to substitute the optimization variables into the input deck and run the analysis. The general analysis driver for KO is given below, and the changes for 1D and 2D CTH are given after.

```
#imports and dependancies
import sys
import os
import subprocess
#cmd line args come from sys.argv,
#they are [driver name, input file, output file]
paramsFileName = sys.argv[1]
resultsFileName = sys.argv[2]
#read dakota output and replace parameters
params = open(paramsFileName, 'r')
#read the first line for number of variables
line = params.readline()
tokens = line.split()
numVars = int(tokens[0])
#look for the variable names and values
varNames = []
values = []
for i in range(0,numVars):
    line = params.readline()
    tokens = line.split()
    varNames.append(tokens[1])
    values.append(float(tokens[0]))
#continue reading to the last line to get the #
otherNames = []
otherValus = []
for line in params:
    tokens = line.split() #store the extra name/value
    otherNames.append(tokens[1])
    otherValus.append(tokens[0])
#the final line has the evaluation number
evalNum = tokens[0]
#clean up our open resources!
params.close()
#KO is fixed width
for i in range(0,len(varNames)):
    varNames[i] = "{" + varNames[i] + "}"
    values[i] = str(values[i])
#name of the input file and simulation run command
inputName  = "ko.in"
runCommand = "./a.out"
#open up the input template file
inputFile = open(inputName, 'r')
fileData = ""
for line in inputFile:
```

```
45      fileData += line
46  for i in range(0, len(varNames)):
47      fileData = fileData.replace(varNames[i],
48                                  values[i])
49  inputFile.close()
50  #open, clear, and rewrite the input file
51  inputFile = open(inputName, 'w')
52  inputFile.write(fileData)
53  inputFile.close()
54  #run the simulation with any needed inputs
55  #arguments are
56    #['Program Name', 'arg1', 'arg2', etc...]
57  args = runCommand + ' ' + inputName
58  err = subprocess.call(args, shell=True)
```

In general the only lines that change are 43 and 44 where the run command and input deck name are defined. To run the 1D CTH simulations these lines are changed to:

```
1 inputName  = "setup id=h"
2 runCommand = "cth"
```

The 2D simulations required a second bash script to submit a parallel job. This is accomplished by again changing lines 43 and 44.

```
1 inputName  = "run_cth.bash"
2 runCommand = "qsub"
```

11.2 CTH Preprocessor

The CTH input deck uses the inner radius and outer radius to define geometry, but the Dakota input deck uses outer radius and thickness. To resolve this problem a preprocessing script was developed to modify the parameters file and insert these new variables.

```
# Python Preprocessor for the 1D Cylindee
# Performs calculations on variables
#     October 6, 2016import os
import sys
import StringIO
#cmd line args come from sys.argv
#they are [driver name, input file, output file]
paramsFileName = sys.argv[1]
#read the parameters file and copy it into a strg
params = open(paramsFileName)
paramText = params.read()
params.seek(0,0)
###pull all of the variables out of the params
#  line 1 contains the number of variables
line = params.readline()
tokens = line.split()
numVars = int(tokens[0])
#read the width of the vars line
lineWidth = len(line) - len(tokens[1] + " ")
#look for the variable names and values
varNames = []
values = []
for i in range(0,numVars):
    line = params.readline()
    tokens = line.split()
    varNames.append(tokens[1])
    values.append(float(tokens[0]))
#clean up our open resources!
params.close()
###   Calculate any new variables/values
###  This portion of pre-processing chages
newVars = []
newVals = []
#calculate inner radius
ro = values[varNames.index('r_o')]
t  = values[varNames.index('t')]
ri = ro - t

newVars.append('r_i')
newVals.append(ri)
## End of the variable definition section ##
###Write the new variables to the params###
#  Initial writing to a .tmp file
params = open(paramsFileName + '.tmp', 'w')
```

```
45 #increase the number of variables
46 #numVars is stored on the first line
47 strbuf = StringIO.StringIO(paramText)
48 line = strbuf.readline()
49 line=line.replace(str(numVars), str(numVars+1))
50 params.write(line)
51 #write the existing variables
52 for i in range(0, numVars):
53     line = strbuf.readline()
54     params.write(line)
55 #write the new variables
56 for i in range(0, len(newVars)):
57     line = str(newVals[i]).rjust(lineWidth-1) +
58     " " + newVars[i] + "\n"
59     params.write(line)
60 #write the rest of the file
61 for line in strbuf:
62     params.write(line)
63 #atomicly write the parameters file
64 os.rename(paramsFileName + '.tmp', paramsFileName)
65 params.close()
```

The only lines that should ever need to change in the preprocessor are 35-41. This is where new variables, in this case the inner radius, are defined. This region can was changed for the design problem to instead define the inner, outer, and middle radii for the design problem.

```
1  ###  This portion of pre-processing chages per script
2  newVars = []
3  newVals = []
4  #calculate inner and middle
5  ro = values[varNames.index('r_o')]
6  t1  = values[varNames.index('t1')]
7  t2  = values[varNames.index('t2')]
8
9  ri = ro-t1-t2
10 r1 = ro-t1
11
12 newVars.append('r_i')
13 newVals.append(ri)
14 newVars.append('r_1')
15 newVals.append(r1)
16 ## End of the variable definition section ##
```

This flexible approach allows any compatible combination of CTH and Dakota variables to be used while changing only a few lines of code in a single file.

11.3 CTH Postprocessor

The postprocessing python file is given below.

```
# Python Postprocessor for the 1D Cylinder
# Converts the KO output of ke.dat
    #into a single KE value
#     October 6, 2016
import sys
import glob
import subprocess
#file that dakota reads
resultsFileName = sys.argv[2]
### Run the C postprocessor
#get the filenames for each time step
for filename in glob.glob("allout_*"):
## arguments are
  #['Program Name', 'arg1', 'arg2', etc...]
    args = ["./processKE.out", filename]
    err = subprocess.call(args)
### KO process script output file name ###
data = open('ke_profile.dat', 'r')
### the post-processing specifics ###
ke = 0
numL = 0
for line in data:
    if line != "\n":
    tokens = line.split()
    ke_n = float(tokens[0])
    ke = max(ke, ke_n)
data.close()

f0 = -1*ke; ## this is the function evaluation value ##

#write the results file for dakota to read
results = open(resultsFileName, 'w')
results.write(str(f0) + " f0")
results.close()
```

Line 13 specifies a compiled C++ program to actually read through the CTH output files with the name allout_########.dat. For the 1D model this script creates the text file 'ke_profile.dat' which contains a list of kinetic energies ordered by time. This file is used on lines 20-24 to calculate the objective function. The C++ file can be found in the C++ Processing Script section.

In the 2D case the C++ processor created a copy of the allout data file with the extension '_profile'. This additional file contained the total kinetic energy of the wall as a function of axial position. Due to the parallel operation of CTH these allout files split the x axis into four domains which needed to be recombined. Lines 18-25 were modified to calculate the kinetic energy as a function of Y for each time step, then the maximum value from each was taken as the objective function value.

```
nProcs = 32
ke_values = []
for n in range(0, 32/4): #x is 4 files
    str1 = "allout_*." + str((n*4)+0)
                    + ".dat_profile"
    str2 = "allout_*." + str((n*4)+1)
                    + ".dat_profile"
    str3 = "allout_*." + str((n*4)+2)
                    + ".dat_profile"
    str4 = "allout_*." + str((n*4)+3)
                    + ".dat_profile"
    list1 = sorted(glob.glob(str1))
    list2 = sorted(glob.glob(str2))
    list3 = sorted(glob.glob(str3))
    list4 = sorted(glob.glob(str4))
    #instantiate the array for first iteration
    if n == 0:
        ke_values = [0]*len(list1)
    #loop through each file in time
    for i in range(0, len(list1)):
        ke_x = 0
        with open(list1[i], 'r') as f:
            for line in f:
                ke_x += float(line)
        with open(list2[i], 'r') as f:
            for line in f:
                ke_x += float(line)
        with open(list3[i], 'r') as f:
            for line in f:
                ke_x += float(line)
        with open(list4[i], 'r') as f:
            for line in f:
                ke_x += float(line)
        ke_values[i] = max(ke_maxes[i], ke_x)

max_ke = max(ke_values)

f0 = -1*max_ke  ##function evaluation value ##
```

The 20x20 domain was split into 32 equal chunks by CTH, one per processor, and in this case the radial axis was divided 4 times. For this reason an indexing variable ranged from 0 to 32/4 on line 3, and lines 4-7 count up by four to cover the whole range. The loop on line 16 then moves through every time step and calculates the kinetic energy at each. This eventually results in an array of the maximum energy values for each time step, and the overall maximum is taken on line 32.

For 1D simulations the 'ke_profile' data file also contains information on the inner radius of material. The initial postprocessing file was modified at one point to include an expansion ratio with the following code:

```
1  ri_o = 0
2  ke = 0
3
4  for line in data:
5      if line != "\n":
6        tokens = line.split()
7        ke_n = float(tokens[0])
8        ri   = float(tokens[1])
9        #set the initial inner radius
10       if (ri_o == 0):
11           ri_o = ri
12       if ri < ri_o * 2.2: #expansion ratio
13           ke = max(ke, ke_n);
14
15 data.close()
16
17 f0 = -1*ke  ## this is the function evaluation value ##
```

11.4 C++ Processing Scripts

```
#include <string>   //C++ strings
#include <iostream> //ifstream file read
#include <fstream>  //cout

int main(int argc, char** argv) {
  //vars to read
  float x, dx, dens, v2, m2, vx, T, P, dens2
  float toal_ke;
  std::string filename = "";
  //constants
  float PI = 3.14159265359;
  //calcualted values
  float ri = 0;

  if (argc >= 2) {
    filename = argv[1];
  } else {
    std::cout << No Args Passed in!
        Filename Required << std::endl;
    return 0;
  }

  //open the file
  std::ifstream file(filename.c_str());
  if (file.fail()) {
    std::cout << "Error! Could not open file!"
        << std::endl;
    return 0;
  }
  //gobble the first line of headers
  std::string tmp;
  getline(file, tmp);
  //calculate total KE of solid (material 2)
  while(file>>x>>dx>>dens>>v2>>m2>>vx>>T>>P>>dens2) {
    total_ke += (2*PI*(m2*x)*vx*vx)/2.0;
    if (ri==0 && m2>0) {
      ri = x;
    }
  }

  //open and append the KE file
  std::ofstream of("ke_profile.dat", std::ofstream::app);
  if (!of) {
    std::cout << "Could not open ke_profile.dat"
```

```
45          << std::endl;
46      return 0;
47    }
48    of << total_ke << " " << ri << "\n";
49    of.close();
50
51    return 0;
52  }
```

For the case with two materials the kinetic energy was updated to use a 'mass fraction' variable which was calculated as the aluminum+iron mass in each cell.

For the 2D simulation the C++ processor was significantly altered to generate a kinetic energy profile for each time step. A new section was added before the file read loop between lines 30 and 31 to grab the current x0 position in the divided mesh.

```
1 file>>x>>y>>dx>>dy>>dens>>v2>>m2>>vm>>T>>P>>dens2
2 x0 = x;
3 mf2 = m2;
4 total_ke += (2*PI*(mf*x)*vm*vm)/2.0;
```

Within the file read loop the kinetic energy was stored as

```
1 if (x == x0) {
2     ke_profile.push_bac(total_ke);
3     total_ke = 0;
4 }
5 mf = m2;
6 total_ke += (2*PI*(mf*x)*vm*vm)/2.0;
```

where 'ke_profile' is of the type

```
1 std::vector<float>()
```

To record the kinetic energy profile an output file was created and the profile was stored in a new file with the suffix '_profile' added to differentiate it from the original.

```
max_ke = 0;
std::ofstream kep(std::string(
                filename+"_profile").c_str(),
                std::ios::out | std::ios::trunc);
if(!kep) {
    std::cout << "ERROR! Could not open file"
              << std::endl;
    return 0
}
for (std::size_t i=0; i<ke_profile.size();i++) {
    kep << ke_profile[i] << "\n"l
}
return 0;
```

12. Appendix IV - Visualization Scripts

12.1 Dakota Visualization

A Matlab script was developed to read optimization results from Dakota. The file "ReadDakota" was developed by opening the data table file into Matlab's import editor and selecting the 'import section → generate function' option.

```
%visualizes the results of the 1d simulations
%DIRECT, Multistart gradient descent, SOGA
% and parametric study results
%% Part 1 - load data
clear; close all; clc;

ANIMATE = false;

%constants for the explosive
g = 2.44 * 1000; %[m/s]gurney constant
rho_e = 1.630 * 1000; %[kg/m3]explosive density
rho_c = 2.6993 * 1000;%[kg/m3]case density

%pull out the simulation results
[d_t, d_ro, d_ke] = ...
        ReadDakotaData('direct/data_table.dat');
[g_t, g_ro, g_ke] = ...
      ReadDakotaData('gradient/data_table.dat');
[s_t, s_ro, s_ke] = ...
        ReadDakotaData('soga/data_table.dat');
[t, ro, ke] = ...
    ReadDakotaData('parametric' ... '
                /data_table_aluminum.dat');

%convert KE to J -> MJ
d_ke = d_ke * 1e-7 * 1e-6;
g_ke = g_ke * 1e-7 * 1e-6;
s_ke = s_ke * 1e-7 * 1e-6;
ke   = ke   * 1e-7 * 1e-6;
% max values:
d_i = find(d_ke == max(d_ke));
g_i = find(g_ke == max(g_ke));
s_i = find(s_ke == max(s_ke));
p_i = find(ke == max(ke));

fprintf('Maximum Values:\n')
fprintf('DIRECT METHOD:----------------\n')
fprintf('\t KE=%g\t r=%g\t t=%g\n', ...
    d_ke(d_i), d_ro(d_i), d_t(d_i));
fprintf('GRADIENT METHOD:---------------\n')
fprintf('\t KE=%g\t r=%g\t t=%g\n', ...
    g_ke(g_i), g_ro(g_i), g_t(g_i));
fprintf('SOGA METHOD:-------------------\n')
fprintf('\t KE=%g\t r=%g\t t=%g\n', ...
```

```
45      s_ke(s_i), s_ro(s_i), s_t(s_i));
46  fprintf('PARAMETRIC METHOD:--------------\n')
47  fprintf('\t KE=%g\t r=%g\t t=%g\n',
48      ...ke(p_i), ro(p_i), t(p_i));
49  %save some vars for later
50  tp = t; rop = ro; kep = ke;
51  %line width var for visualization
52  lw = 3;
53
54  %parametric surface
55  x = linspace(min(t), max(t), ...
56              sqrt(length(t)));
57  y = linspace(min(ro), max(ro), ...
58              sqrt(length(ro)));
59  [xx, yy] = meshgrid(x, y);
60  %2d interpolation
61  zz = griddata(t,ro,ke,xx,yy, 'v4');
62  %% Part 2 - visualization
63  figure(1)
64  surf(xx, yy, zz);
65  xlabel('Thickness [cm]');
66  ylabel('Outer Radius [cm]');
67  zlabel('Kinetic Energy [MJ]')
68  SetPlotStyle();
69
70  figure(10)
71  contourf(xx, yy, zz, 10);
72  hold on;
73  plot([0 t(p_i)], [0, ro(p_i)], 'k', ...
74   'linewidth', lw);
75  xlabel('Thickness [cm]');
76  ylabel('Outer Radius [cm]');
77  zlabel('Kinetic Energy [MJ]')
78  SetPlotStyle();
79  grid off;
80  %get the resolution of the mesh
81  %on the thickness (x) axis
82  tu = unique(t);
83  resT = tu(2) - tu(1);
84
85  fprintf('Slope:\n')
86  fprintf('Min: %g \t Max: %g \t ...
87          Avg: %g \t\n', ...
88  ro(p_i)/(t(p_i)+resT), ...
```

```
ro(p_i)/(t(p_i)-resT), ro(p_i)/(t(p_i)));

%% DIRECT results
figure(2)
hold on;
contourf(xx, yy, zz, 10);
plot(d_t, d_ro, 'ok', 'markerfacecolor', 'k');
xlabel('Thickness [cm]');
ylabel('Radius [cm]');
%plot(t_cth(ind), ro_cth(ind), '.k', ...
                'linewidth', 2);
SetPlotStyle();
grid off;

figure(20)
surf(xx, yy, zz);
hold on;
plot3(d_t, d_ro, d_ke*1.01, 'ko', ...
            'markerfacecolor', 'k');
xlabel('Thickness [cm]');
ylabel('Radius [cm]');
zlabel('Kinetic Energy [MJ]');
SetPlotStyle();
%% animation
if ANIMATE
    figure(200); clf; hold on;
    contourf(xx, yy, zz, 10);
    xlabel('Thickness [cm]');
    ylabel('Radius [cm]');
    SetPlotStyle();
    grid off;
    clr = bone(length(d_t));
    for i = 1:length(d_t)
        plot(d_t(i), d_ro(i), ...
        'ok', 'markerfacecolor', clr(i,:));
        saveas(gcf, ['direct-images/' ....
                    num2str(i) '.png']);
        pause(0.1);
    end
end
%% GENETIC ALGORITHM
figure(3)
hold on;
contourf(xx, yy, zz, 10);
```

```
plot(s_t, s_ro, 'ok', 'markerfacecolor', 'k');
xlabel('Thickness [cm]');
ylabel('Radius [cm]');
SetPlotStyle();

figure(30)
surf(xx, yy, zz);
hold on;
plot3(s_t, s_ro, s_ke, 'ko', ...
                'markerfacecolor', 'k');
xlabel('Thickness [cm]');
ylabel('Radius [cm]');
zlabel('Kinetic Energy [MJ]');
SetPlotStyle();

%% animation
if ANIMATE
    figure(300); clf; hold on;
    contourf(xx, yy, zz, 10);
    xlabel('Thickness [cm]');
    ylabel('Radius [cm]');
    SetPlotStyle();
    clr = bone(length(s_t));

    for i = 1:50:length(s_t) - 50
        plot(s_t(i:i+50), s_ro(i:i+50), ...
        'ok', 'markerfacecolor', clr(i,:));
        saveas(gcf, ['soga-images/' ...
                        num2str(i) '.png']);
        pause(0.1);
    end
end
%% GRADIENT ALGORITHM
figure(4)
hold on;
contourf(xx, yy, zz,10);
plot(g_t, g_ro, '-ok', 'markerfacecolor',...
'k', 'linewidth', 2);
xlabel('Thickness [cm]');
ylabel('Radius [cm]');
SetPlotStyle();
grid off;

figure(40)
```

```
surf(xx, yy, zz);
hold on;
plot3(g_t, g_ro, g_ke*1.01, 'ko',...
'markerfacecolor', 'k');
xlabel('Thickness [cm]');
ylabel('Radius [cm]');
zlabel('Kinetic Energy [MJ]');
SetPlotStyle();

%% animation
if ANIMATE
    figure(200); clf; hold on;
    contourf(xx, yy, zz, 10);
    xlabel('Thickness [cm]');
    ylabel('Radius [cm]');
    SetPlotStyle();
    clr = bone(length(g_t));
    for i = 1:length(g_t)-1
        clf; hold on;
        contourf(xx, yy, zz, 10);
        plot(g_t(1:i), g_ro(1:i), '-k', ...
                            'linewidth', 2);
        for j = 1:i
            plot(g_t(j), g_ro(j),...
            'ok', 'markerfacecolor',  clr(j,:));
        end
        xlabel('Thickness [cm]');
        ylabel('Radius [cm]');
        SetPlotStyle();
        saveas(gcf, ['gradient-images/'...
                        num2str(i) '.png']);
        pause(0.1);
    end
end

%% Part 6-Nondimensional plot colored by thickness
%grab vars from parametric study
% & convert to standard units
tn = tp / 100; ron = rop / 100; ken = kep;

width = find(ron(1) == ron, 1, 'last');
depth = length(ron) / width;

tn  = reshape(tn, [width, depth])';
```

```
ron = reshape(ron,[width, depth])';
ken = reshape(ken,[width, depth])' * 1e6;

radii = unique(ron(:,1));

%remove values where t >= ro
b = find(tn >= ron);
tn(b) = 0; ron(b) = 0; ken(b) = 0;
Cn = pi*(ron-tn).^2 * rho_e;

lim = 5; %"small" limit
linecolors = jet(length(tn));

figure('position', [500, 50, 800, 800]);
clf; hold on;
%plot this in a for loop for color
hold on;
for i = 1:length(tn)
    marker = 'o';
    if i <= lim
        marker = 's';
    end
    plot(tn(:,i)./ron(:,i), ...
            2*ken(:,i)./(Cn(:,i)*g^2), ...
    marker, 'markerfacecolor', linecolors(i,:),...
            'color', 'k', 'markersize', 10);
end

xlabel('Thickness / Radius');
ylabel('MV^2/Cg^2');

%set the legend
legendCell = {};
tu = unique(tn);

for i = 1:length(tu)
    legendCell = {legendCell{:}, ...
    ['t = ' num2str(tu(i)*100, '%2.2f') ' cm']};
end

legend(legendCell);
SetPlotStyle();
```

12.2 Curve Fitting

A matlab script was also developed to visualize and fit Equation 12 to the CTH data using the 'cftool' command.

```
%Fits a curve to the CTH parametric study data
%of the form E = A (1-e^-{k1*x}) * (e^-{k2*x})
clear; close all; clc;
[t,r,ke]=ReadDakotaData('data_table_copper.dat');
%scale variables
ke = -1*ke;
%split into 2d arrays
nRows = length(find(r == r(1)));
nCols = length(r) / nRows;
%delete t, r, ke that are non-physical
inval = find(t >= r);
t(inval) = NaN;
r(inval) = NaN;
ke(inval)= NaN;
%finish reshaping
t = reshape(t, [nRows, nCols]);
r = reshape(r, [nRows, nCols]);
ke = reshape(ke, [nRows, nCols]);

%%
%x variavle for curve fit
x_f = linspace(0, 1);
%calculate nondimensional X and Y, and fit!
x = t./r;
y = 2*ke./(C*g^2);
x = reshape(x, [1, numel(x)]);
y = reshape(y, [1, numel(y)]);
%don't try to fit NaN
dontfit = find(isnan(x) | isnan(y));
x(dontfit) = [];
y(dontfit) = [];
%the function we want to fit this to is
% E = A*exp(-k1x)*(1-exp(k2x))
figure(); clf; hold on;
ool with a linear term and no weights (copper)
a = 1.234;
k1 = 0.7316;
k2 = 34.7;
d  = 1;
%equation to fit
y_f = a * exp(-k1*x_f) ...
        .* (1 - exp(-k2*x_f)).*(1-x_f).^d;
%plot CTH results
plot(x, y, 'ok', ...
```

```
'markerfacecolor', [0.9100    0.4100    0.1700],
'markersize', 8);
%plot curve fit
plot(x_f, y_f, '-.k', 'linewidth', 4);

xlabel('Nondimensional Size');
ylabel('Nondimensional Energy');
legend('CTH', 'Curve Fit');
SetPlotStyle();
```

12.3 Gurney Equation

Matlab was used to visualize the dimensional and nondimensional forms of the Gurney equation. A sample of this code is given below.

```
%% Gurney Results
%material constants
g_c = 2.44; %gurney constants in [mm/us]
rho_e = 1.630; %explosive density [g/cc]
rho_c = 2.6993; %case density       [g/cc]
%convert to standard units
g_c = g_c * 1000;       %% mm/us to m/s
rho_e = rho_e * 1000;   %% g/cc to kg/m3
rho_c = rho_c * 1000;   %% g/cc to kg/m3
%generate mesh of t, ro pairs
t = linspace(0, 2, 20)/100; %% m
ro= linspace(0, 6, 20)/100; %% m
[t, ro] = meshgrid(t, ro);
ri = ro - t;
%calculate charge and case mass
 %charge mass [kg/m]
C = pi * ri.^2 * rho_e;
%case mass [kg/m]
M = pi * (ro.^2 - ri.^2) * rho_c;
%calculate velocity
V = g_c./sqrt( (M./C) + 0.5); % velocity in m/s
V(1,1) = 0;
%remove values where M or C are less than zero
M(t>ro) = 0;
%calculate KE
ke = 0.5 * M.*V.*V;
%plot dimensional values
figure(3);
surf(t*100, ro*100, ke/1e6);
xlabel('Thickness [cm]')
ylabel('Outer Radius [cm]')
zlabel('Kinetic Energy [MJ]');
axis equal square
SetPlotStyle();
%plot nondimensional values
t(t>ro) = 0;
figure(4);
plot(t./ro, 2*ke./(C*g_c^2), 'ok', ...
'markerfacecolor', 'k', 'markersize', 8);
xlabel('t/r');
ylabel('2KE / Cg^2');
SetPlotStyle();
```

12.4 Figure Formatting

A general Matlab function was developed to create a consistent style for each figure. This function 'SetPlotStyle' can be seen in all of the previous code snippets.

```
function SetPlotStyle(fs)
%Styles matlab generated plots to be consistent
%Logan Beaver, 2/28/2017
%set figure size to a default of 14 if none is specified
if ~exist('fs','var')
    fs=14;
end
%Set font size and type to all elements
set(findall(gca,'type','text'),'FontSize',fs+2, ...
    'fontName','Times New Roman')
set(gca,'FontName','Times New Roman', 'fontsize', fs);
%turn on the grid and set the axes
grid on
axis equal square
set(gca,'XMinorTick','on','YMinorTick','on')
%centers the plot in the figure window
fig = gcf;
fig.PaperPositionMode = 'auto';
fig_pos = fig.PaperPosition;
fig.PaperSize = [fig_pos(3) fig_pos(4)];
%turns on the black box around the the window
box on;
end
```