Automated Shaped Charge Design: Applying Dakota Optimization to CTH Kinetic Energy Results

Sebastian Arcangelo Konewko
Marquette University

AUTOMATED SHAPED CHARGE DESIGN: APPLYING DAKOTA OPTIMIZATION TO CTH KINETIC ENERGY RESULTS

By
Sebastian Arcangelo Konewko

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 2019

ABSTRACT
AUTOMATED SHAPED CHARGE DESIGN:
APPLYING DAKOTA OPTIMIZATION TO CTH KINETIC ENERGY RESULTS

Sebastian Arcangelo Konewko

Marquette University, 2019

Advances in computational power present an opportunity to further optimize the design of an engineered energetic system. This work presents the application of a proposed optimization scheme which combines the shock-physics hydrocode CTH with the DAKOTA optimization package to automate shaped-charge jet design. The formation of an explosively driven hypervelocity jet is highly dependent on the original shaped charge liner geometry. By parameterizing this geometry, and by developing a characteristic objective function from CTH simulations, a process can be established where the Dakota code iteratively builds an optimal shaped charge.

This work attempts to use this methodology to reproduce a reference geometry. This is done by characterizing the liner geometry with two parabolas and post-processing an objective function from the kinetic energy profile of the resulting jet. Multi-dimensional parameter studies, gradient optimizations and genetic algorithms are used to probe the parameter space.

ACKNOWLEDGMENTS

Sebastian Arcangelo Konewko

I would like to thank my committee members Dr. John Moore, Dr. Geremy Kleiser and especially my advisor Dr. John Borg for taking the time to mentor and patiently support me. Additionally, I would like to thank the Air Force Research Laboratory and Dr. Geremy Kleiser again, for funding this research.

This work represents the support from family and friends (especially those in the Shock Physics lab) that I have received my whole life.

Finally, a special thanks to my parents, Mark and Simonetta Konewko, and my brother, Leonardo Konewko, who have always managed to lift my spirits through the difficult times. This work could not have been accomplished without their presence in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The advent of computational power has greatly increased the potential for optimization design processes. In the field of explosive design, this can be a very useful tool. Traditionally, explosives have been designed through an iterative experimental process. This is particularly time-consuming for shaped charge design as slight changes in liner geometry can have extensive impacts on the hypervelocity jets.

The shaped charge's hypervelocity jet is highly dependent on the initial liner geometry. Extensive experimental research has been done characterizing different geometries and their corresponding jets (P.Y. Chanteret, 1984) (U.S. Army Materiel Command). Additionally, hydrocodes are being increasingly used to simulate different shaped charge geometries to compare simulated and experimental results (Wickert, 2013)(Woodley, 2016) (Coddet, 2015). However, seeing as this work has industrial and military applications, the intellectual property regarding the design of modern shaped charge geometries is a closely held secret.

Today, the presence of high performance computers allows one the ability to iteratively simulate different shaped charge geometries and compare their results. Therefore, the ability to couple an optimization package, such as Dakota, to CTH, a hydrocode, has extreme potential. In a practical setting, one can choose a reference metric produced by the CTH simulation and optimize it by varying the geometry of the liner. To this end, this study is unique in that the optimization target is not the maximization of a property of the shaped charge jet. As a proof of concept, the goal of the study is to reproduce a reference shaped charge geometry. To do this, one must architect an objective function such that its global minima uniquely maps to a set of reference

parameters. This process and its mathematic implications is discussed in much more detail further in the body of work.

When conducting an optimization one has, almost literally, an infinite set of parameters that can constrain a problem. To reduce computational load, this work specifically focuses on the geometric implications of changing a shaped charge liner.

Equally numerous are the number of metrics one can use to assess the validity of the iterative parameters. Advances in optimization have produced techniques which can evaluate the legitimacy of a design based on multiple objective functions. Once again, to simplify the optimization problem, this study forces the assessment of parameters on one objective function.

In concurrence with the previous constraint, this broadly defines the system in question as:

$$OF = f(x_1, x_2, \dots, x_n)$$

where x_n are variables that represent the geometry of a liner and *OF* is the objective function.

In many ways, this study can be thought of the continuation of work done by Logan Beaver (Beaver, 2017). His optimization study concerned the optimization of a cylindrical explosive charge and involved coupling Dakota to a hydrocode.

This problem was unique in that, due to the geometry of the system, it was possible to reduce the optimization to a 1D problem. In parameter space, this corresponded to two dimensions, the inner and outer radius of the liner. In addition, analytic solutions exist that describe the behavior of the cylindrical charge. Solutions outlined by Gurney provide a surface relating the two liner values to the kinetic energy that the charge can produce.

This work differs from that study mainly in the level of pre and post processing needed to accurately simulate the energetic material system. First a shaped charge is inherently a multi-parameter object. The simplest liner geometry, a “V” shape with constant thickness, requires at least three parameters to create it (Baker, 2011). These include the thickness, the slope and the height of the liner. Therefore, when parameterizing the geometry of the charge, an additional pre-processing step is needed where a “geometry script” takes the parameters to be optimized and outputs a set of points that define the shaped charge liner.

Second, the creation of an objective function is more involved. One can look to many different metrics to assess the legitimacy of a liner geometry. Therefore, additional post-processing needs to be done on the results of the CTH simulation.

Finally, Beaver allows for the variation of materials in his study. It would be very interesting to see the relationship between material properties, specifically how a material reacts differently to principle versus shear stress, influences the design of the shaped charge liner. However, this introduces too much non-linearity to the problem and would shift the focus from a geometric optimization to a study on material properties. Therefore, it is outside the scope of this work.

2 COMPUTATIONAL ENVIRONMENT

2.1 HPC HARDWARE

The optimization study was designed to be run on a United States High Performance Computer (HPC). Specifically, the Topaz machine was used to run the optimization studies (SGI ICE X (TOPAZ) USER GUIDE). It is built with a total of 3,456 standard computing nodes each with two 2.3-GHz Intel Xeon Haswell 18-core processors (36 cores) and 128 GBytes of DDR4 memory. As its operating system, Topaz uses SGI's Performance Suite. This is a combination of Linux and SGI-specific tools. More information can be found in SGI ICE X (TOPAZ) USER GUIDE.

Optimization jobs were submitted on these computation nodes and stored in work directories. Data was post-processed in these directories and visualized locally.

2.2 COMPUTATIONAL TOOLS

This work relies on the coupling of the CTH and Dakota projects.

CTH is a shockphysics based hydrocode developed out of Sandia Natinal Laboratories and is used to simulate the shaped charge event (McGlaun, 1990). It is primarily an Eulerian code with the exeption of an intermediate Lagrangian step where cells deform to track material motion. CTH conatins one, two and three dimensional rectiliner, cylindrical and spherical meshes. CTH uses a second order convection scheme to advect material, flux thermodynamic quantities and material properties through cells. The Jones-Wilkins-Lee and other equatons of state are availible to model the reaction products of explosives.In addition, the CTH employs a variable time step determined by

the Courant stability criterion (Simon G. Edwins, 2002) (Crawford). In a two dimenional calculation, a saftey factor of 0.6 is multiplied by the minimal allowed time step.

Dakota is an optimization suite also based out of Sandia National Laboratories (Adams). The Dakota software provides a flexible environment for the user to explot when optimizing systems. One can “loosly couple” Dakota to any input and output system and use the various optimization schemes to search for optima. The Dakota toolkit includes gradient and non-gradient based schemes, stochastic expansion methods, surrogate optimization and others.

Dakota and CTH were “loosly coupled” so that Dakota fed parameters controling liner geometry into a “black-box function” and optimized the resulting objective function. CTH was used to simulate the various shaped charge geometries. The shaped charge jet kinetic energy was parsed from the simulation, characterized and developed into the final objective function.

To maximize computational efficency, ideally, Dakota and CTH would both be run in parallel together. However this is currently not possible on an HPC. Due to quirks of PBS on Topaz, if Dakota launches a parallel CTH job, this job will not run on the computer nodes that are already reserved by the Dakota launch. Instead, the parallel CTH job is submitted to the PBS queue were more time would be spent waiting for a job to launch. This is a problem which can be remeiated however it lies outisde the scope of this work.

Therefore, for this study, a parallel Dakota is optimization is launched. Subsequently, Dakota will spawn concurrent serial CTH simulations. This prevents the user from fully exploiting the node.

All shell scripts were written in bash. Any script that involved mathematic operations was written in python3 (version 3.6.7). Numpy (version 1.14.2), Scipy (version 1.1.0) and Matplotlib (version 1.4.3) modules were used to support the python libraries. The Topaz GCC version of these codes was used.

2.3 COMPUTATIONAL FRAMEWORK

The Dakota/CTH coupling was executed using several scripts. These can be classified into three groups: governing scripts, executables and template scripts. Governing scripts are scripts that are only run once and meant to control the entire optimization environment. Executables are scripts that are run every time a new simulation is needed and finally, template scripts are shell scripts that are modified every time a new simulation is launched.

Dakota/CTH Scripts	
Governing Scripts	**Rundak**: This bash script submits the optimization job to the PBS queue.
	Dakota.in: This is the Dakota input script. It contains information about the optimization scheme and the parameters that are being controlled.
	Cth_simulation.sh: This script controls the interface between Dakota and CTH. It can be thought of as a "black-box" which takes inputs from Dakota to CTH and outputs from CTH to Dakota.
Executables	**Cth.processor.sh**: This bash script controls and launches each CTH simulation. From it, geometry is updated, CTH is launched and post-processing is performed.
	CurveFit.py: This python script post-processes the CTH results and produces an objective function that Dakota can optimize.
Template Scripts	**Geometry.template.py**: This python script produces the geometry that is unique to every CTH simulation. Dakota directly changes the parameters in this script.
	Cth.template: This is the CTH input script. Every simulation, new geometries are inserted in this script.

Table 1: A list of scripts and their functions used in the optimization study

As previously explained, a “black-box” style interface was used to couple Dakota’s optimization toolkit with CTH’s simulations. For this technique, Dakota interacts with a “black-box” function by changing allotted parameters. These variables control the CTH simulation. Once the simulation has concluded, post-processing is done on the results and an objective function is found. This is fed back into the Dakota optimization and the parameters are changed according to whichever optimization scheme is used. This framework is illustrated below.

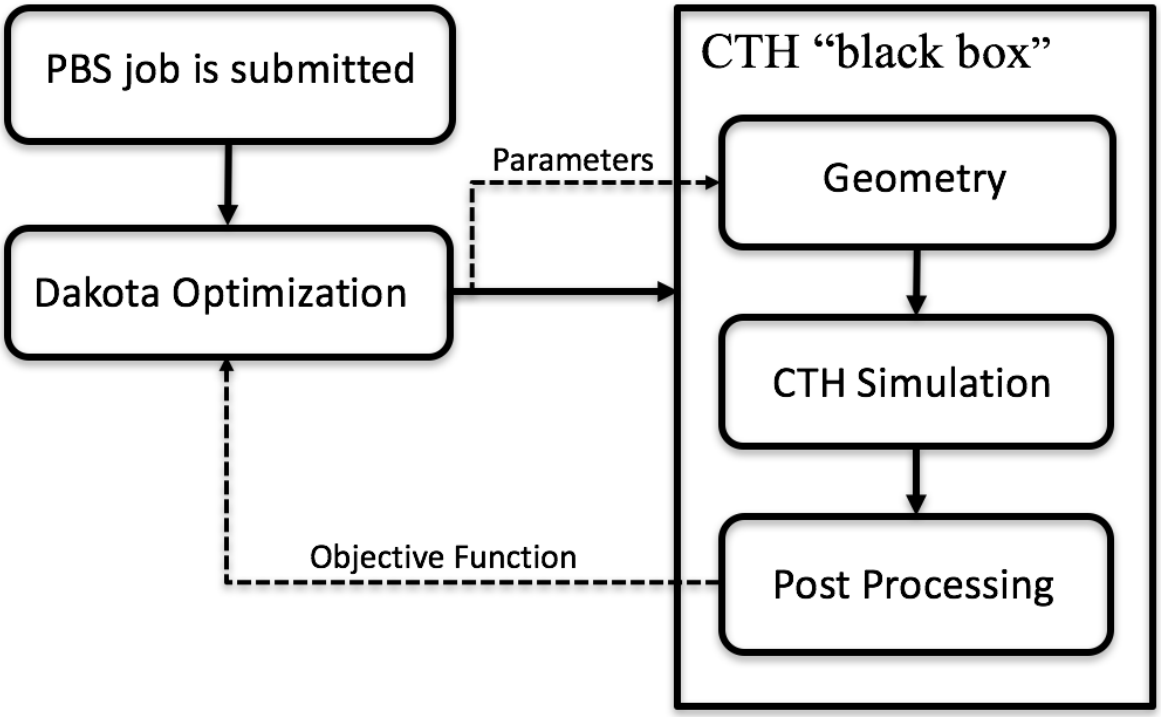


Figure 1: A schematic of the optimization study

Depending on the optimization scheme, different levels of parallelization could be accomplished. This depends on how strict the order of simulations must be. For example, a local gradient method calculates one “absolute point” and the derivative of each

dimension around it. Therefore, it can only run 2n+1 parallel simulations (where n is the number of dimensions). On the other hand, a gradient-free, evolutionary algorithm requires a large sample size each iteration for its optimization scheme. Therefore, the parallelization is equal to the iteration's sample size. A basic scheme, such as a parameter study, does not make informed decisions based on past function evaluations. These types of schemes can run all CTH processes independently.

In theory, one could compile all of these processes coupling the codes in a more rigous way however this would not give the user much computational releif. This is because the vast amount of computational resourcees used in the optimization are dedicated to the various CTH simulations. Minimal post processing is needed to form the objective function and any calculations that Dakota performs are equally trivial. More information on how the scripts interact can be found in the Appendix.

3 OPTIMIZATION SCHEMES

Three optimization schemes were used to examine the parameter space of the shaped charge. The architecture of the Dakota schemes (and subsequent objective functions) is such that the optima is defined to be the minima of the objective function.

Many modern numerical optimization techniques exist [(Rao, 2009)]. This study exploits three main schemes.

The underlying assumption throughout all of this is that the objective function can explicitly be expressed as a function of the chosen parameters

$$OF = f(x_1, x_2, \dots, x_n)$$

3.1 MULTIDIMENSIONAL PARAMETER STUDY

The most basic of the optimization schemes, this technique involves partitioning the domain of the parameters and evaluating the objective function at these points. Then, one can manually select the maximum from the resulting points. The Dakota call for this optimization scheme is “multidim_parameter_study” and an example of the required input for three variables is shown below (ADAMS):

```
method,
      multidim_parameter_study
            partitions = 9 10 15
```

The number of partitions indicates how many times the domain is divided evenly. This means, for this example, the first variable is evaluated at ten different locations, the second variable is evaluated at 11 and the third variable is evaluated at 16 different locations. In all, this results in 1,760 different function evaluations. One can see that the

number of total evaluations climbs very fast. This is especially true as the number of variables increase.

To use this method as a reliable optimization scheme, one must partition the domain finely so that a smooth surface is established and the optima is evident. This is computationally very expensive. In addition, the drawback is that the multidimensional study does not use past evaluations to inform decisions on what parameters to test in the future. Therefore, it will spend time testing points that are not in an optimal domain. To reduce the number of "bad" iterations, one could reduce the number of partitions along the domain. However, this would result in a poorly resolved parameter space and a low fidelity optima.

However, computationally speaking, a silver lining can be drawn from this. Since all the test points are determined *a priori*, the concurrency of this optimization method is very high. In fact, this is an *embarrassingly parallel* process as no iteration requires information from past simulations.

Regardless, this optimization scheme is good to use as a launching pad for future simulations. One can resolve a sparse parametric space to find a rough optimal domain. Then, one can use a secondary optimization scheme that focuses on the constrained domain. In the case of gradient methods, one can use the optima found by the parameter study as the initial point of the scheme. In general, these types of two stage techniques are referred to as *hybrid schemes* (ADAMS).

3.2 GRADIENT DECENT

A more efficient optimization scheme is the gradient decent method. To call this optimizer the "conmin_frcg" call is placed in the input deck (Adams). The conmin_frcg optimization scheme uses a Taylor expanded, central difference technique to calculate the gradient at a point (J Haslinger, 2003). One assumes that the function being optimized is continuous and differentiable. The infinitesimal interval (h) is found by multiplying the initial variable value by 10^{-4} then adding or subtracting appropriately. The central difference gradient used is shown below:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

This is operation applied to each dimension independently. The gradient is formed by assessing these values along their respective dimensions.

$$f'(\vec{x}) = [f'(x_1),\ f'(x_2) \ldots f'(x_i)\ ]$$

The process for selecting a new point involves two steps. First an intermediate test point is selected along each dimension. The new value along the dimension in question is determined by using Newton's method (Ahmad Shukri Nazri, 2017):

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

However, the rest of the dimensions are given by subtracting their appropriate central difference values from the previous point such that the function is minimized. In totality, this can be expressed as (Adams):

$$\overrightarrow{x_{i,n+1}} = \begin{matrix} x_{1,n} - \dfrac{f(x_{1,n})}{f'(x_{1,n})} \\ x_{2,n} - f'(x_{2,n}) \\ \vdots \\ x_{i,n} - f'(x_{i,n}) \end{matrix}$$

Once an intermediate test point has been found for each dimension, the objective function is calculated. Finally, the parameters that correspond to the optimal objective function are selected as the initial point for the new iteration.

Even if a forward or backward difference scheme is more computationally efficient, the central difference scheme is used to ensure greater fidelity in the resulting gradient (Rhinehart, 2018).

Since a central difference scheme is used to determine the gradient, the objective function must be calculated 2n + 1 times per iteration (where n is the number of dimensions in the function space). This means that the concurrency of this optimization scheme alternates between 2n + 1 for the gradient step and n when the algorithm is searching for a new initial point. This is evident as each iteration informs the location for where the subsequent iteration will be calculated. There is no way to separate the order of the optimization and therefore it is an inherent drawback to using a gradient method.

An example of the input used for subsequent optimizations is show below:

```
method,
      conmin_frcg
            convergence_tolerance = 1e-8
            max_iterations = 100
```

In this example, two optional stopping calls are added to the input. For the convergence tolerance option to be satisfied and stop the optimization, the given value

must be greater than the difference in objective function divided by the previous objective function.

$$Convergence\ tolerance\ > \frac{OF_{i-1} - OF_i}{OF_{i-1}}$$

The second call is more straight forward. For the optimization to be terminated because of this flag the program must complete more than the listed number of iterations. Note that this is not referencing individual evaluations of objective function rather total calculations needed to establish a gradient. Therefore, if a function's parameter space has three dimensions, seven calculations of objective function are needed to establish the gradient and another three are required to find a new initial point. However, these ten objective function evaluations would only count as a single iteration.

In general, this method is extremely efficient at finding minima. However, it is vulnerable and subject to get stuck in local solutions (J Haslinger, 2003). Without an informed initial guess, it is hard to find global optima.

Even with a reasonable initial condition, if the function being optimized is not well behaved, it is very difficult to converge on a reasonable solution. This is subject to happen especially in computational studies (such as this one) where the simulations themselves are subject to noise. This introduces many local optima that will likely trip up this gradient optimization scheme.

3.3 SINGLE OBJECTIVE GENETIC ALGORITHM

To combat the local solutions one converges upon in the previous method one can move away from a gradient based method. An example of a "gradient-free" optimizer is

the genetic algorithm. In Dakota, this optimization scheme is called “Soga” or Single-Objective Genetic Algorithm.

The concept for this algorithm is based on the genetics seen in everyday life (Michaeli, 2003). It likens the parameters that make up a function to the genes that make up life forms.

$$OF = f(gene_1, gene_2, \dots, gene_n)$$

Comparable to natural selection, once a new objective function is calculated, it is compared to other iterations. Typically, the parameters which produced less optimal objective functions are discarded. The general approach to a genetic optimization is shown below.

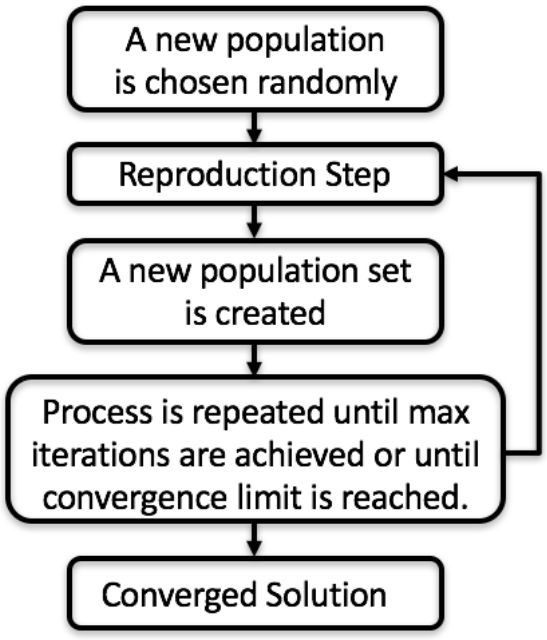


Figure 2: A schematic of a genetic algorithm

Where the genetic algorithm mostly resembles natural selection is in the reproduction step. These comparisons are shown in the table below.

Principle	Biology	Optimization
Replication	Asexual Reproduction	Copying existing parameters
Mutation	Random changes in genetic material	Changing random parameters randomly
Birth	Creation of new genetic string	Selecting a new parameter set
Life	Survival in an environment	Evaluation of the new parameter set
Selection	Selection of most adapted organism	Discarding select parameters

Table 2: A list of comparisons between biology and the genetic optimizer. Table adopted from Michaeli, 2003.

Using these steps, one can create a schematic for a reproduction step. This is shown in the graphic below. The objective of the example shown below is to maximize height in the population.

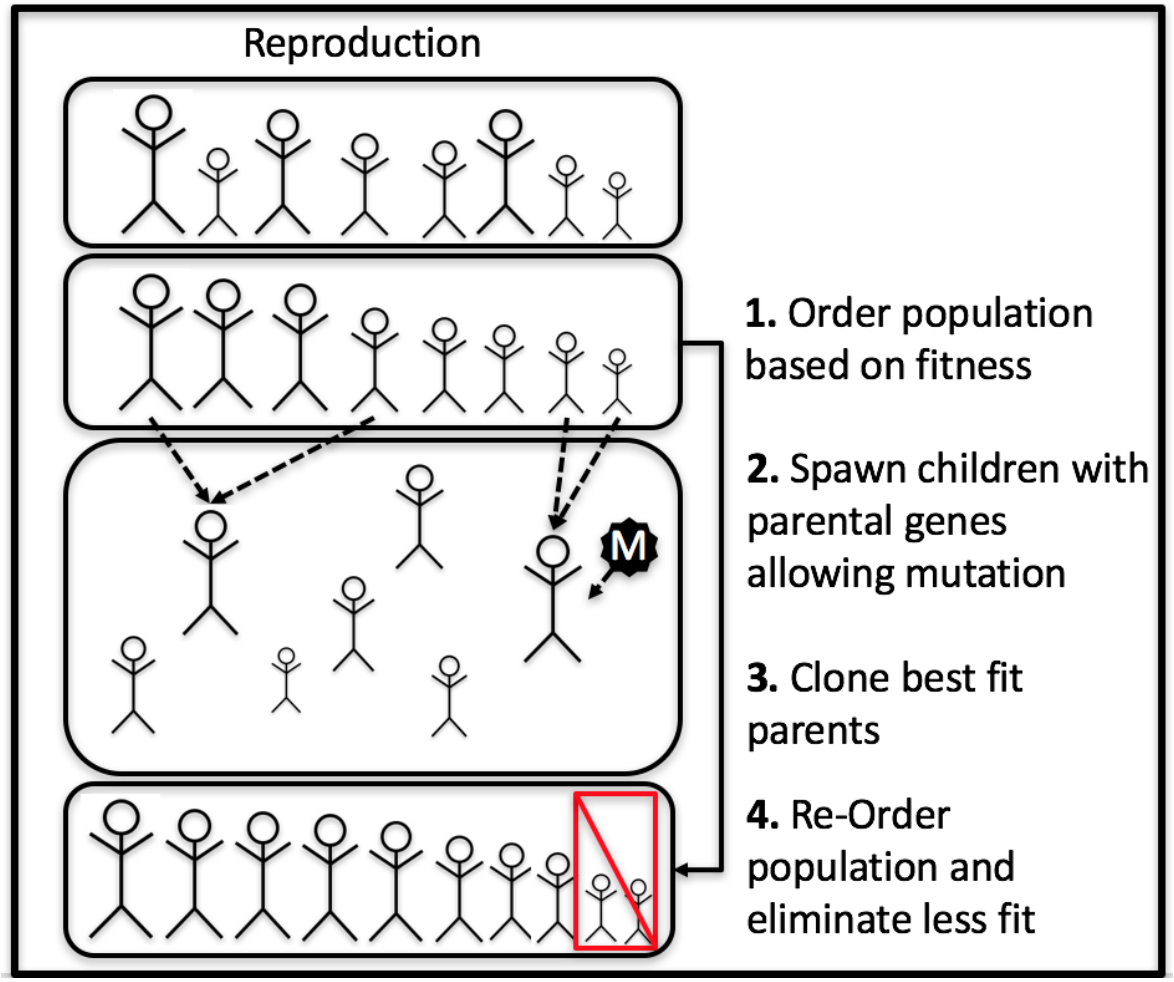


Figure 3: A schematic of the reproduction cycle in a genetic algorithm. Adapted from the Dakota Reference Guide.

This "spawning process" can be expressed more formally. First, any number of parents (up to the total number of variables) are selected. In this scenario, two parents will be used.

$$OF_{Parent\ 1} = f(x_1, x_2, \dots, x_n)$$

$$OF_{Parent\ 2} = f(y_1, y_2, \dots, y_n)$$

Now a child is spawned. The crossover step is implemented and variables from both parents are selected for the child.

$$OF_{Child} = f(x_1, y_2, \dots, y_n)$$

Next, the mutation step occurs. In this step, a user supplied probability dictates the likelihood that a "mutation" will occur in the child's genes. This is manifested by replacing a parent variable with a mutation value.

$$OF_{Mutant\ Child} = f(x_1, y_2, \dots, M_i, \dots, y_n)$$

The purpose of the mutation step is to create some artificial noise in the optimization scheme. This is vital to avoid focusing on local minima.

One exerts control over the optimization by manipulating the different levels of the reproduction (Nikos D. Plevris, 2013).

For example, one can control the way that crossover occurs. One can choose a bit crossover method where the list of parental parameters are exchanged N number of times. This is the type that was shown above. However, one can choose a more stochastic approach where child parameters are chosen randomly from any parent above a fitness threshold.

Additionally, one can control the mutation that occurs within the child population. Besides simply changing the likelihood of a mutation occurring, one can control how it manifests in the "child genome" (Adams). One can specify the distribution of the variable domain so that some values are more statistically likely than others. Distributions included uniform, normal and Cauchy. Furthermore, one can even choose to convert a variable's value into binary. In this case, the mutation manifests by "flipping" a random bit.

Finally, one can control the portion of the population that can crossover and the portion that is eliminated at the end of the reproduction. Traditionally, only the fittest are allowed to contribute variables to the next population and the least fit are eliminated.

However, this can encourage local solutions and does not include “genetic variety” outside of the domain that was initially provided. Therefore, a more sophisticated approach is to select a random distribution weighted towards the fit population for crossover and the least fit for elimination.

4 OPTIMIZATION SETUP

4.1 OBJECTIVE FUNCTIONS

Finally, one must find an objective function to optimize. Maximum jet kinetic energy, penetration length or even spall angle are valid examples of objective function. In theory, this can be something completely arbitrary and "loosely coupled" to the parameters that control it. In other words, it is not mandatory to have full control over the all parameters contributing to functionality of the objective function when looking for optima.

$$OF = f(x_{known}, x_{unknown})$$

In this case, leaving relevant variables "free" creates a family of solutions whose dimension is equal to the number of unknown variables. However, the purpose of this study is to reproduce a unique result not a family of solutions. Therefore, to create an objective function whose global minima is uniquely defined, one must list and optimize all parameters controlling the objective function. This section discusses the formation of the two objective functions used in this study, the next section outlines the methodology used to select the full scale of representative variables for the objective function. These points will be further discussed when parameterizing the shaped charge.

In addition, it is equally important to create an objective function that emphasizes the physics of the shaped charge system over numerical artifacts in the CTH code. For example, while one may find some solutions by maximizing the peak kinetic energy in the domain, most Dakota solutions will be of liners producing low mass flecks of high velocity material; a fundamentally poor shaped charge design. To this end, it is more

advantageous to post-process global CTH results into objective functions. It is also useful for the post-processing itself to introduce some functionality so that the CTH noise is minimized. This noise is mainly attributed to the low resolution of the CTH simulations. Essentially, some noise in the results was chosen as a tradeoff for computational speed.

As previously explained, since the goal of this study is to converge on a set design, information from both the reference and iterative designs must be included in the objective function. Therefore, the general structure of the *OF* is to select a metric and subtract the iterative result from the reference result. This way the objective function space converges on the reference parameters.

This study focuses primarily on the "kinetic energy profile" of the shaped charge jet. After a CTH simulation is run, a data dump file is generated containing the values of kinetic energy along the centerline of the jet. The kinetic energy profile is created by pairing the kinetic energy values with their respective location along the jet. To create a normalized profile, the kinetic energy values are normalized with the reference run's maximum kinetic energy and the y values are normalized with the reference run's jet length. An example of these profiles is shown below.

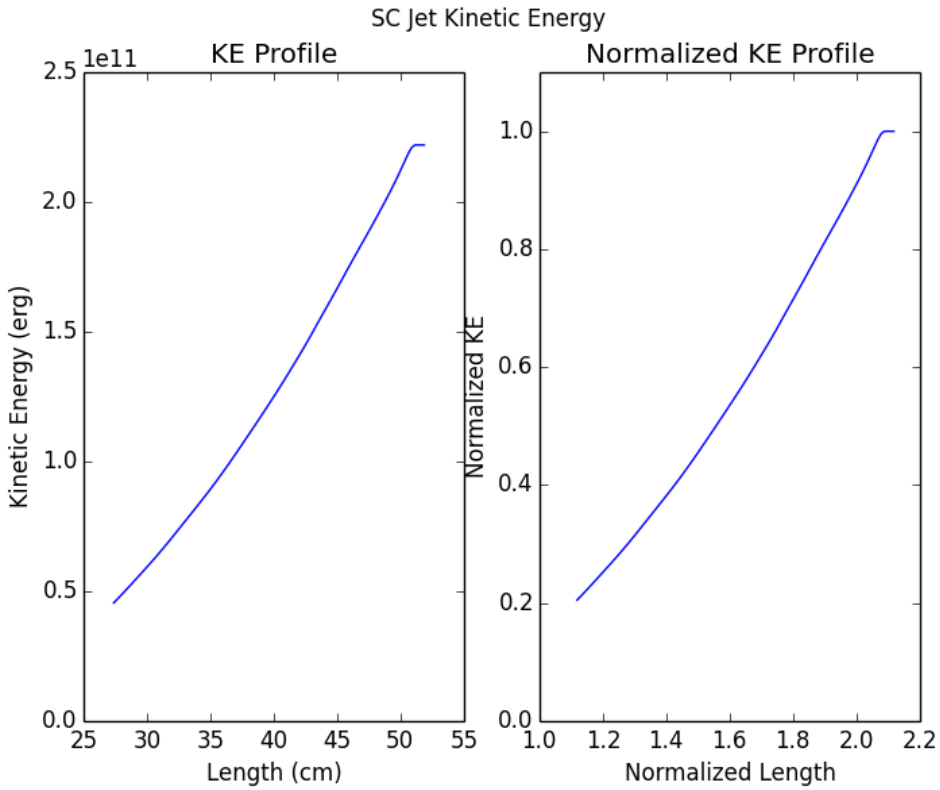


Figure 4: A comparison between a kinetic energy profile built from raw data and a kinetic energy profile whose axes have been normalized.

Further post processing parses metrics from the normalized profile to be used for objective functions.

The first metric used to formulate the objective function is a normalized angle of the kinetic energy profile. The following steps are used to parse this value from the normalized kinetic energy profile.

1. Curve fit a linear profile to the normalized plot. This will yield the slope of the fit.
2. Convert the slope to radians and normalize this value with $\pi/2$
3. Construct the final objective function by subtracting the iterative normalized angle from the reference normalized angle. Square this difference.

$$OF = (\theta_r - \theta_i)^2$$

An example of this is illustrated below.

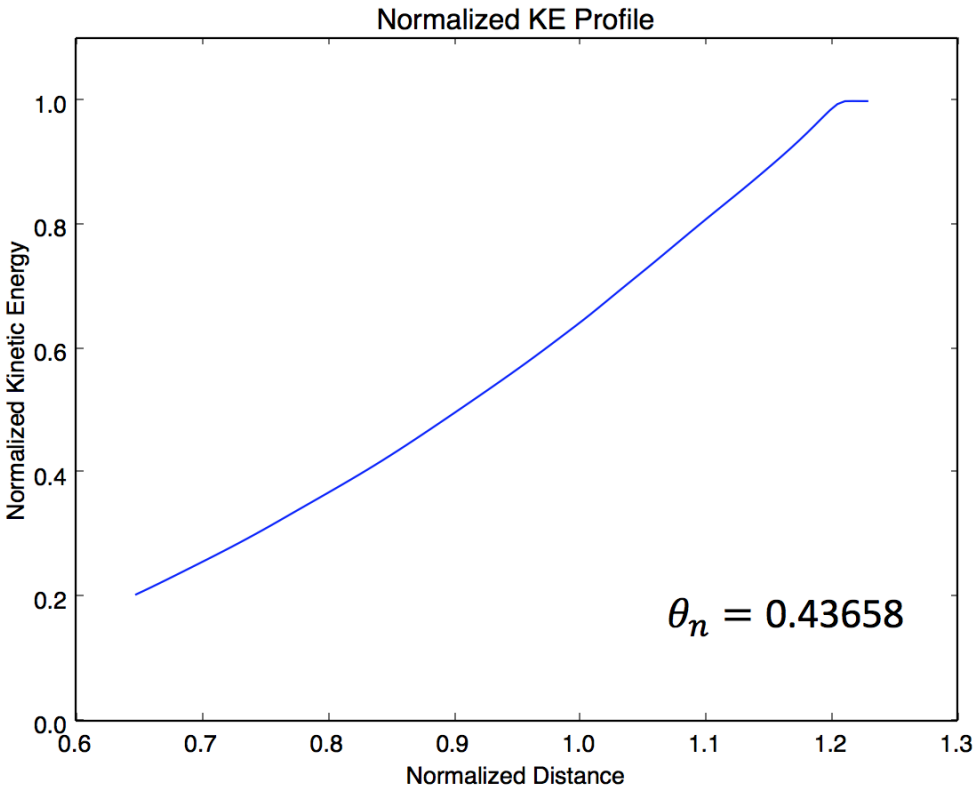


Figure 5: An illustration of a normalized kinetic energy profile and corresponding normalized angle

This objective function is useful as it directly describes the differences in slope between the reference and iterative CTH simulations. One can easily observe that if the iterative normalized angle is equal to the reference value the function is zero and is considered "optimized."

In addition, this process is thought to minimize CTH noise. Firstly, the root values that form the objective function are pulled from global CTH phenomena. Secondly, the combination of linear and non-linear scaling introduces a "post-processing" layer of functionality that also smooths the objective function.

Exploiting a linear regression, the second objective function is even more of a functional "black box" as the first. The following steps are taken to calculate it.

1. Curve fit a linear profile to the normalized plot of the reference kinetic energy profile. This will yield reference values for the slope and y-intercept.
2. Every iteration, find the R^2 value between the iterative normalized KE profile and the curve fit reference profile.
3. Construct the final objective function by subtracting this value from one.

$$OF = 1 - R_i^2$$

An example of this is shown below.

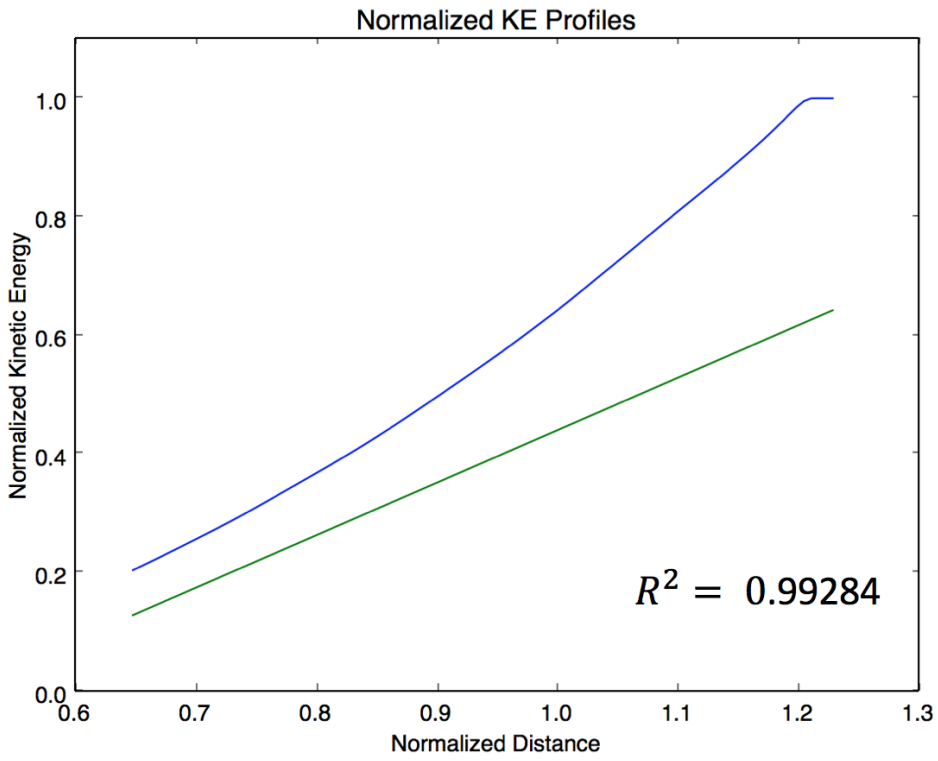


Figure 6: An illustration of the linear regression between two normalized kinetic energy profiles. The blue is the normalized KE profile and the green is the reference KE profile generated by the target shaped charge geometry.

Just like the first *OF*, this objective function exploits a global trend of kinetic energy, reducing possible noise from CTH.

4.2 SHAPED CHARGE PARAMETERIZATION

As previously stated, the goal of this study is to reproduce a "reference" SC geometry by characterizing its kinetic energy profile. To begin, the SC liner was parameterized by characterizing it as two equal parabolas separated by a constant thickness. Specifically, the parameterized liner profile is given by:

$$y_1 = ax^2 + bx + c$$

$$y_2 = ax^2 + bx + (c - \Delta h)$$

Arbitrary values of coefficients a, b and c are selected (and show below) to construct the "target design." The goal of the Dakota optimization will be to reproduce these coefficients.

a	b	c
0.30	0.40	5.00

Table 3: The list of reference parameters used to generate the reference shaped charge metrics. These will be the target parameters for the Dakota optimization

The parabolas range from 0 to 3 cm and when they are inserted in the CTH input deck they are expressed by 15 evenly spaced points. An example of a SC with this parabolic geometry is shown below.

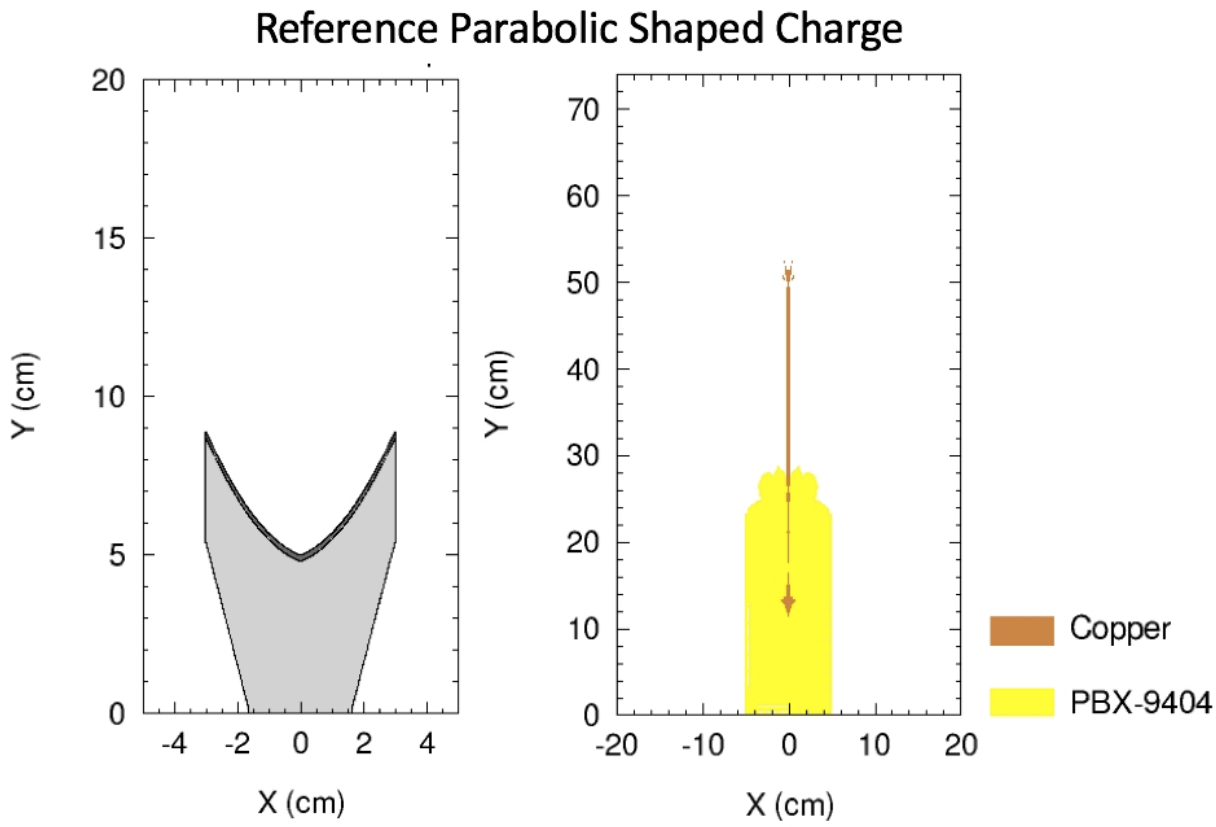


Figure 7: An illustration of the reference shaped charge geometry corresponding jet. Thickness is 0.20 cm a=0.30 b=0.40 c=5.00

One could argue that Δh is a natural fourth parameter for Dakota to optimize. However, due to the added computational cost this would cause, the liner thickness was chosen to be constant. In addition, and perhaps more important, leaving the thickness of the parabolic liner constant grants the optimization conservation of mass throughout each simulation. In other words, no matter how a, b and c vary, the mass will stay the same. This counterintuitive result is shown to be true in the appendix.

Choosing a value for the liner thickness requires an iteratively informed decision. Preliminary CTH simulations were run varying the thickness of the liner. Then, metrics were compared to assess the quality of the resulting jet. These include maximum and

total kinetic energies, jet length, and candidate reference values for future objective function. These comparisons are shown in the table below:

	Δh (cm)	L (cm)	ΔL (cm)	MKE (KJ)	ΣKE (MJ)	a,b	θ_n
Parabolic Geometry	0.05	62.95	34.0	33.60	9.21	0.869, -0.660	0.45530
	0.1	58.25	29.9	27.43	9.11	0.857, -0.687	0.45111
	0.2	50.65	23.2	20.39	5.58	0.818, -0.790	0.43658
	0.3	44.45	17.3	15.17	3.40	0.749, -0.905	0.40929
	0.4	39.75	13.2	11.69	2.16	0.670, -0.985	0.37594
	0.5	35.95	9.6	9.27	1.36	0.568, -1.085	0.32899

Table 4: A comparison between different liner thicknesses and corresponding max jet length, jet length, maximum kinetic energy, total kinetic energy, coefficients to the linear fit of the normalized kinetic energy profile and normalized angle.

Shown below are CTH images of jet profiles and corresponding density plots for select thicknesses.

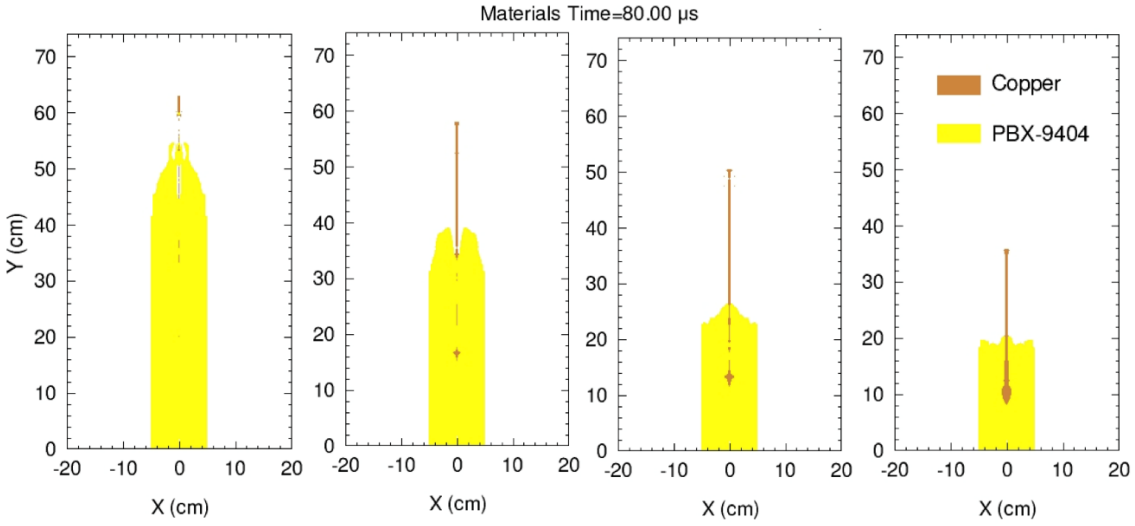


Figure 8: Illustrations of the shaped charge jet procured from various thicknesses. From left to right, thicknesses of 0.05, 0.10, 0.20, 0.50 cm

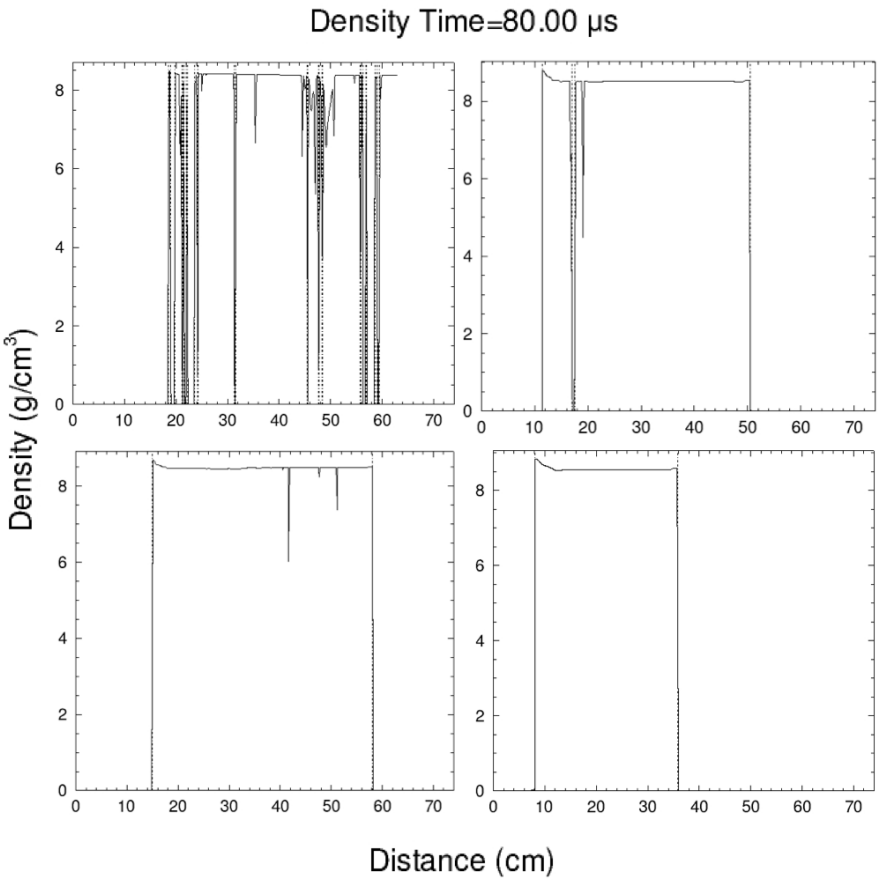


Figure 9: A comparison of density versus distance of shaped charge jets of varying thickness. Upper left, t= 0.05 cm, upper right t=0.20 cm, lower left t=0.10 cm, lower right t=0.50 cm

The criterion for selecting a constant thickness was to maximize the metrics shown in the table above. Naturally, the lower thicknesses, having less mass, were accelerated much faster. This results in higher kinetic energy. However, creating well-formed jets each simulation is an additional concern. Besides the obvious reduction in efficiency that occurs when jet breakup occurs in real world applications, this phenomenon cripples the numerical methods as it reduces the sample size of points from which an objective function is parsed.

Observing the images above, for the lower thicknesses it becomes evident that jet breakup is prevalent. Therefore, it is important to choose a value that will consistently result in well-formed jets regardless of the range of geometries that the optimization requires.

Keeping the above in mind, a value of 0.20 cm was chosen both to maintain quality and to maximize length and kinetic energy values of the jet. Between the 0.10 and 0.20 cm thickness run, one observes jet breakup in both simulations. The 0.20 cm thickness value was favored as the breakup occurs to separate the jet from the slug. This contrasts with the thinner thickness where breakup occurs in the body of the jet.

In conclusion, the shaped charge geometry has been parameterized as two parabolas separated by a constant thickness of 0.20 cm. The target parameters that the Dakota optimization will attempt to reproduce are:

$$a = 0.30$$
$$b = 0.40$$
$$c = 5.00$$

In conjunction with the objective function formulation, one has assumed that the objective function can explicitly be expressed a function of a, b and c.

$$OF = f(a, b, c)$$

Additionally, one can imagine what an ideal solution space would look like. In an applied setting, it is usually allowed for an objective function to have multiple solutions. In other words, if the objective function characterizes penetration, the presence of multiple geometric solutions is not a problem.

However, given that this particular optimization problem is to reproduce a specific geometry, the presence of multiple solutions would indicate a failure to capture the relevant physics in the objective function. As was alluded to in the previous section,

this would indicate that there is hidden functionality in the objective function that is not represented due to the absence of relevant parameters.

$$OF = f(a, b, c, x_{unknowns})$$

An idea case is a space which is a bijection between a, b, c and the metric being measured (Cormen, 2009). By subtracting all the values of the space from the reference value of the metric, one centers the space with the reference geometry. Finally, the objective function is found by squaring the space.

$$OF = (f(a, b, c)_{reference} - f(a, b, c)_{iteration})^2$$

The resulting objective function solution space uniquely defines the reference geometry and sets it at zero.

$$0 = f(a_r, b_r, c_r)$$

In other words, an ideal solution space is one where the *OF* has only one zero that is uniquely mapped by the reference parameters.

Realistically, one should expect a surjective map between a, b, c and the objective function. Then, it is the user's job to find the appropriate global minima of the solution space.

4.3 DEBUGGING CTH

A challenge to this optimization problem, and algorithmic optimization in general, is that if a single evaluation of the objective function is not properly calculated the entire scheme fails. In other words, if the black box CTH simulation does not produce an objective function the Dakota optimizer will abort the optimization. Therefore, it is of the utmost importance to keep CTH functioning properly.

Post detonation, flecks of material can come off the main body of the jet. These flecks are very small and have proportionally low mass. As they advect through the domain, numerical instabilities within the CTH code will propose un-physical conditions for them. These can be entertaining as the hydrocode can propose temperatures well above that of the sun or material sound speeds that are higher than the speed of light.

As these instabilities present themselves, the CTH will increase its time step. Once the lowest allowable time step is achieved, CTH aborts the simulation prematurely. Since the data dump file is written only after the first and last time step, the post-processing scripts will fail to produce an objective function. This obviously presents problems for Dakota as it will not receive an output to further the iterative process. Therefore, what started as a numerical instability manifesting on a tiny little fleck of material results in the total shutdown of the optimization.

To debug this issue, the first recourse should be to use the CTH discard section. In the CTH input deck, one can use this section to judiciously eliminate problematic material based on thermodynamic values. While this may work for individual CTH runs, developing discard conditions that asses all numerical instability for the entire domain of geometries is very difficult. In fact, as optimization studies were conducted, about one in five hundred runs was stopped because of this problem.

Instead, a different methodology was used, in conjunction with the discards, to combat this problem. If the simulation terminates unexpectedly, a sub-routine in the black box *cth_simulator.sh* file is activated. This process generates new liner geometry using parameters that are composed from original input added to a perturbation. A second simulation is run with the new liner. A maximum of two additional runs can be simulated

after the original has failed. If CTH does not conclude the third simulation, the problems are likely deeper than a numerical instability.

Effectively, the accuracy of the objective function parsed from the simulation is scarified for the health of the entire optimization. Since the discards take care of most of the DTMIN issues, prevailing sentiment is that the sparse use of this technique is justified.

A second issue with broadly defining the parameter space is that not all parabolic liners will produce a suitable jet. This is inherently different from the first issue. First, the simulation runs to completion. Therefore, the data dump file is created and the perturbative geometry sub-process is not run. Second, and more important, it is generated by user error not numerical instability.

An example of a problematic shape charge with this issue is shown below.

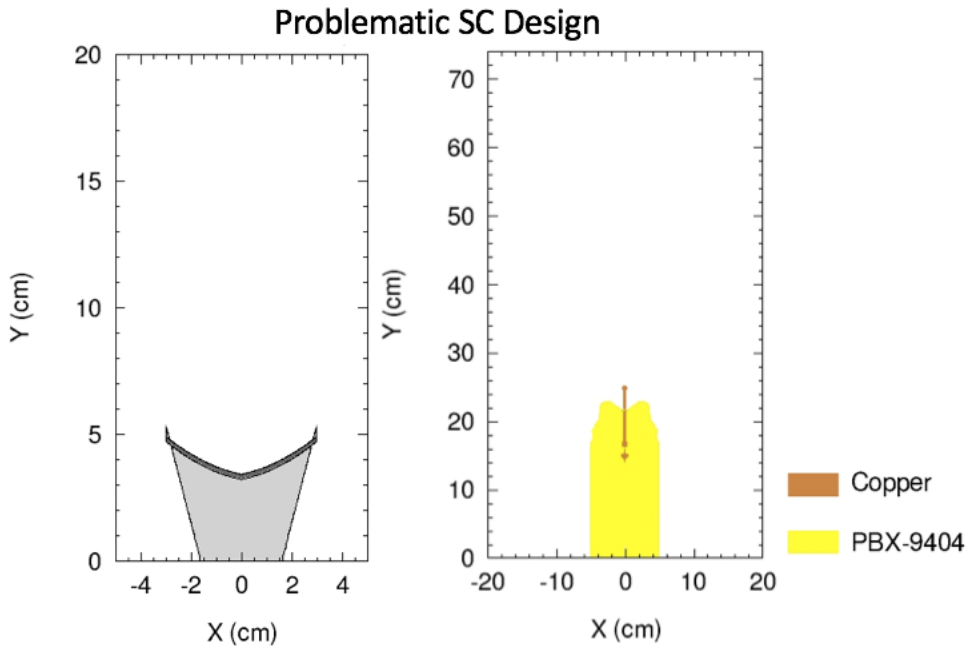


Figure 10: An illustration of poor liner geometry and corresponding jet. Observe the underdeveloped shaped charge jet. The parameters used for this simulation were a = 0.10, b = 0.20, c = 3.00.

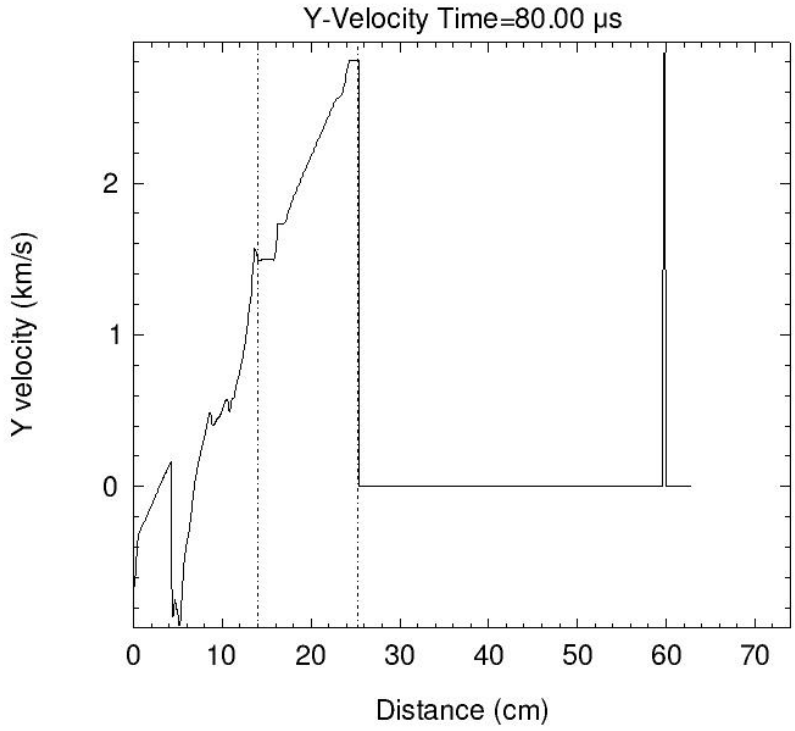


Figure 11: A plot showing the y - velocity profile (from a tracer) of the problematic SC jet. Observe that it does not meet the three km/s minima to write to the data dump file.

To solve this problem, this study proposes the simple solution of shrinking the domain in question. If one restricts the domain to a comfortable range of values the problem is eliminated. However, this prohibits the user from exploring the limits of the domain and it does not elucidate the stark transition from the null to jet producing shaped charge design.

A more sophisticated solution would be to create an additional discrete parameter for Dakota to optimize. This can be a simple binary output that describes whether a well-developed jet is formed. Of course, this requires an additional level of post-processing and the computational power to explore the added initial parameter domain.

This is left as future work.

5 RESULTS AND ANALYSIS

Before one even starts with the optimization process, it is important to see that varies enough to be optimized. In case of this study, both objective functions are based on the normalized kinetic energy profile. One can plot the KE profiles generated by the extremal cases of the geometric domain. This acts as a check to make sure there is enough variation between the cases.

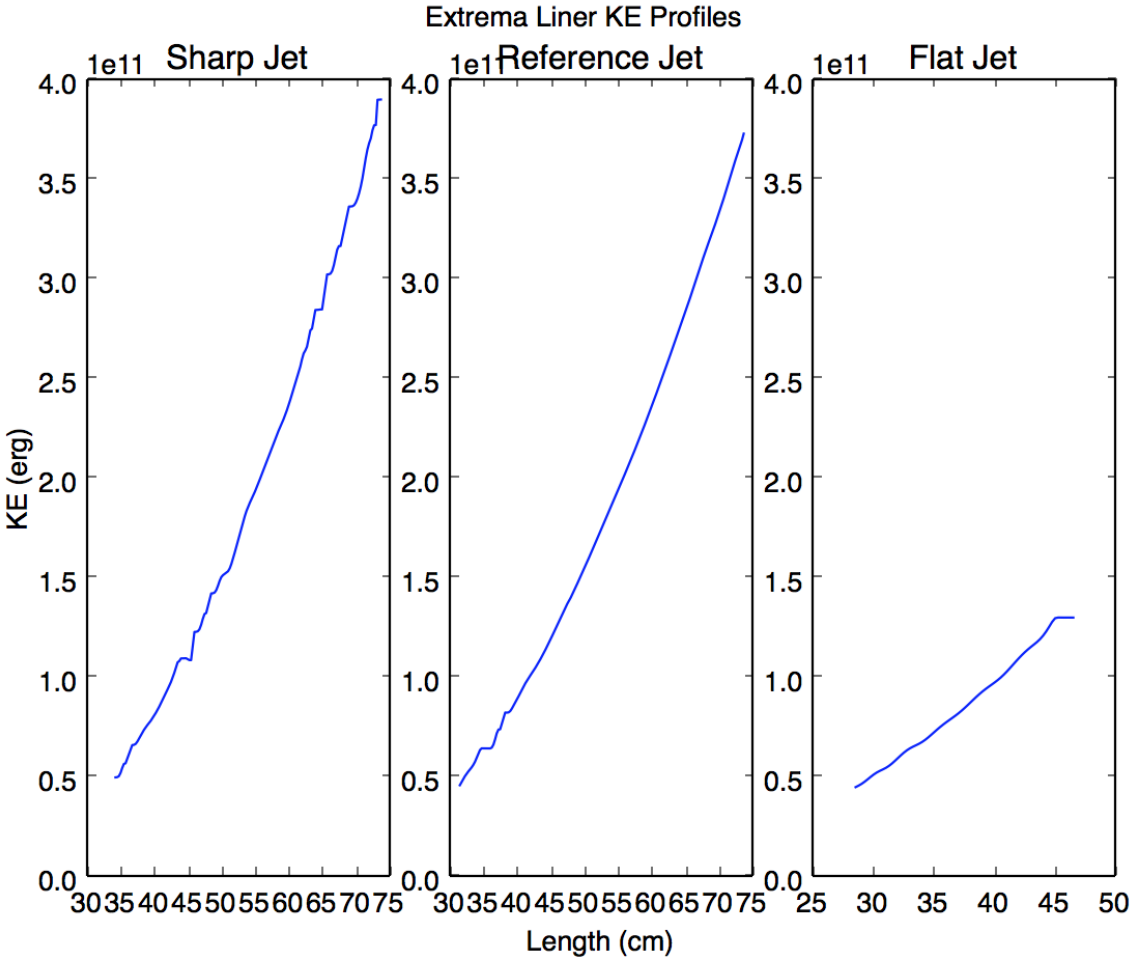


Figure 12: A comparison of kinetic energy profiles from various geometries. Sharp Geometry: a = 1.00, b=1.50, c=3.00 Reference Geometry: a = 0.30, b = 0.40, c=5.00 Flat Geometry: a = 0.13, b = 0.255 c = 7.00

By inspection, one observes that the extremal cases vary significantly. Therefore, assuming the *OF* is smooth, this is a space that can be optimized.

5.1 NORMALIZED ANGLE

The first objective function used to analyze the parabolic shape charge was the normalized angle metric. A parametric study was run on this space. A sample of the input file is shown below:

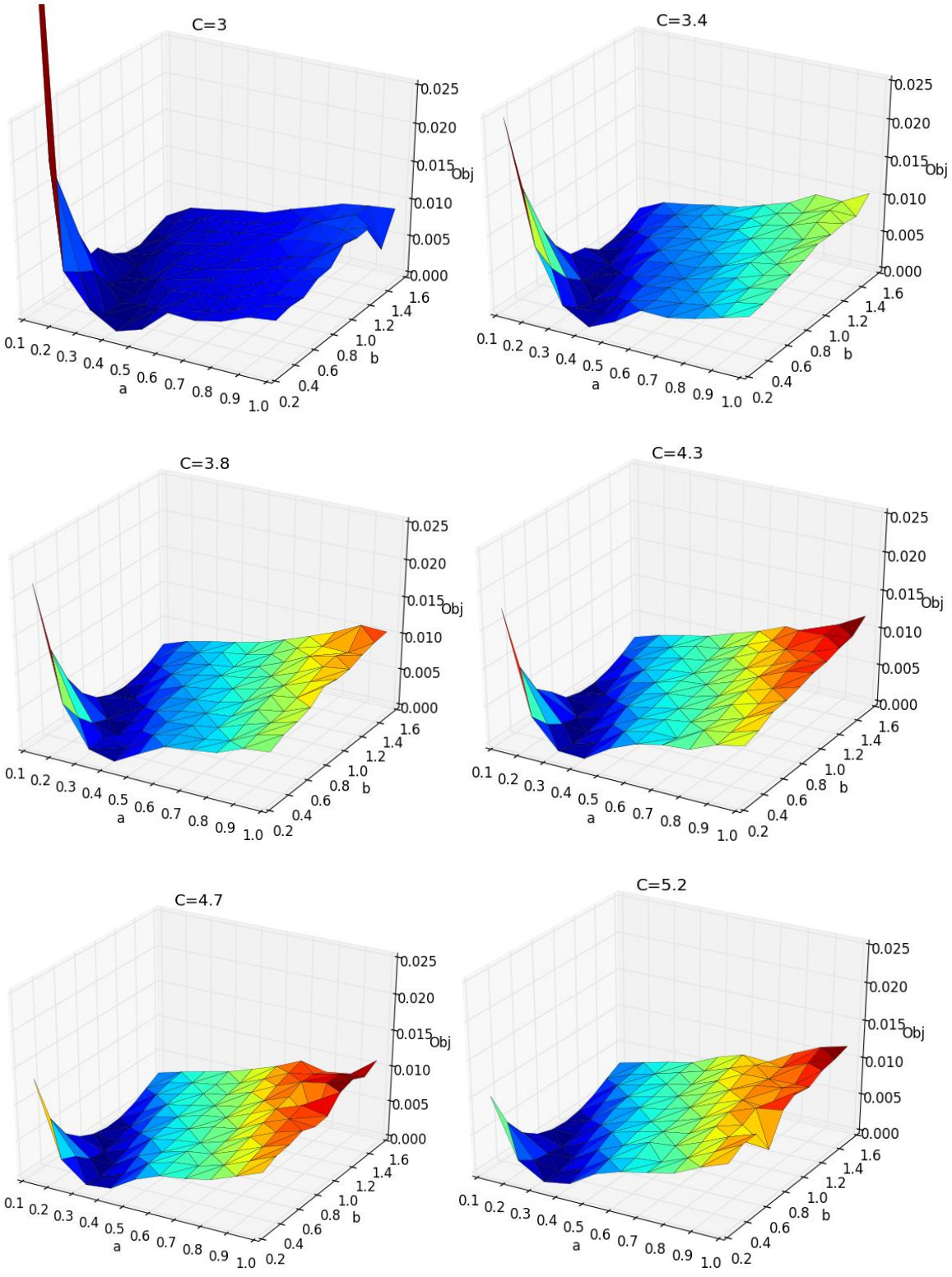
```
method,
      multidim_parameter_study
      partitions = 9 9 9

variables,
      continuous_design = 3
      lower_bounds          0.13  0.255  3.00
      upper_bounds          1.0   1.50   7.0
      descriptor            'A'   'B'    'C'
```

As one can see, nine partitions are called for each dimension. This divides the space 1000 times. In addition, one can see the domain that the variables can span. Reiterating what was stated in the Reference Run section, the target parameters are:

$$a = 0.30$$
$$b = 0.40$$
$$a = 5.00$$

This study was run on ten nodes with ten parallel calculations allowed on each node. This means that 100 concurrent CTH simulations could be run. The results from this initial calculation are shown in the images below:


C=3
C=3.4
C=3.8
C=4.3
C=4.7
C=5.2
Obj
a
b

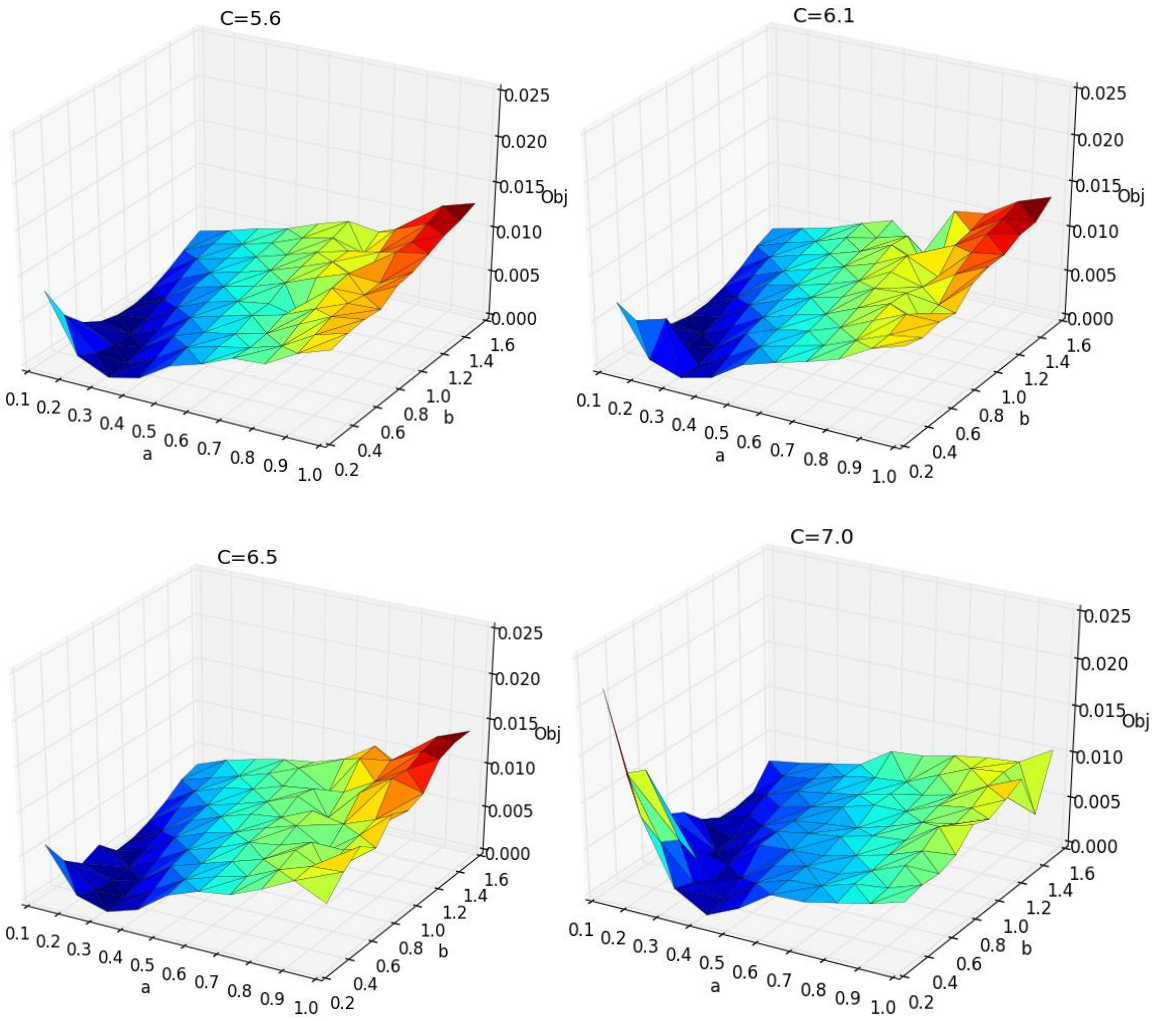


Figure 13: A collection of surfaces varying c showing relationships between a, b and *OF*. Recall that one seeks to minimize the *OF* therefore dark blue areas are target points. The first plot's axis has been changed to match the rest of the plots. Original plots are shown in the appendix.

Immediately, one observes that the functionality of the surface is comparable across all plots. One can describe the surface as a curved sheet with its ends pointing upwards. It appears that by varying the parameter "c" one affects two things. First, the general trend is that by increasing c the function is scaled. Specifically, as the left point decreases the right gradually increases. The exception to this is the shift seen between the last two plots. Here the right point is decreased while the left is increased.

Secondly, as these extremal points move, the “optimal valley” seems to shift accordingly. If the leftmost point is high the valley shifts away from it while if it moves down, the valley is attracted. This behavior is reminiscent of placing weights on a bed sheet and moving the edges of the sheet up and down. If one holds an edge of the sheet up high, the weight will travel away and *vice versa*.

By parsing through the array of test points, this optimization scheme found an optima at:

a_f	b_f	c_f	*OF*
0.3233	0.3933	4.7778	0.1266e-8

Table 5: The optimal parameters as found by the parametric study

Using this insight, a gradient optimization scheme was used to probe the solution space and attempt to reproduce the reference result. An example input that flags this gradient method is shown below:

```
method,
      conmin_frcg
            convergence_tolerance = 1e-8
            max_iterations = 100

variables,
      continuous_design = 3
      cdv_initial_point    .6    0.1   6.50
      lower bounds         0.0   -1.0  3.00
      upper bounds         1.5   1.00  7.0
      descriptor           'A'   'B'   'C'
```

As one can see, the domain for a, b and c has been increased. This should not influence the results as the gradient should keep the function bounded. However, if the function does escape the parameter study domain, this would indicate an unknown process is occurring. In other words, it is a good check.

Displayed below is a table of seven different optimization studies.

Run	a_i	b_i	c_i	a_f	b_f	c_f	*OF*
1	0.60	0.10	6.50	0.6001	0.0772	6.5009	0.3046e-2
2	0.50	0.20	6.00	0.4972	0.1986	6.0003	0.1658e-2
3	0.50	0.50	5.00	0.4864	0.5129	4.9734	0.2835e-2
4	0.15	0.30	3.25	0.1513	0.3006	3.2501	0.1988e-1
5	0.30	0.40	5.00	0.3054	0.4003	5.0032	0.5356e-5
6	0.3233	0.3933	4.7778	0.3241	0.3933	0.4778	0.1266e-8
7	0.1167	0.8632	6.5468	0.1142	0.9142	6.5446	0.3895e-8

Table 6: A comparison of a gradient optimization study run with different initial parameters. Run 1-3 are arbitrary points. Run 4 is on the "Hill" to the lower left of the domain. Run 5 is run on the reference point. Run 6 is on the minima found by the multidim param study. Run 7 is run on the Soga point (shown later).

These results are largely unsatisfactory. When examining the Dakota output files, the low objective function value suggests that all results have converged upon a valid solution. However, scrutinizing the final a, b and c values, it is evident that the optimization scheme did not march far and that every solution that is displayed is caught in some local minima. Additionally, the small variation between the input and output parameters indicates that these are not minima characteristic to the objective function, rather they are valleys caused by the noise of the CTH simulations.

Additionally, it is concerning that even Run 5, which was initialized at the target coordinates, traveled to an objective function that was greater than objective function values at supposed non-solutions. However, this may be explained by some numerical artifact.

In summary, since this gradient method is too susceptible to converge on local minima, a new methodology that searches for global minima needs to be implemented.

Therefore, the natural optimization scheme to use next is a gradient-free genetic algorithm. For this particular case study, a single objective function genetic algorithm (or evolutionary algorithm) is used.

```
method,
     soga
         max_iterations = 1000
         population_size = 50

variables,
      continuous_design = 3
      lower_bounds          0.1   0.2  3.00
      upper_bounds          1.0   1.50  7.0
      descriptor            'A'   'B'   'C'
```

For this run, the number of maximum iterations is set to 1000 and the initial population size is set to 50. Additionally, one can see, the domain is once again constricted. This is due to the problems outlined in the Debugging CTH section. To summarize, a larger domain introduces geometries that do not produce valid jets. Therefore, instead of developing methodologies to deal with these "bad" geometries, the geometries are omitted from the domain.

On these settings, the Soga optimizer uses all 1000 iterations. The optimal geometry that the Soga optimizer selected is as follows:

a_f	b_f	c_f	*OF*
0.1167	0.8632	6.5468	3.0377e-10

Table 7: The optimal parameters found by the Soga study

Below a figure reporting the various Soga iterations is displayed. The color scheme represents the calculated objective function at each point. Once again, dark blue is considered optimal.

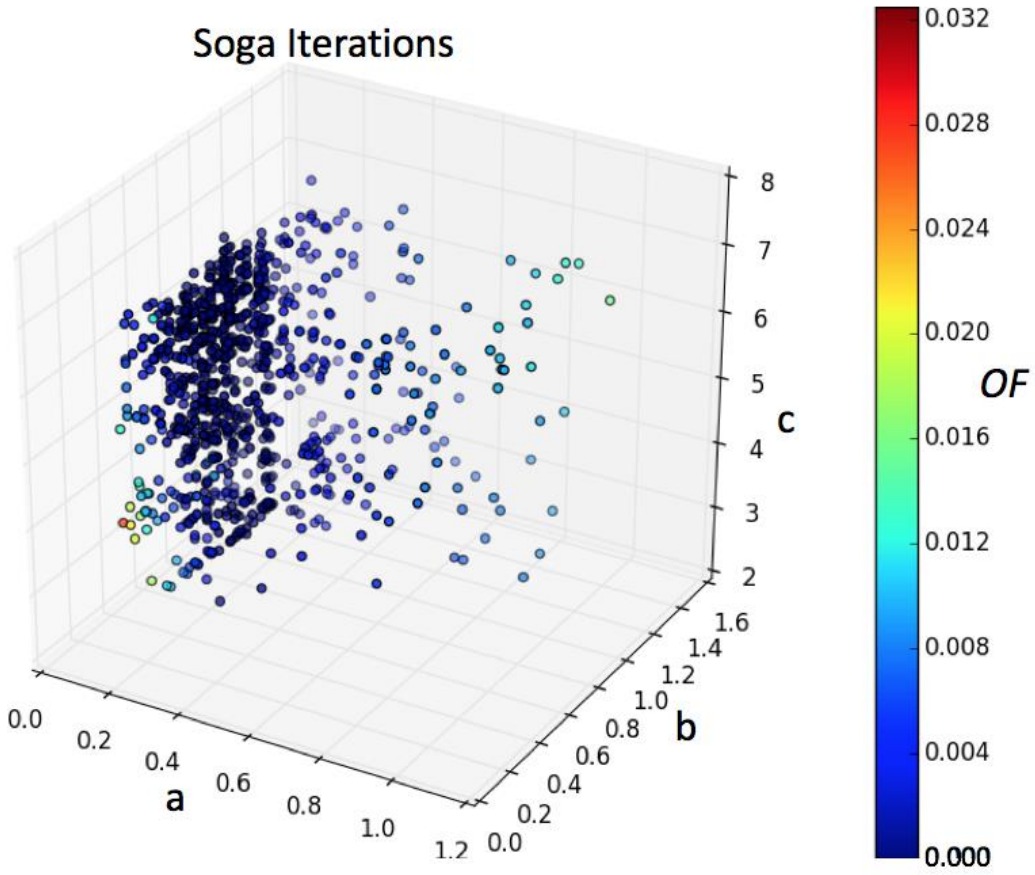


Figure 14: A scatterplot illustrating the trial points chosen by the Soga algorithm. The color scale displays the value of the normalized angle objective function

While one can say that the Soga algorithm does a good job focusing on areas where the global min might be, there is still the problem that the target parameters are not being reproduced.

Therefore, a shift in approach is necessary. Perhaps the most enlightening optimization was the parameter study. If it is true that c is just a scaling factor, then it can be removed from the objective function.

Indeed, the Soga run also supports this conclusion. Looking at the Soga scatter plot, there appears to be a column of test points that originate from the a-b plane and rise

along the c axis. This would be a clear indicator that the objective function is independent of c.

A possible technical explanation could revolve around the fact that the actual shape of the liner is independent from c. When the liner is defined, c is just the length where the liner is created. Therefore, there exists a family of identically shaped liners who are just being raised or lowered along the y axis. In effect, the only role c has is to increase the amount of explosive below the liner. It is fathomable to think that the objective function is independent of c.

Therefore, one can reformulate the optimization problem as a two-variable minimization! Now, keeping c constant at 5.00, one redefines the objective function as:

$$OF = f(a, b)$$

One can repeat the parameter study and Soga optimization. The gradient based optimization is not repeated as it will still be influenced by local noise. The optima from both simulations are shown in the table below:

	a_f	b_f	*OF*
Parameter Study	0.3100	0.3733	2.7342e-7
Soga	0.1910	0.7659	3.4237e-9

Table 8: A comparison showing the optima obtained by the parameter study and Soga optimizer on the a and b space.

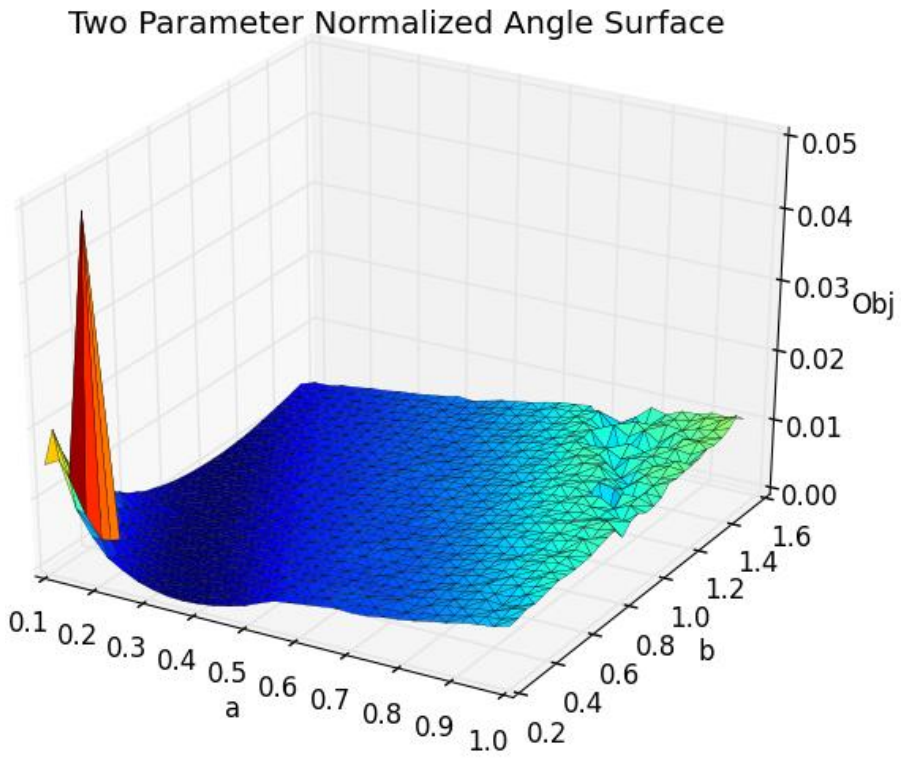


Figure 15: The normalized angle objective function surface with respect to a and b

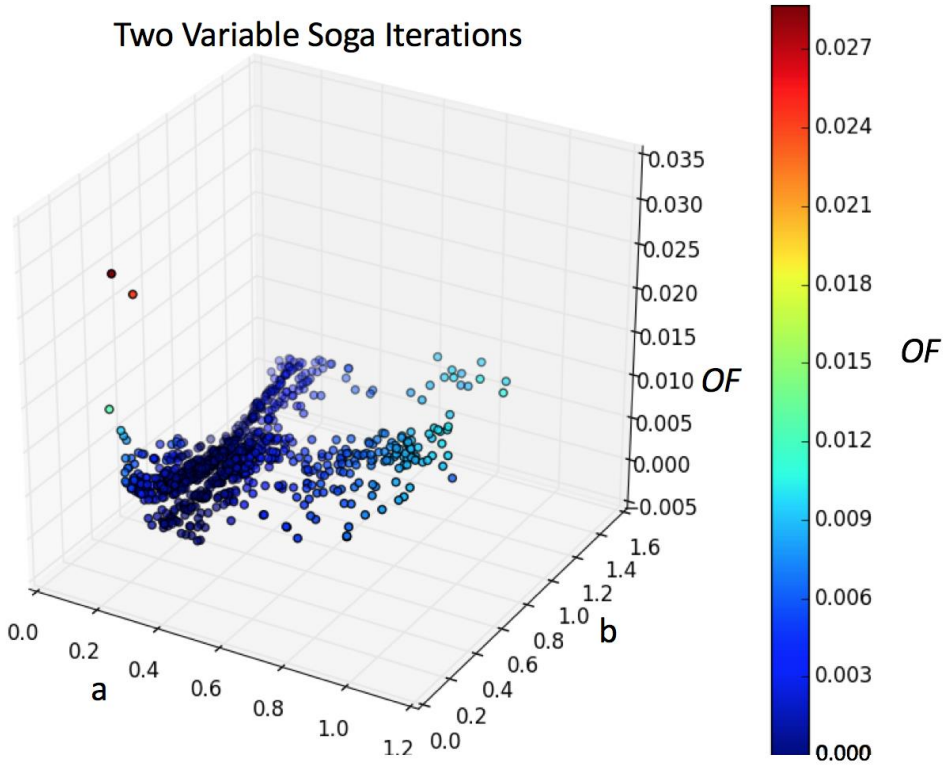


Figure 16: The scatter plot of test points chosen by the Soga algorithm in a and b space.

Looking at both plots (especially the parameter study surface) it becomes evident that, unfortunately, a unique minima is not defined. Instead the objective function exhibits a family of solutions that lie in the "optimal valley." One can slice the surface into multiple a, *OF* planes and select the minima of each plot. By fitting a linear curve to the minimal values, an expression is obtained explicitly relating the relationship between optimal a and b. This correlation is illustrated on the graph below.

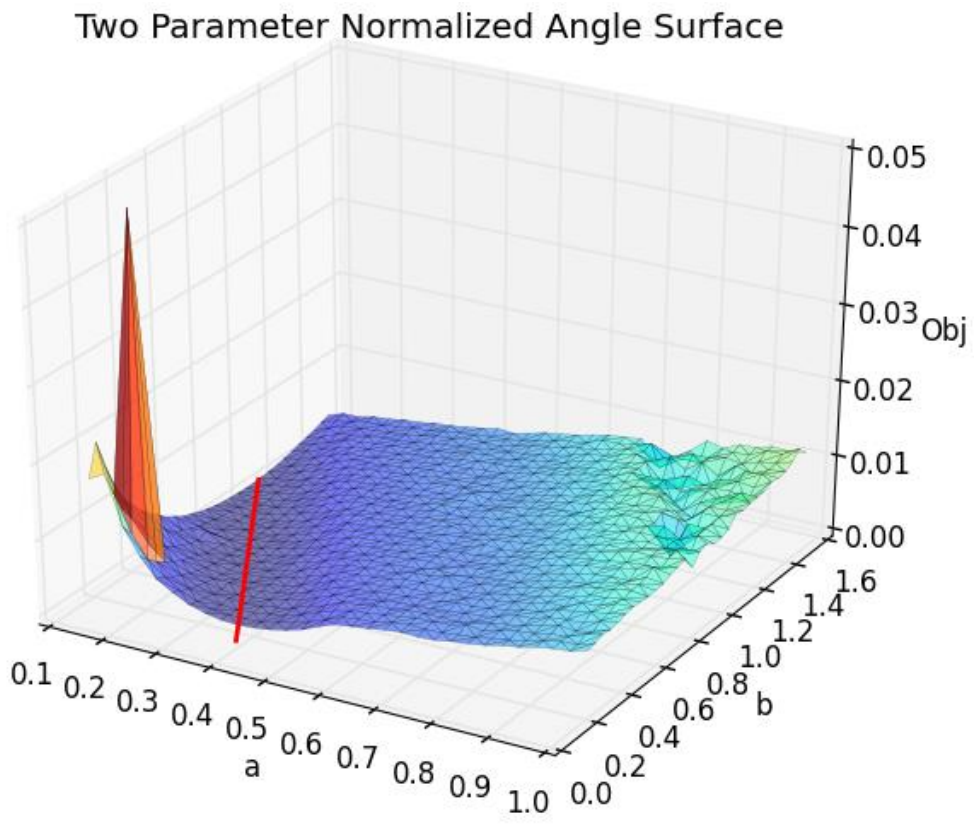


Figure 17: The a and b surface of the normalized angle objective function with a linear fit of minima

The solution space is visibly degenerate as the minima of the objective function is not uniquely represented by a_r and b_r.

The equation of the line expressing the degenerate solution set is given by:

$$b = \emptyset a + \tau$$
$$\text{For } \emptyset = -3.291 \; and \; \tau = 1.424$$

In other words, going back to how the objective function is defined, any a and b parameters that lie along that line produce a shaped charge jet whose kinetic energy profile has the same slope as the reference kinetic energy profile.

While this is not the result that was expected, mathematically, it is a very interesting outcome. It would be one thing if there existed a set of points that expressed local or global minima. However, this case does not produce those kinds of unique solutions. What is observed instead is a linear solution space between a and b.

Usually, this type of space is characterized by a system which is under-constrained (Shilov, 1977). In other words, there is a missing parameter which was not considered.

A second, more probable explanation, is that the functionality of this solution space is naturally produced by the physics of the CTH code. Reference the objective function.

$$OF = (\theta_r - \theta_i)^2$$

Considering θ_r is a constant and θ_i can be expressed as a CTH function of a and b, this can be rewritten as:

$$OF = (\theta_r - CTH(a, b))^2$$

Looking through this result, one can reinterpret the a, b and OF surface as a paraboloid with some CTH functionality at its minima. What is interesting is that it

appears like the parabola is rotated from the a, *OF* plane to incorporate some functionality in b.

In any case, one can gain insight by selecting from the family of solution geometries to analyze the respective KE profiles. Six different shaped charge geometries are shown below.

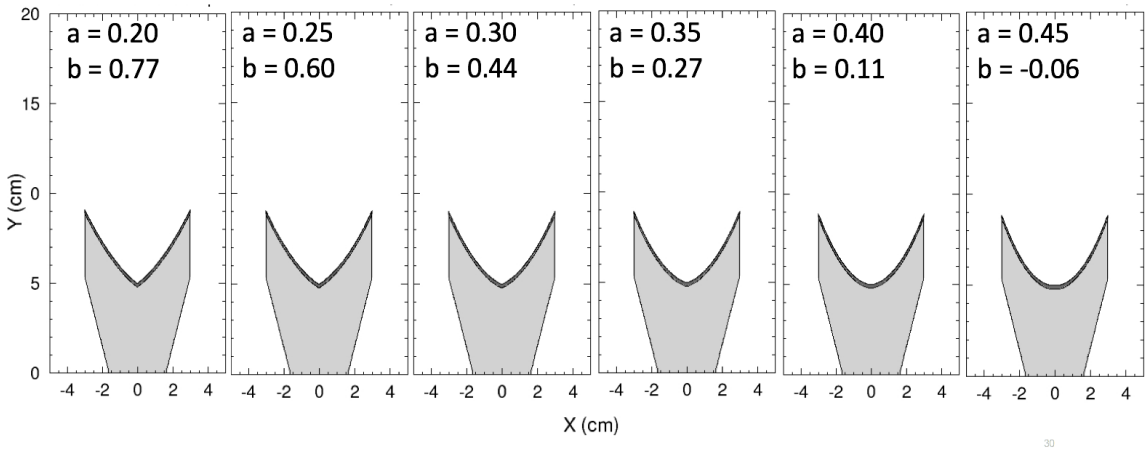


Figure 18: Displayed above are different shaped charge geometries with their respective a and b values generated from $b = -3.291\ a + 1.424$.

Pictured below are different comparisons displaying the resulting normalized kinetic energy profiles.

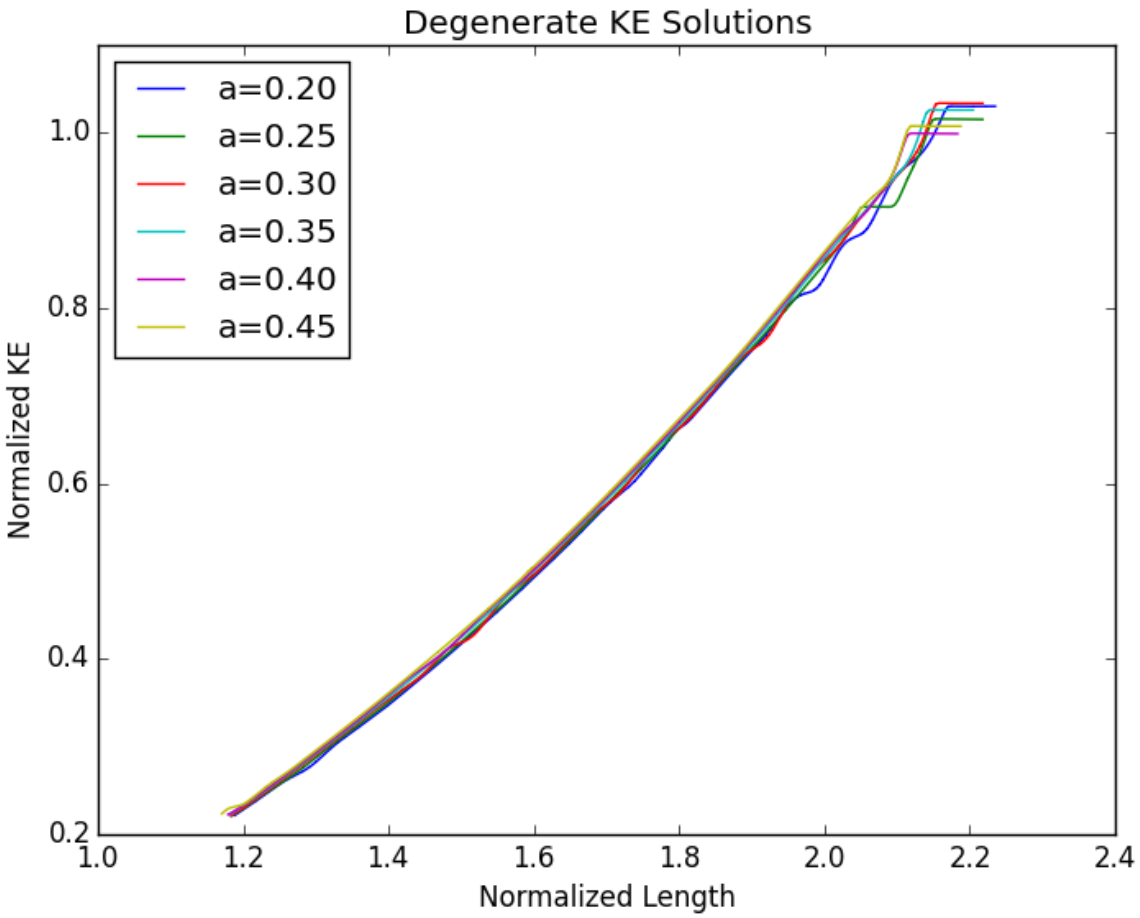


Figure 19: Illustrated above are the plots of each normalized KE profile. Observe that while the slopes are all very similar, there are differences with the length of the profile.

KE=mx+b	a=0.20	a=0.25	a=0.30	a=0.35	a=0.40	a=0.45
m	0.8226	0.8176	0.8276	0.8256	0.8191	0.8198
b	0.8026	-0.7920	-0.8050	-0.7990	-0.7867	-0.7843

Table 9: Above a comparison of the coefficients of the linear curve fit of the normalized KE profile. Observe that, while the values of the slope stay uniform, there are differences in the y-intercept values.

These comparisons are very illuminating. It appears that, while the slope of the KE profiles is the same, the total kinetic energy is different! This makes sense as the objective function is a measure of the slope of the KE profile, not its magnitude. Using the KE profile's y-intercept as the "missing parameter" one can explain the degeneracy in

the solution space. Therefore, one can describe the functionality of the objective function by the geometric parameters a and b and the y-intercept of the KE fit.

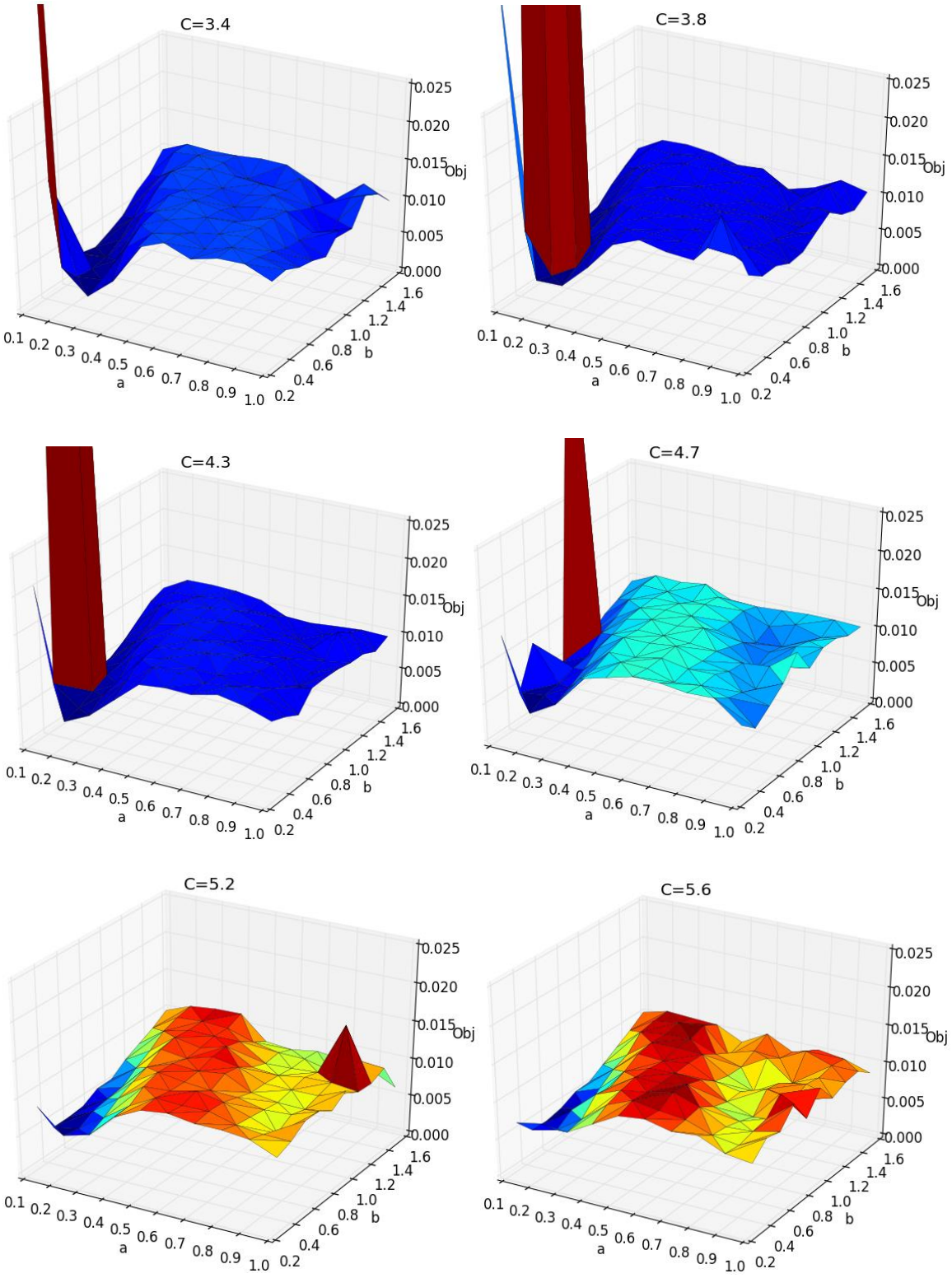
$$OF = f(a_{geom}, b_{geom}, b_{KE\ y-intercept})$$

Therefore, by constraining the additional y-intercept parameter, one can uniquely define a relationship between the shaped charge geometry and the kinetic energy profile of the resulting jet. Unfortunately, this parameter cannot be established before to the shaped charge's detonation. There is no way to know the magnitude of the jet kinetic energy *a priori*. Since the only way to measure this value is after the detonation, this is a natural setup for a multiple objective function optimization problem. However, since this study focuses solely on single objective problems, this next step is left as future work.

5.2 REGRESSION ANALYSIS

A secondary group of studies was conducted using the linear regression metric to gauge to the parameter space. The procedure for this study is like the first however no gradient schemes are employed.

The results from a multidimensional parameter study are shown below.


C=3.4
C=3.8
C=4.3
C=4.7
C=5.2
C=5.6
Obj
a
b

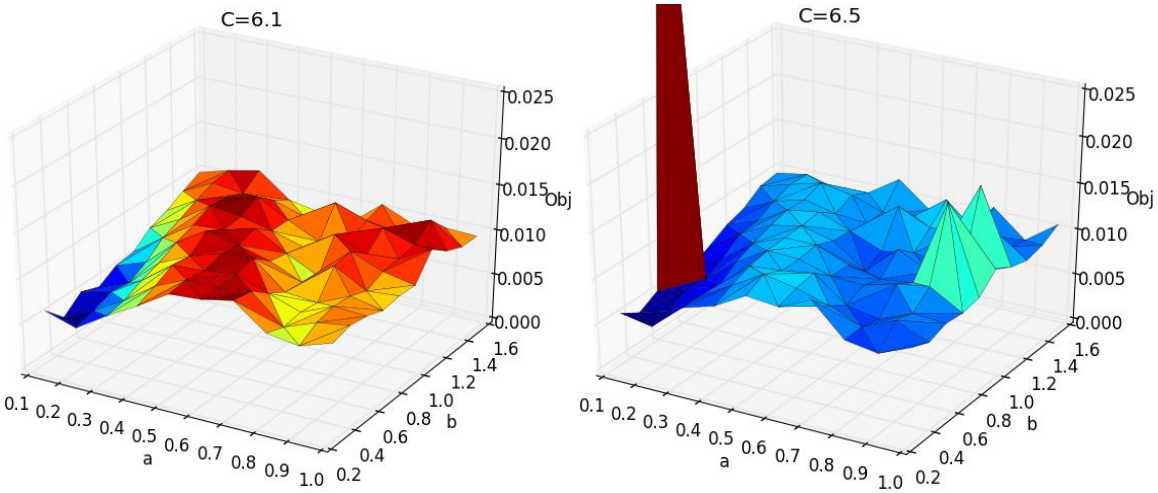


Figure 20: Surface plots of a, b versus linear regression *OF* at different c values. The *OF* axis has been changed to properly compare the solution surface. Original plots are shown in the appendix.

These results are interesting because they strongly resemble the surfaces produced by the normalized angle criterion. This suggests that the behavior of the objective function is governed by the physics of the CTH simulation and not the mathematical processes of the post-processing.

Additionally, one can run a Soga optimization across them domain. This result is displayed below.

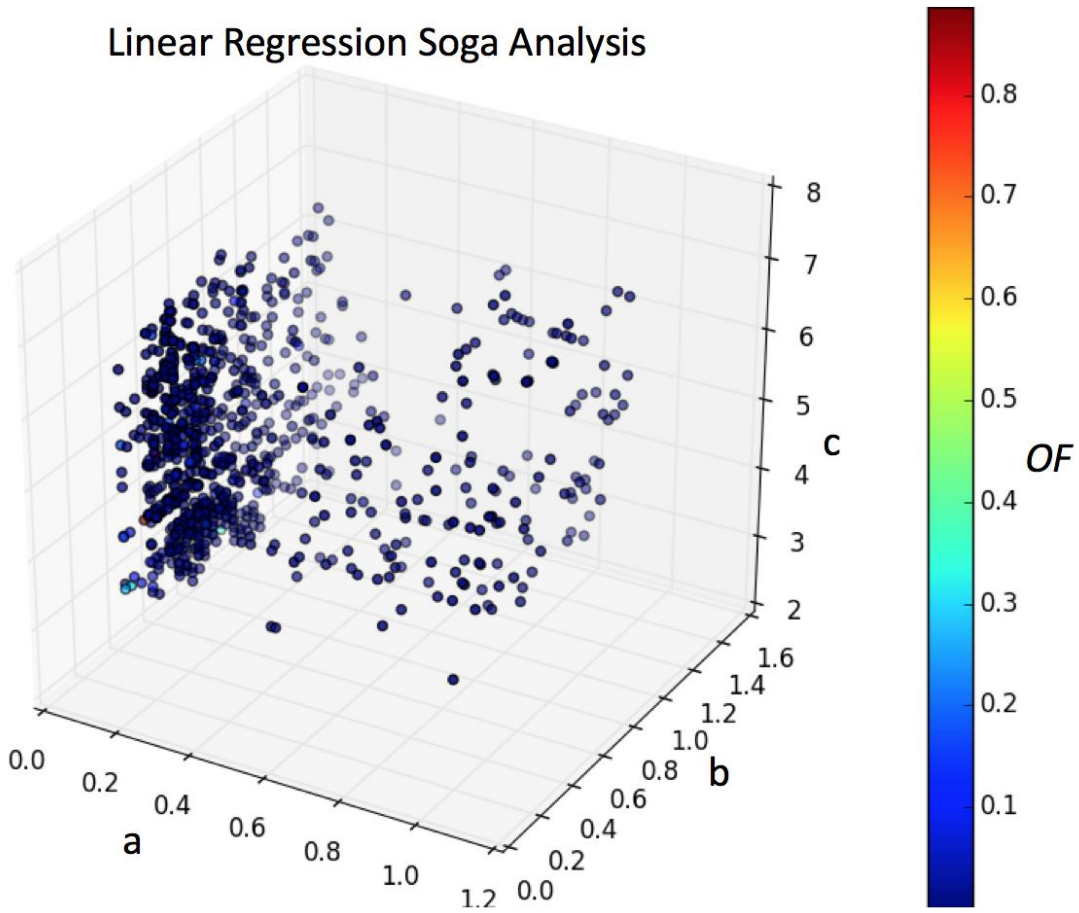


Figure 21: A scatterplot illustrating the trial points chosen by the Soga algorithm. The color scale displays the value of the linear regression objective function

These results are also comparable to the Normalized Angle Soga run. Once again, it appears that the c parameter does not influence the overall functionality of the system. This is best illustrated in the Soga scatter plot by the column of data points that are chosen.

All together, these results either suggest that the linear regression objective function is much more related to the normalized angle objective function or that the physics in the CTH simulation dominates the functionality of these objective functions. This point is reinforced by the two-variable parameter study shown below.

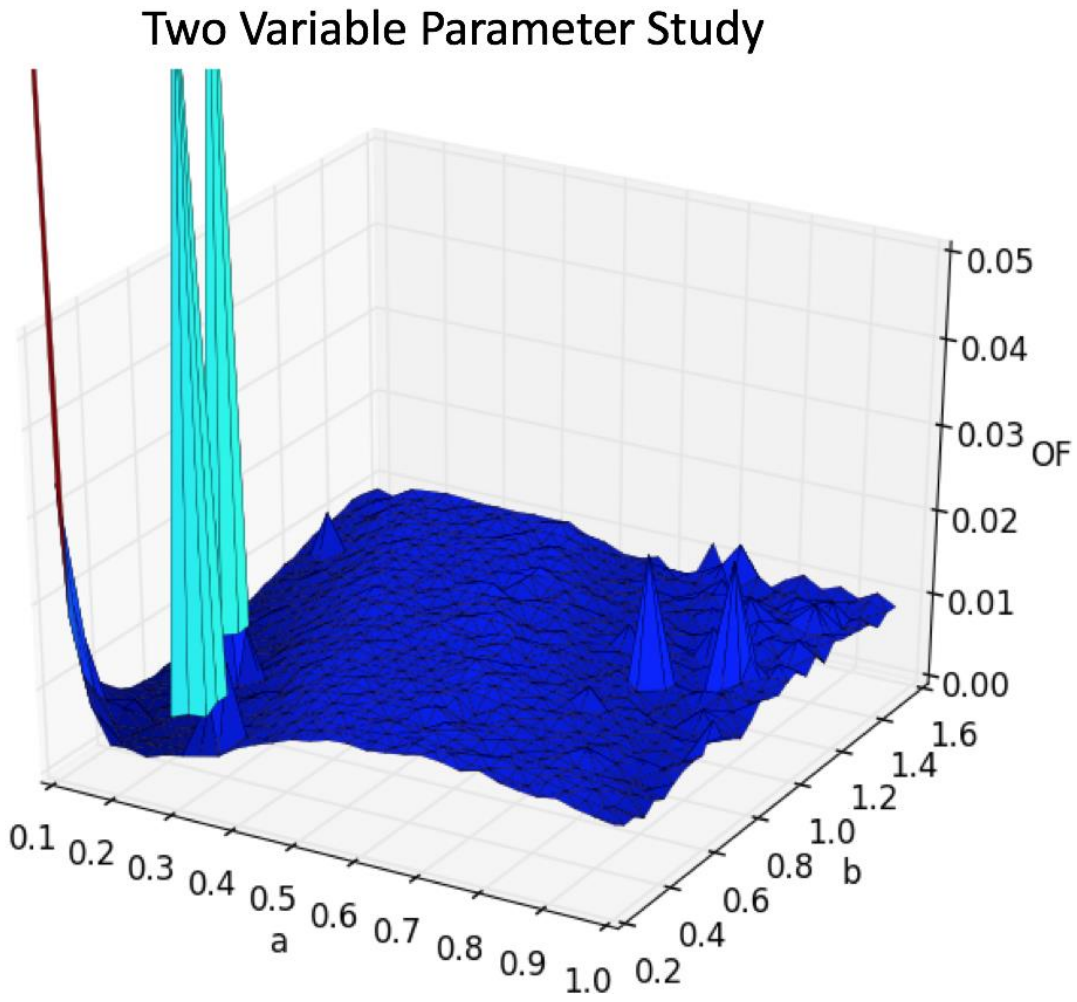


Figure 22: The two-variable surface calculated from the linear regression objective function.

Once again, this surface strongly resembles the two-variable surface that was generated by the Normalized Angle criterion.

6 CONCLUSION AND FUTURE WORK

In conclusion, the way that the shaped charge liner was parameterized and the methodology used to architect the objective functions has produced a relationship between parameters defining liner geometry. This relationship outlines how different geometries can produce similar shaped charge jets.

This result was surprising as the objective function minima was not unique to the reference geometry used originally used to create it. However, by looking at the kinetic energy profiles generated by the family of solutions, one can conclude that the missing parameter producing the degeneracy is related to the maximum kinetic energy in the jet. This sets up a multiple objective optimization problem where both the slope and the y-intercept of the KE profile are optimized.

While the result hypothesized by this work was not reached, an optimization methodology has been established that can be wieldy applied to the design of future energetic material systems. Ideally, one could choose an objective function (such as maximum kinetic energy) and the optimization would produce the ideal design. To this end, future work can be done to improve the methodologies that were used.

One can always introduce techniques that improve computational efficiency. With regards to the CTH simulations, resolving the computational mesh with AMR would give the user the ability to increase resolution around areas in question without sacrificing computational time. Additionally, the ability to run individual CTH simulations in parallel would greatly impact computational efficiency.

However, most improvements primarily regard the parameterization and post-processing techniques used to create an objective function. Future work should increase

the number of parameters that control the shaped charge's liner. This could be as simple as increasing the order of the polynomial that defines the liner however this approach will always impose a constrained family of shapes on the liner. Alternatively, one can define the liner with several points but leave the x and y locations variable. This approach would yield a wider range of solutions.

Work can be done to develop more sophisticated objective functions. One can always choose to optimize a metric when designing a liner however, it would be interesting to exploit different optimization techniques that explore multiple objective functions. An easy next step could be to exert control over both thickness and length of the jet.

Finally, one could study the impact of different EOS and strength models has on the geometry of a shaped charge. This would combine many different parametric inputs, controlling EOS models and liner geometry, with a set of outputs measuring jet kinetic energy or shape as well as thermodynamic states of the jet.

7 BIBLIOGRAPHY

Adams, B. M. (n.d.). *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.9 User's Manual.*

Ahmad Shukri Nazri, e. a. (2017). *Engineering Calculus - 2.9 Numerical Methods: Newton-Raphson.* Penerbit Universiti Sains Malaysia.

Baker, E. T. (2011). *Ballistics 2011 - 26th International Symposium on Ballistics, Miami, Florida, 12-16 September 2011, Volume 1 and 2 - 17.3 Detonation Velocity Dependence on the Explosive Massdensity in the Numerical Simulation.* DEStech Publications.

Beaver, L. E. (2017). *A Parametric Investigation and Optimization of a Cylindrical Explosive Charge.* Marquette University.

Coddet, C. (2015). Metal, Ceramic and Composite Materials - Selected, Peer Reviewed Papers from the 2015 International Conference on Metal, Ceramic and Composite Materials (ICMCCM 2015), January 24-25, 2015, Shanghai, China - 24.4.1 Flash X-ray Radiography (FXR) Setup.

Cormen, T. H. (2009). *Introduction to Algorithms (3rd Edition) - B.3 Functions.* . MIT Press.

Crawford, D. (n.d.). *CTH Course Notes.* Sandia National Laboratories.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*.

Ian Woodley, C. C. (n.d.). *Ballistics 2016 - 29th International Symposium on Ballistics, Edinburgh, Scotland, UK, 9-13 May 2016 - 162.6 Anisotropic Distribution of Dislocation Densities in the Axial Plane of the Jet.* DEStech Publications.

J Haslinger, e. a. (2003). Introduction to Shape Optimization - Theory, Approximation, and Computation - 4.1 Gradient Methods for Unconstrained Optimization. *Society for Industrial and Applied Mathematics.*

McGlaun, J. M. (1990). CTH: A three-dimensional shock wave physics code. *International Journal of Impact Engineering*.

Michaeli, W. (2003). *Extrusion Dies for Plastics and Rubber - Design and Engineering Computations (3rd Edition) - 4.7.3.4 Evolutionary Methods.* Hanser Publishers.

Nikos D. Plevris, e. a. (2013). *Design Optimization of Active and Passive Structural Control Systems - 11.3.1.1 Evolutionary Optimization, Terms, and Definitions.* IGI Global.

P.Y. Chanteret, F. J. (1984). *Quasi Non-Stretching Hypervelocity Jets.*

Rao, S. S. (2009). *Engineering Optimization - Theory and Practice (4th Edition) - B.6 Computer Programs for Modern Methods of Optimization.* John Wiley & Sons. .

Rhinehart, R. R. (2018). *Engineering Optimization - Applications, Methods, and Analysis - 4.3.1 Newton's Methods.* John Wiley & Sons.

SGI ICE X (TOPAZ) USER GUIDE. (n.d.). Retrieved June 4, 2019, from DoD Supercomputing Resource Center: https://www.erdc.hpc.mil/docs/topazUserGuide.html

Shilov, G. E. (1977). *Linear Algebra - 4.6 The Range and Null Space of a Linear Operator.* Dover Publications.

Simon G. Edwins, e. a. (2002). *Encyclopedia of Vibration, Volumes 1-3 - Boundary Conditions.* Elsevier.

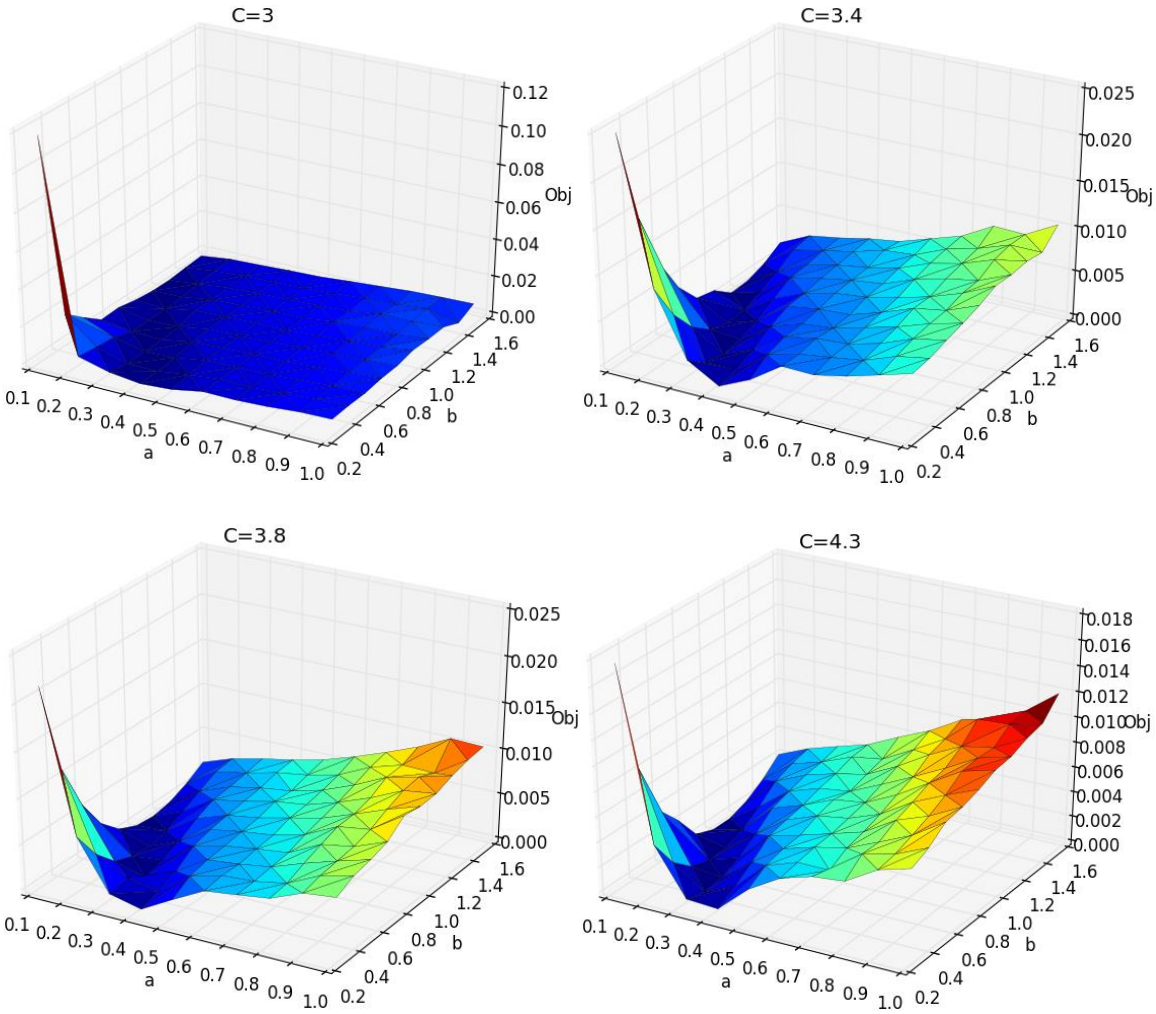
U.S. Army Materiel Command. (n.d.). *Engineering Design Handbook - Elements of Terminal Ballistics, Parts One and Two: (AMCP 706-160, 706-161) - 2.15 Introduction.* Retrieved from https://app.knovel.com/hotlink/pdf/id:kt00UCTD91/engineering-design-handbook/advantages

Wickert, e. a. (2013). *Ballistics 2013 - 27th International Symposium on Ballistics, Freiburg, Germany, 22-26 April 2013, Volume 1 and 2 - 162.3.2 Numerical Simulation on Penetration Process of EFPs.* DEStech Publications.

8 APPENDIX

Original Plots

Below are the original parameter study plots generated with the normalized angle objective function.



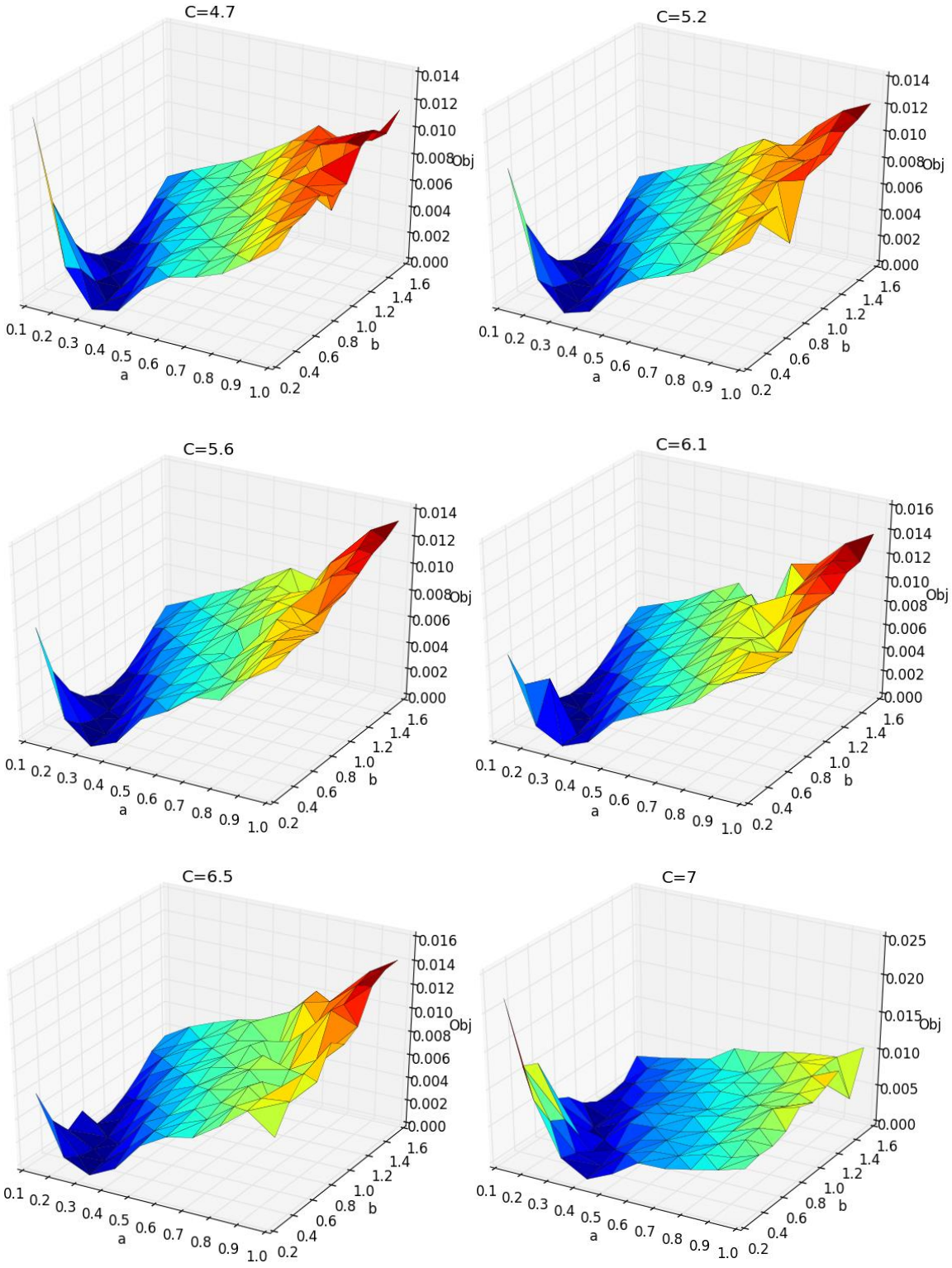
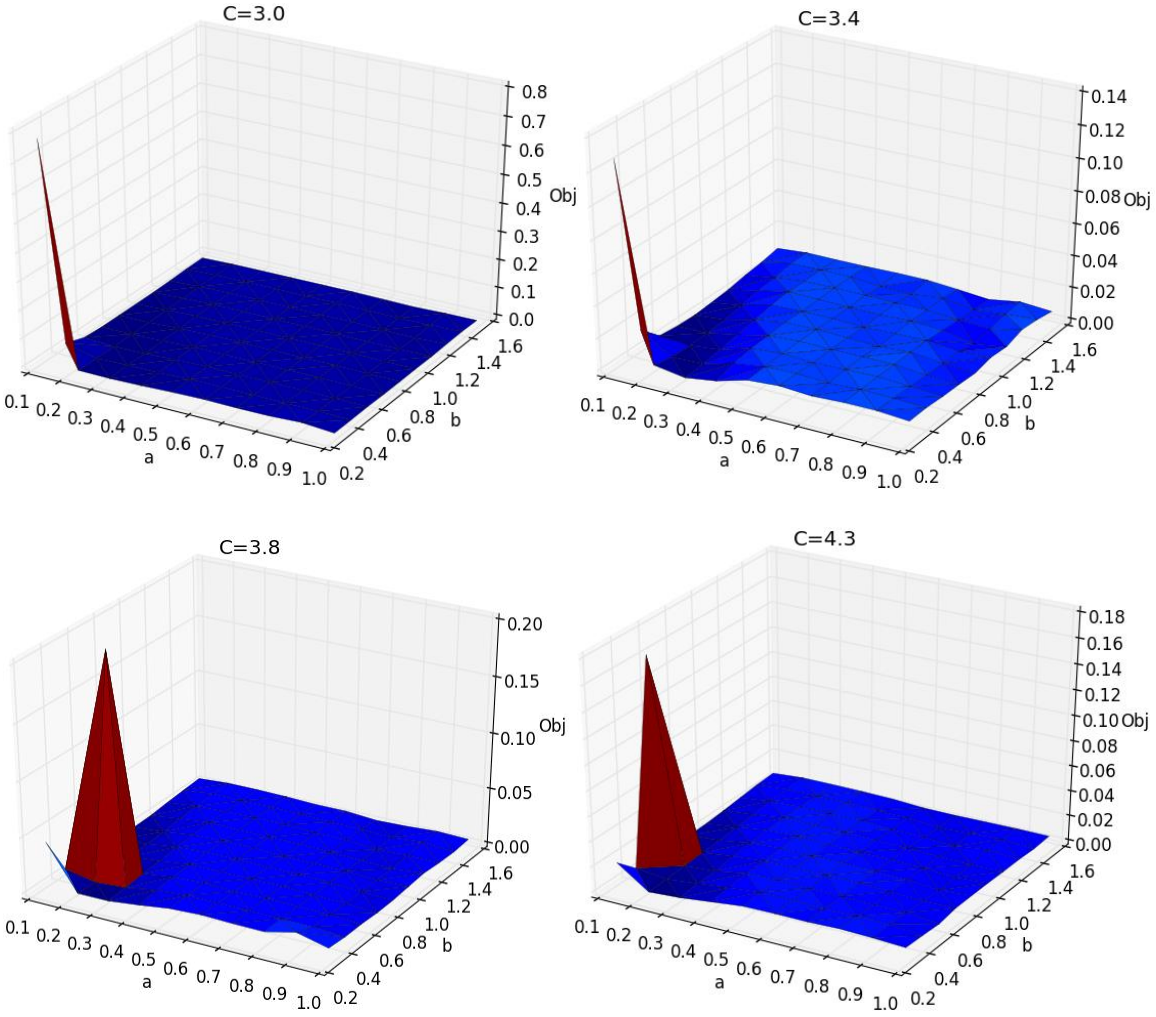


Figure 1-A: The original, non-scaled surface plots of the normalized angle objective function

Below are the original parameter study plots generated with the linear regression objective function.



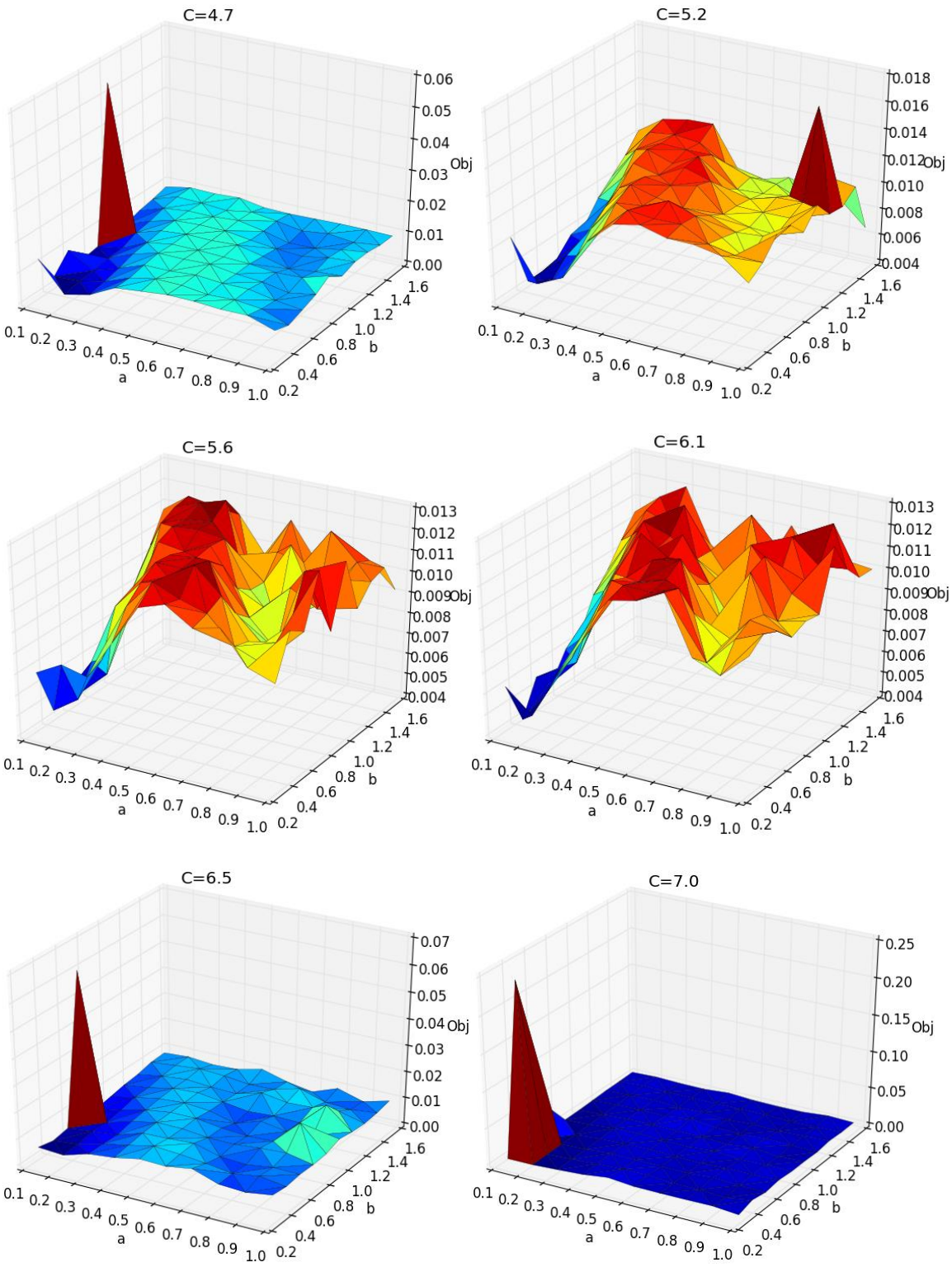


Figure 2-A: The original, non-scaled surface plots of the linear regression objective function

Conservation of Mass

The following derivation will prove that due to the way the shaped charge liner is parameterized, mass will always be conserved for variable a, b and c and constant Δh. Displayed below is the parabolic geometry that defines the shaped charge liner.

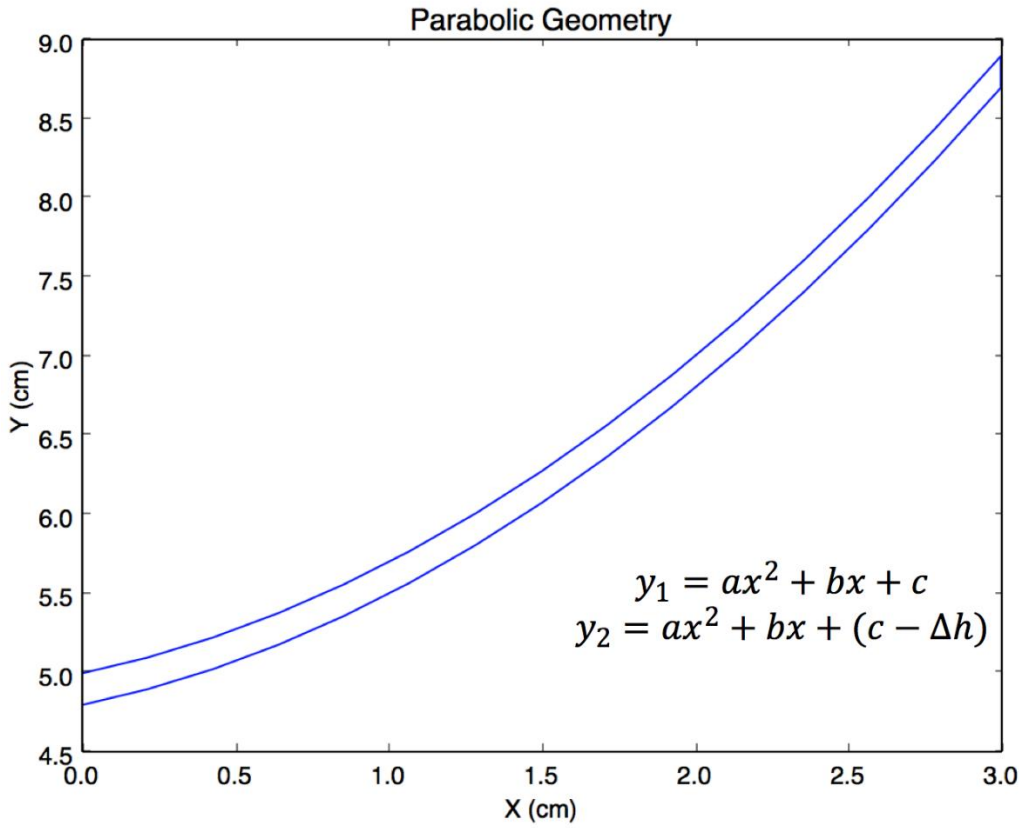


Figure 3-A: A plot of the parabolic shaped charge liner

Since the material remains constant then the density thought the liner is constant as well. Therefore, when one discusses conservation of mass, one discusses conservation of "volume" or area contained between the two curves.

To calculate the area contained between both curves one can set up an integral as following:

$$A = \int y_1 \, dx - \int y_2 \, dx$$

Inserting known values, this becomes:

$$A = \int_0^3 (ax^2 + bx + c) \; dx - \int_0^3 (ax^2 + bx + c - \Delta h) \; dx$$

By inspection, the identical terms cancel and this simplifies down to:

$$\mathrm{A} = \; 3 * \Delta h$$

Therefore, one can concluded that the area has no dependence on variables a, b and c. This result may seem very counter-intuitive at first however it can be explained simply. If one imagines a rectangle whose length is separated into infinitesimal strips, one can offset these strips and create any "double profile" if it is separated by a constant value. This is exactly the case presented by the double parabola problem. In fact, the area derived from the two parabolas is identical to the area of a rectangle.

Input Scripts

The entire optimization study is launched with a PBS launch script. This script is responsible for submitting the optimization job to the PBS queue and for establishing the

environment for the shell scripts to run in. This includes setting a path for Dakota, CTH and loading the appropriate python modules.

This script is displayed below.

```
 1 #!/bin/bash
 2 #PBS -l select=10:ncpus=36:mpiprocs=5
 3 #PBS -l walltime=18:00:00
 4 #PBS -q standard
 5 #PBS -A ERDCS97270PET

 6 #PBS -N R_Squared
 7 #PBS -j oe
 8 #
 9 source ${MODULESHOME}/init/bash
10 cd $PBS_O_WORKDIR
11
12 module load dakota/6.6_parallel
13 module load cth/12.0
14
15 module swap compiler compiler/gcc
16 module load costinit
17 module load python3/gnu/3.6.7
18 module load numpy/gnu/1.14.2
19 module load scipy/gnu/1.1.0
20
21 export CTHPATH="/p/home/apps/cth/CTHV12.0"
22 export CTHBINPATH="/p/home/apps/cth/CTHV12.0/bin"
23 export CTHMPIBINPATH="/p/home/apps/cth/CTHV12.0/bin"
24 export CTHDATA="/p/home/apps/cth/CTHV12.0/data"
25 export MPIRUN="mpiexec_mpt -np"
26 export ncpus=$BC_MPI_TASKS_ALLOC
27 export input=dakota_cth_SC.in
28 export output=output.out
29
30 $MPIRUN $ncpus dakota -i $input -o $output
```

Lines 2 through 7 set PBS variables. These include the wall time, the priority and name of the job. In addition, one must specify the number of nodes and the number of parallel processes allowed on each node. Multiplying these two numbers together one can figure out the total number of CTH jobs that can be run at once. As previously defined in

the body of this work, this is the concurrency of the optimization. Therefore, it cannot exceed the concurrency limit set by the optimization scheme.

In this example, the number of nodes is 10 and the number of parallel processes allowed on each node is 5. In total, this supports 50 concurrent simulations. The ncpus value represents the number of cores per node. For Topaz, this is set constant at 36. Lines 12 through 19 load the modules necessary for the optimization. These include Dakota, CTH and the various python modules. Lines 21 through 24 set the paths for CTH. Finally, the rest of the sets the global variables and Dakota input script.

Once the optimization study finally starts running, Dakota is initiated via the input script. The following script is a broad representation of the various controls used to initiate the Dakota optimization.

```
 1 environment
 2  tabular_data
 3     tabular_data_file = 'cth_dakota_simulations.dat'
 4
 5 method,
 6      multidim_parameter_study
 7          partitions = 9 9 9
 8
 9      conmin_frcg
10          convergence_tolerance = 1e-8
11          max_iterations = 100
12
13      soga
14          max_iterations = 1000
15          population_size = 50
16
17 variables,
18         continuous_design = 3
19         cdv_initial_point    .5    1.00  3.5
20         lower_bounds         0.1   0.2  3.00
21         upper_bounds         1.0   1.50  7.0
22         descriptor           'A'   'B'   'C'
23
24 interface,
25
26         fork,
27             parameters_file = 'params.in'
```

```
28             results_file    = 'results.out'
29             work_directory
30               named = 'workdir'
31               directory_tag
32               directory_save
33             file_save #Comment out later
34             analysis_driver = 'cth_simulator.sh'
35
36 responses,
37         num_objective_functions = 1
38         no_gradients
39    numerical_gradients
40      method_source dakota
41      interval_type central
42         no_hessians
```

Since this is a broad representation of what was used lines must be commented out according to the type of optimization being conducted. Lines 6-15 control the type of optimization scheme. One must comment out everything except lines outlining the methodology one desires.

The next section controlling the type of variables being optimized generally stays the same from optimization to optimization. However, the command specifying an initial point at line 19 must be commented out when not using a gradient scheme.
The “environment” and “interface” sections control the directory structures of an optimization and how data gets written to files. Therefore, this remains unchanged throughout all optimization studies.

Finally, the last section controls settings regarding the objective function. As was outlined in previous sections, line 37 will always stay the same as this study focuses on optimizing one objective function. Depending on whether one is conducting a gradient study, lines 39-41 alternate getting commented out with line 38. Hessian analysis was not performed in this work.

As one sees in line 34, what Dakota will be interacting with is a program called *cth_simulator.sh*. This shell script is meant to act as the CTH "black box." Dakota will pass its iterative parameters to it through a file called *params.in* and it will receive the objective function from *results.out* (lines 27 and 28). This shell script is shown below.

```
 1 #!/bin/bash
 2
 3 ############
 4 #Geometry and CTH step
 5 cp ../cth.template.in ../Geometry.template ../CurveFit
     ../cth.processor.sh ./
 6 dprepro params.in Geometry.template Geometry
 7 ./Geometry 1
 8 ./cth.processor.sh
 9 #############
10 #PATCHING STEP
11 #if maximum.out does not exist cth failed. Rerun with noise
     params
12 FILE=maximum.out
13 if test -f "$FILE"; then
14     echo "$FILE does exist"
15     max=$(cat maximum.out)
16     echo 'THIS IS THE MAXIMUM'
17     echo $max
18     #Formats results.out
19     echo $max"     f" >> results.out
20
21 else
22     echo "$FILE does not exist"
23     rm rscth
24     rm spcth
25     rm *.jpg
26     rm shape.in
27     rm hscth
28     rm octh
29     rm cthout.
30     rm *.dat
31     ./Geometry 2
32     ./cth.processor.sh
33
34     if test -f "$FILE"; then
35        max=$(cat maximum.out)
36        echo 'THIS IS THE MAXIMUM ON SECOND TRY'
37        echo $max
38        echo $max"     f" >> results.out
39
40     else
```

```
41        echo "$FILE STILL DOES NOT EXIST"
42        rm rscth
43        rm spcth
44        rm *.jpg
45        rm shape.in
46        rm hscth
47        rm octh
48        rm cthout.
49        rm *.dat
50        ./Geometry 3
51        ./cth.processor.sh
52        max=$(cat maximum.out)
53        echo 'THIS IS THE MAXIMUM ON THIRD TRY'
54        echo $max
55        echo $max"     f" >> results.out
56     fi
57 fi
```

The *cth_simulator.sh* script is divided in two main parts. Lines 4-8 set up a regular Dakota iteration. When Dakota launches a new iteration by running this script, it creates a new work directory to run the individual simulation. Therefore, line 5 goes back into the original directory and copies the necessary files.

Line 6 exploits the *dprepro* program that is built into Dakota. This program passes the parameters that Dakota outputs into the *Geomoetry.template* file creating the *Geometry* executable. Once executed, a set of files containing the coordinate defining the liner and corresponding explosive shape are created.

The *cth.processor.sh* script is executed. This script is responsible for the actual CTH simulation and post-processing. It will be shown later.

The second portion of *cth_simulator.sh* concerns the existence of a *maximum.out* file. As was explained in the body of the work, the existence of this file indicates if CTH simulation ran to completion or if it was terminated due to a time step issue. Therefore, lines 12 and 13 test the validity of the simulation. If there are no problems, the objective function is printed to screen and a *results.out* file is created for Dakota to iterate on. The

“f” that is inserted after the *OF* communicates that what is displayed is a floating-point number. The simulation continues smoothly.

However, if the conditional at line 13 fails, the CTH simulation will be rerun with new values generated by the *Geometry* executable. Before this happens, an error message is printed to screen and the CTH output files are removed so that new files can be created. The *Geometry* and *cth.processor.sh* scripts are executed as usual with the exception that the flag “2” is used when generating the liner points. This indicates it is the second time CTH is being run and the corresponding adjusted values should be used.

This test is repeated a second time. A maximum of three CTH simulations can be run before the program “gives up” and the optimization fails.

Displayed below is the *Geometry.template* python script.

```
 1 #!/usr/bin/env python3
 2
 3 import sys
 4 import numpy as np
 5 ############################
 6 #Conditional For Patch
 7 ############################
 8 runtype = sys.argv[1]
 9 if runtype == 1:
10 ####################
11 #VARIABLES
12 ####################
13 #Parabolic curve
14  a = {A}
15  b = {B}
16  c = {C}
17 elif runtype == 2:
18  a = {A}+np.random.uniform()/10000
19  b = {B}+np.random.uniform()/10000
20  c = {C}+np.random.uniform()/10000
21
22 else:
23  a = {A}+np.random.uniform()/10000
24  b = {B}+np.random.uniform()/10000
25  c = {C}+np.random.uniform()/10000
26
27 t = .2
```

```
w = c-t

####################
#UPPER LINE Liner
with open('Xval', 'w') as f:
    for item in xupper:
        f.write("%s\\n" % item)

yupper = a*xupper**2+b*xupper+c

with open('Yval', 'w') as f:
    for item in yupper:
        f.write("%s\\n" % item)

#####################
#LOWER LINE Liner
xlower = xupper[::-1]

with open('Xval', 'a') as f:
    for item in xlower:
        f.write("%s\\n" % item)

ylower = a*xlower**2+b*xlower+w
with open('Yval', 'a') as f:
    for item in ylower:
        f.write("%s\\n" % item)

#########################
#Creates Coordinate file

with open('Coordinates', 'w') as file3:
        with open('Xval', 'r') as file1:
                with open('Yval', 'r') as file2:
                        for line1, line2 in zip(file1, file2):
                                print( line1.strip(),
    line2.strip(), file = file3 )

############################
#Explosive
############################
exlower = xupper

with open('xexplosive', 'w') as f:
    for item in exlower:
        f.write("%s\\n" % item)

eylower = a*exlower**2+b*exlower+w
with open('yexplosive', 'w') as f:
    for item in eylower:
        f.write("%s\\n" % item)

with open('Explosive', 'w') as file3:
        with open('xexplosive', 'r') as file1:
```

```
80                 with open('yexplosive', 'r') as file2:
81                         for line1, line2 in zip(file1, file2):
82                                 print( line1.strip(),
    line2.strip(), file = file3 )
```

Lines 3 and 4 import the necessary modules for the script to run. Lines 8 and 9 identify what type of simulation is being run and point to the appropriate parameters that are used to generate the shaped charges geometry. These parameters are defined at lines 14 through 25. One can see that for the second and third CTH runs, the original variables are perturbed by a small random value.

Dakota's *dprepro* searches for variables in brackets and substitutes its values accordingly. An idiosyncrasy of *dprepro* is that it also eliminates the first occurrence of the forward slash. For this reason, on lines 34, 40, 48, 53, 71 and 76 a second forward slash is inserted.

The upper and lower liner geometries are defined by parabolas on lines 36 and 50. Naturally, the order that these points are listed in is important therefore, the x values on the lower parabola are reversed on line 44. Lines 56 through 62 create a file named *Coordinates* with all of the generated points.

Similarly, lines 67 through 76 yield the points needed to define the upper surface of the explosive and lines 78 through 82 create a file named *Explosive* with these points. Now one must open the *Coordinates* and *Explosive* file and insert the points in the *cth.template.in* file. This is done in the first steps of the *cth.processor.sh* script displayed below.

```
1 #!/bin/bash
2
3 ##############
4 #Add the word "point" to Coordinates then creates POINTS
```

```
for number in {1..30}
do
line=$(sed -n "$number p" Coordinates)
point="point $line"
echo "$point" >> POINTS
done

for number in {1..15}
do
line=$(sed -n "$number p" Explosive)
point="point $line"
echo "$point" >> EXPLOSIVE
done
################################
#Adds POINTS to cth.template.in
li=$(awk '/\*insertliner/{ print NR; exit }' cth.template.in)
awk 'NR>'$li'{while((getline a < "POINTS") > 0){print a}}1'
  cth.template.in>> liner

li=$(awk '/\insertexplosive/{ print NR; exit }' liner)
awk 'NR>'$li'{while((getline a < "EXPLOSIVE") > 0){print a}}1'
  liner>>    shape.in

rm Coordinates  Xval Yval liner POINTS xexplosive yexplosive
  Explosive EXPLOSIVE
##############
#Executes
echo cth shape.in
$CTHBINPATH/cth shape.in >& cthout.$num
###############
#Clean up excess data dump files
for file in $(ls *.dat); do if [[ ! -s $file ]]; then rm $file;
   fi; done


#FOR DATA DUMP
#Sorts files and combines processor files into 1 timestep file
files=$(ls DataDump*)
fileindex=$(echo "${files}" |cut -c 9-14)
index=$(echo "$fileindex" | xargs -n1 | sort -u | xargs)

for i in $index; do
    list=$(echo "DataDump$i")
    cat $list* >> File.$i

   if [ $i -eq 0 ]; then
      rm File.$i
   else
      sed -i '1d' File.$i
#     awk '{ print $1 }' File.$i >> XLOC.File.$i
      awk '{ print $2 }' File.$i >> YLOC.File.$i
#     awk '{ print $3 }' File.$i >> DX.File.$i
#     awk '{ print $4 }' File.$i >> DY.File.$i
```

```
54       awk '{ print $5 }' File.$i >> KEFile.$i
55 #########
56 #Optimization STEP
57       ./CurveFit KEFile.$i YLOC.File.$i >> maximum.out
58 ###########
59       rm *File.$i
60    fi
61 done
```

Lines 4 through 24 accomplish this task. Due to CTH input deck format, the first section (4-17) adds the word “point” to each line in the *Coordinates* and *Explosive* files. Then, lines 20 through 24 search the CTH input deck for the *insertliner* and *insertexplosive* flags and insert the geometries respectively. This process changes *cth.template.in* to the final input deck: *shape.in*.

After cleaning up excess files, the CTH simulation is launched on line 30 with the appropriate input deck.

The CTH simulation will produce a data dump files that divide the computational domain at the beginning and at the end of the simulation. Lines 32 through 40 sort through these files, and combine them into one complete file. Lines 46 through 54 break the complete dat file into the required data sets for the optimization. In this case, the y location and respective kinetic energy is used. If one creates a more involved objective function one could uncomment lines 50, 52 and 53 to post process the full data set.

Finally, on line 57 the *CurveFit* python script is launched post-processing the data into the desired objective function. This result is written on the *maximum.out* file. The *CurveFit* script is shown below.

```
1 #!/usr/bin/env python3
2
3 import sys
4 import numpy as np
5 from scipy.optimize import curve_fit
```

```
 6 from scipy import stats
 7
 8 xdata = np.loadtxt(sys.argv[2])
 9 ydata = np.loadtxt(sys.argv[1])
10
11 ynorm = ydata/203862000000.0 #Obtained from Reference Run
12 xnorm = xdata/23.2 #Obtained from Reference Run
13
14 def func(x, a, b):
15         return a*x+b
16
17 ################################
18 #Normalized Angle
19 refangle = 0.43657940317644234 #Obtained from Reference Run
20 popt, pcov = curve_fit(func, xnorm, ynorm)
21 theta = np.arctan(popt[0])*2/np.pi
22 print((refangle-theta)**2)
23
24 ####################
25 #R_SQUARED
26 A = 0.81826164 #Obtained from Reference Run
27 B = -0.78969017 #Obtained from Reference Run
28 yref = func(xnorm, A, B)
29
30 slope, intercept, r_value, p_value, std_err =
     stats.linregress(ynorm, yref)
31
32 print(1-r_value**2)
33
34 #For Scaled Version
35 OB = (1-r_value**2)**.5
36 print(OB)
```

Lines 3 through 6 load the necessary modules and lines 8 and 9 load the data fed in by the *cth.processor.sh* script. Subsequently, this data is normalized as previously described. Depending on what objective function the study demands, the second and third sections are commented out accordingly. At the end of the routine, the objective function is printed and this result ends up written on the *maximum.out* file.

Altogether, the full schematic of how all shell scripts and input decks interact is illustrated below.

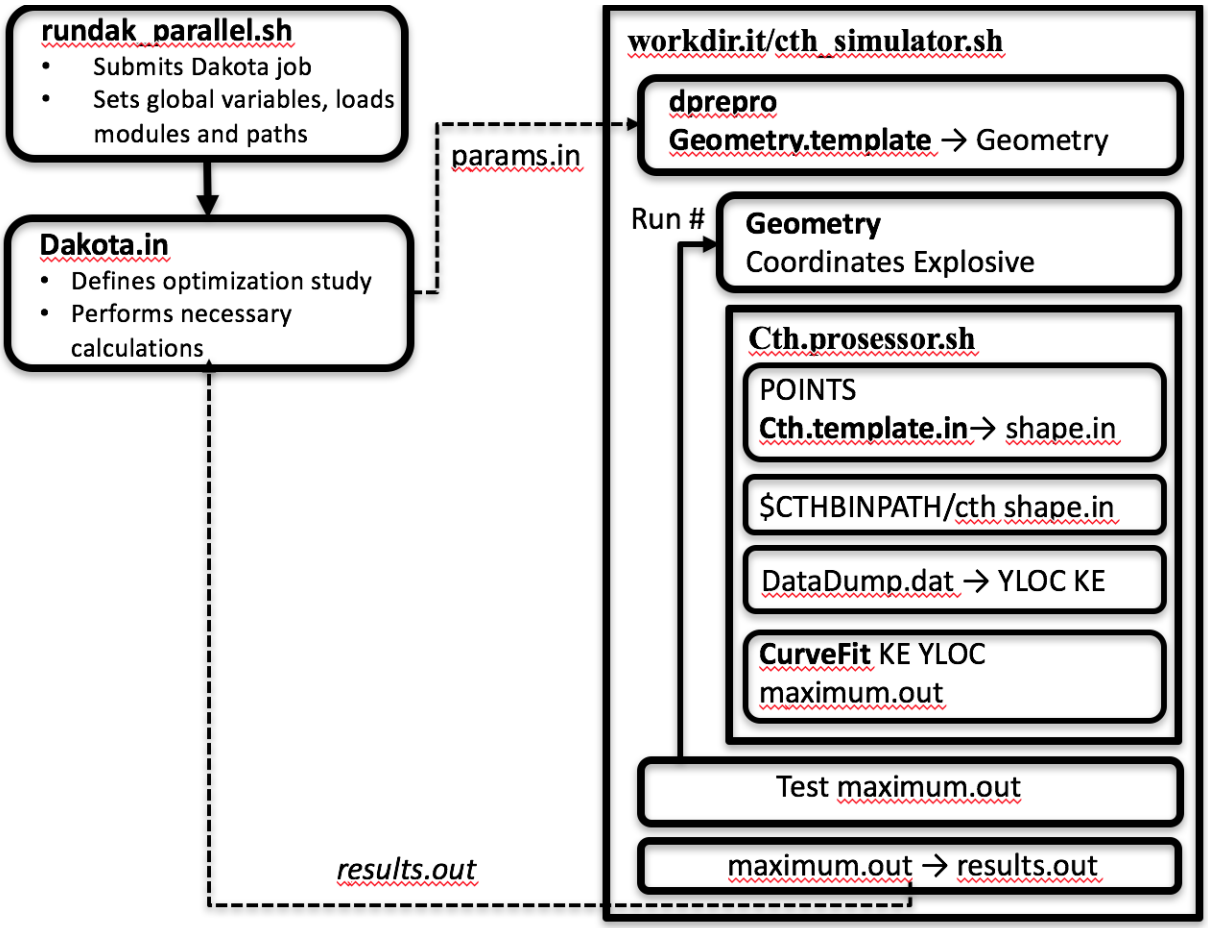


Figure 4-A: A full schematic of the optimization study

Finally, and perhaps most importantly, the *cth.template.in* script is displayed below.

```
 1 ************************************************************************
 2 *eor* cthin
 3 ************************************************************************
 4 *
 5 * CTH template script
 6 *
 7 ************************************************************************
 8 *
 9 control
10   tstop=80.e-6
11   mmp
12 endcontrol
13 *
14 ************************************************************************
15 *
```

```
mesh
  block  geometry 2dc  type e
    x0=0.0
      x1  dxf 0.03  dxl 0.03  w 3.5
      x2  dxf 0.05  dxl 0.35  w 1.5
   endx
    y0=0.0
      y1  dyf 0.25  dyl 0.05  w 3.0
      y2  dyf 0.03  dyl 0.03  w 10.0
      y4  dyf 0.1  dyl 0.1  w 50.0
    endy
    xact=0.0,1.0
    yact=0.0,5.0
  endblock
endmesh
*
**********************************************************************
*
spy
 Save("M,VOLM,VX,VY,P,KE");
 SaveTime(0, 80e-6);
 PlotTime(0, 80e-6);

 ImageFormat(1024,768);

 define main()
 {
   pprintf(" PLOT: Cycle=%d, Time=%e\n",CYCLE,TIME);
   XBMirror(ON);
   XLimits(-5,5);
   YLimits(0,20);

   Image("shap-Pressure");
    Window(0,0,0.75,1);
    MatColors(DIM_GRAY,LIGHT_GRAY);
    Plot2DMats(0.0001);
    ColorMapRange(2e6,1e9,LOG_MAP);
    ColorMapClipping(ON,OFF);
    Label(sprintf("Pressure Time=%0.2f |c03BC|cs",TIME*1.E6));
    Plot2D("P");
    Draw2DTracers(3);
    Draw2DMatContour;
    DrawColorMap("(dyn/cm^2^)",0.75,0.4,0.9,0.9);
   EndImage;

   XLimits(-20,20);
   YLimits(0,74);

   Image("shap-Mats");
    Window(0,0,0.75,1);
    MatColors(PERU,YELLOW);
    Label(sprintf("Materials Time=%0.2f |c03BC|cs",TIME*1.E6));
    Plot2DMats;
```

```
 69     MatNames("Copper","PBX-9404");
 70     DrawMatLegend("",0.75,0.2,0.99,0.9);
 71    EndImage;
 72
 73    Image("shap-Density");
 74     Window(0,0,0.75,1);
 75     ColorMapRange(1,9);
 76     ColorMapClipping(ON,OFF);
 77     Label(sprintf("Density Time=%0.2f |c03BC|cs",TIME*1.E6));
 78     Plot2D("DENS");
 79     DrawColorMap("(dyn/cm^2^)",0.75,0.4,0.9,0.9);
 80    EndImage;
 81
 82    Image("shap-1d-YV");
 83     Fix1D(0,0,0,74);
 84     Label(sprintf("Y-Velocity Time=%0.2f |c03BC|cs",TIME*1.E6));
 85     Plot1D("VY",ON,AUTOSCALE);
 86    EndImage;
 87
 88    Image("shap-1d-dens");
 89     Fix1D(0,0,0,74);
 90     Label(sprintf("Density Time=%0.2f |c03BC|cs",TIME*1.E6));
 91     Plot1D("DENS",ON,AUTOSCALE);
 92    EndImage;
 93
 94    XLimits(0,0.05);
 95
 96   DataOut("DataDump","KE");
 97   DataOutFilter("VY",3e5,1e99);
 98  }
 99
100  SaveHis("GLOBAL,VX,VY,P,DENS");
101  SaveTracer(ALL);
102  HisTime(0,1.e-8);
103
104  define spyhis_main()
105  {
106    HisLoad(1,"hscth",OFF);
107    HisImageName("shap_history");
108    Label("Y velocity at tracer 1");
109    TPlot("VY.1",1,ON);
110  }
111 endspy
112 *
113 ***********************************************************************
114 *
115 diatoms
116 *
117   package 'CU LINER - 1'
118     material 1
119     pressure 1.0e6
120     insert uds
121 *insertliner
```

```
122     endinsert
123   endpackage
124 *
125   package 'COMP A3 CHARGE - 1'
126     material 2
127     pressure 1.0e6
128     insert uds
129       point 0.0      0.0
130 *insertexplosive
131       point 3.00  5.40233
132       point 1.59385  0.0
133     endinsert
134   endpackage
135 *
136 enddiatoms
137 *
138 ************************************************************************
139 *
140 eos
141   mat1 mgrun COPPER
142   mat2 jwl PBX-9404-3
143 endeos
144 *
145 ************************************************************************
146 *
147 heburn
148   material 2  d 8.8e5  pre 1.0e12
149     dp  0.0 0.01  ti 0.0  radius 20.0
150 endheburn
151 *
152 ************************************************************************
153 *
154 *tracer
155 *  add  0.0 5.19003
156 *endtracer
157 *
158 ************************************************************************
159 *
160 convct
161   interface=high
162 endc
163 *
164 ************************************************************************
165 *
166 discard
167   material 1  denl=20 dens=1e99 volf=1e-6
168   material 1 denl=15 dens=1e99 volf=1e-6 templ=1e3
169   material -1 dens=1e-4 densl=0.0 templ=1e4
170 endd
171 *
172 ************************************************************************
173 *
174 edit
```

```
  shortt
    tim=0.  dt=5e-6
  ends
  longt
    tim=0.  dt=10000.
  endl
  restt
    time=0.  dtfreq=30e-6
  endr
endedit
*
**********************************************************************
*
mindt
  time=0.  dtmin=1.0e-11
  time=20.0e-6  dtmin=1.0e-10
endm
*
**********************************************************************
*
epdata
  matep 1  eppvm user yield 3.5e9  poisson 0.33
  mix 3
endep
*
**********************************************************************
*
fracts
  stress
  pfrac1=-15.0E9
  pfrac2=-1.0E9
  pfmix =-5.0E20
  pfvoid=-5.0E20
endf
*
**********************************************************************
*
boundary
  bhydro
    block=1
      bxbot 0
      bxtop 2
      bybot 2
      bytop 2
    endb
  endh
endb
*
**********************************************************************
```

The input deck is largely inspired by the example CTH shaped charge file. However, some small modifications were made.

First, the resolution was increased. Now there are almost seven computational cells spanning the length of the shaped charge in its non-detonated position. While this is an improvement, the standard for this type of study is to have ten computational cells within the material. The reason for this is that when CTH does strength modeling, it takes information from its nearest neighbors to develop a good value. However, if there aren't enough cells defining the material, one effectively gets a hydrodynamic solution. Additionally, a low-resolution simulation will not properly distinguish the varying geometries. Therefore, when optimizing the charge, it is possible that many different curves produce the same objective function.

However, while the resolution used for this study is less than optimal, these problems are overlooked. If indeed the resolution was so low that many curves formed the same objective function, the data would seem much more stepwise. Therefore, even if the observed data suggests that there are multiple solutions, it is unlikely to be caused by a problem with low resolution.

Insofar as the accuracy of the strength model is concerned, the resolution still is not alarming. If this was an optimization study with the ultimate scope of designing a proper shape charge then accurate strength models would be of importance. However, seeing as this work could be considered as a proof of concept, any inaccuracies in the strength models can be "swept under the rug" and considered part of the "CTH black box." This is not to say that a higher accuracy result would not improve the optimization by reducing noise. However, for the purposes of this study, it is considered unessential.

The materials that were simulated were PBX-9404 and copper. The PBX relied on a JWL model and the copper used a Mie-Gruneisen equation of state. The total simulation time was 80 microseconds and the Spyplot section was changed to include a command to output data dumps with the kinetic energy along the centerline. Here an additional discriminator was used that prevented centerline cells with material traveling slower than 3e5 cm/s to be written.

In addition, flags were added to the diatom section. These flags were used to specify where the *cth.processor.sh* script would output the trial geometries. Finally, the discard section was updated to include a broader spectrum of bad values. These primarily concern the copper's temperature, density and volume fraction.