Optimization of the Space Shuttle Exterior Fuel Tank

Concurrent Design Project
ME 519 (516)

Trenton Carpenter, carpentr@onid.orst.edu
Gabe Gassoway, gassowag@onid.orst.edu
Sherry Tucker, tuckeshe@onid.orst.edu

Oregon State University
Department of Mechanical Engineering

Friday, June 01, 2007

Table of Contents

I. Introduction

The goal of this project is to compare the optimization of NASA's Return on Investment (ROI) for the Space Shuttle's External Fuel tank using Applied Research Laboratory Trade Space Visualizer (ATSV), Microsoft Excel, and MatLab. The three different methods will be compared based on relative ease of use, computation time, insights offered, and departure from actual solution.

Trade space exploration is a key piece to the early stages of conceptual design. It is a study that examines existing designs and evaluates them against the minimum requirements of the new project. A program, or tool, was developed that enables the user to compare thousands of different designs against multiple criteria at once in order to select the optimal design. The tool is appropriately named the ARL (Applied Research Laboratory) Trade Space Visualizer (ATSV). The existing designs may be compared using a 3 dimensional plot that actually conveys a total of 7 variables at once. This is done by not only using the location of the markers in relation to the axis to convey 3 variables, but its size, color, orientation and transparency to convey an additional 4 variables. These existing designs do not have to be actual designs; they may be generated using an exploration engine.

The exploration engine is Java code that establishes the design variables, their limits, and how they are related to the other parameters for each design. For instance, an exploration engine for a cubic box would generate two random variables, the width dimension of the box and the thickness of cardboard used. These variables would then be used to calculate the amount of cardboard used to create the box, the volume inside the box, the weight of the box and, lets say, how many marbles the box can hold (based on size and strength of the box). Using the designs created by this engine the box designer could maximize some aspect, like marble capacity, while minimizing another, like cost of cardboard.

Microsoft Excel is a spreadsheet program available on most computers. The solver function in Excel uses the Generalized Reduced Gradient nonlinear optimization algorithm. An Excel add-in, Crystal Ball was used to randomly generate points that were within the design variable bounds. These designs were then checked against the other constraints and designs that met all constraints were ranked according to ROI.

MatLab is a numerical computing environment and programming language. In this project MatLab was used to randomly generate points which were then checked against the constraints. The designs that met all constraints were then ranked according to ROI.

II. Background

Optimization Problem

The optimization problem is formulated based on the original model as follows:

Maximize: ROI

Subject to:

Bounds on Design Variables:

$$0.01 \leq L_n \leq 5.0$$
$$0.50 \leq R_n \leq 2.0$$
$$0.25 \leq t1_n \leq 2.0$$
$$0.25 \leq t2_n \leq 2.0$$
$$0.25 \leq t3_n \leq 2.0$$
$$0.10 \leq h/R_n \leq 5.0$$

Volume constraint:

$$2826 \leq V_t \leq 3026 \Rightarrow |V_t - 100| - 2926 \leq 0$$

Stress and vibration constraints

$$\sigma_{e,i} \leq 4 \cdot 10^8 \Rightarrow \sigma_{e,i} - 4 \cdot 10^8 \leq 0$$
$$0.8 \leq \zeta \Rightarrow 0.8 - \zeta \leq 0$$

The objective is to maximize ROI subject to the bounds on the design variables and constraints on the tank volume and stresses. The restriction on tank volume is an equality constraint (~3000 m^3 +/- 100 m^3), which creates an interesting tradeoff: the tank volume is dependent upon three parameters (*L, R, h/R*) meaning that any two parameters can be free while the third is dependent upon the others. No restriction is placed on which parameter is chosen as dependent however. Finally, inequality constraints are placed on the maximum allowable component stress and on the first bending moment of the tank. The equivalent stress experienced by each component cannot exceed the maximum allowable stress of the material is used. Also, the first bending moment of the tank must be kept away from the vibrational frequencies experienced during launch to avoid any potential failures.

III. Methodology

The collaborative design approach taken by this team can perhaps best be described as "divide and conquer." The bulk of the ATSV work was undertaken by one individual. The Excel portion was assigned to another individual. The third person decided to work on the problem in MatLab. The problem was essentially solved separately with weekly hour long meetings to compare results, critique methods, overcome setbacks, and suggest improvements. On future projects, it would probably be beneficial for the entire team to get together and work on their sections in the same room, so that results from one portion could be checked or refined in another portion. Part of the reason this wasn't done was due to the different learning curves among the team members. The ideal solution seems to be a mix of Excel's quick small calculations and ATSV's powerful visualization tools. ATSV seems analogous to finding a needle in a haystack by laying out all the stalks side by side and the Excel solver is reaching into the stack and searching by feel.

Excel

Two methods of solving the fuel tank problem in Excel were used. The first method was the solve function available in the Tools menu. The second method is a third party add-in called CrystalBall.

Excel Solver

The solve function in Excel uses the generalized reduced gradient algorithm. The advantages to using the Excel solver are its speed and cost (it comes free with Excel). However, it is not the most powerful optimization package available. The advantages of the GRG algorithm are its speed and stability the disadvantages include a tendency to find local rather than global maxima.

Crystal Ball

Crystal Ball is a simulation tool for performing Monte Carlo analysis. The spread sheet with the optimization problem was open in Crystal Ball. The variables were assumed to be uniformly distributed across the intervals given in the problem description. The constraint cells and the ROI cells were set as outputs and the program was started. The program could only perform 3,549,900 iterations due to the amount of memory available. This took 1 hour and 25 minutes. There was too much data and Excel was not able to extract it. So, 1,000,000 iterations were run which took 24 minutes, the data was extracted and the runs that violated the constraints were discarded. Only two different valid solutions were found after 5 runs of 1,000,000 iterations.

MatLab

In an attempt to explore a different approach to solving the external fuel tank problem, MatLab was used to randomly generate designs in order to populate the problem design space. The code is similar in function to ATSV in that it generates designs and refines the bounds in which designs are created. It differs in that it needs a discrete number of designs to generate and that it automatically refines the bounds rather than the user “brushing” the design space. But what is

sacrificed in terms of control is gained through time efficient computation which means faster population of the trade space.

Code Operation

The program randomly generates values of L_n, R_n, $t1_n$, $t2_n$, and $t3_n$, between the design bounds given in the description of the problem. The program uses the following formula to create the random design parameters.

$$X=a+h(b-a)$$

Where X is the random variable, a and b are the lower and upper design boundaries, respectfully, and h is a randomly generated number that satisfies $0 \leq h \leq 1$.

The program then solves for an h-R ratio based on a constant volume of 2826 m^3. It is important to note that by doing this there is a potential for a more optimal design to be generated within the $2926 \pm 100 m^3$ range. If the h-R ratio passes between the h-R design boundaries, then all the stresses and vibration are calculated. If the resulting stresses and vibration values pass the design criteria then ROI is calculated. Now L_n, R_n, $t1_n$, $t2_n$, and t3n, hR and ROI can be stored as a usable design. If any of the design criteria is not met, then a set of new random numbers are generated and the process is repeated until the desired number of designs are reached.

The user has four inputs to this program;

n= number of USABLE designs the program will generate for each boundary refinement (explained below)

m= number of times the program executes refinements on the variable bounds, for example changing the bound of L_n from $.01<L_n<5$ (from problem statement) to $.5<L_n<2$ (tightening the bounds around the optimal design). How the program chooses these values is determined by the next two variables.

x_max= first boundary refinement width (percentage)

x_min= last boundary refinement width (percentage)

The program iterates closer and closer to the "optimal" boundary range for the design variables. Explanation of this process may be more easily conveyed through an example. Assume n=10,000, m=5, x_max=10, and x_min=1. The program generates 10,000 designs (n), within the initial boundaries defined by the problem statement. Then the top 10% (x_max) of the designs are looked at. The max and min of these top 10% are found for each design variable (L_n, R_n, $t1_n$, $t2_n$, $t3_n$ and hR). These maximums and minimums then become the new design boundaries and another 10,000 designs are generated within these new boundaries. Now the program begins a linear progression from 10% down to 1% in m steps, rounding up to the nearest integer. So the next step would be 8%. Now the top 8% are looked at and the boundaries are redefined and another 10,000 designs are generated within the new boundaries... then 6%... then... this process occurs a total of m times so the last time the top 1% (x_min) of the designs are look at and the final 10,000 designs are generated within this final boundary. This effect of boundary refinement is shown in Figure 1 for a case of 50,000 designs with 5 boundary refinements. The total number of designs generated within the design space is equal to n*m, in this case 50,000.

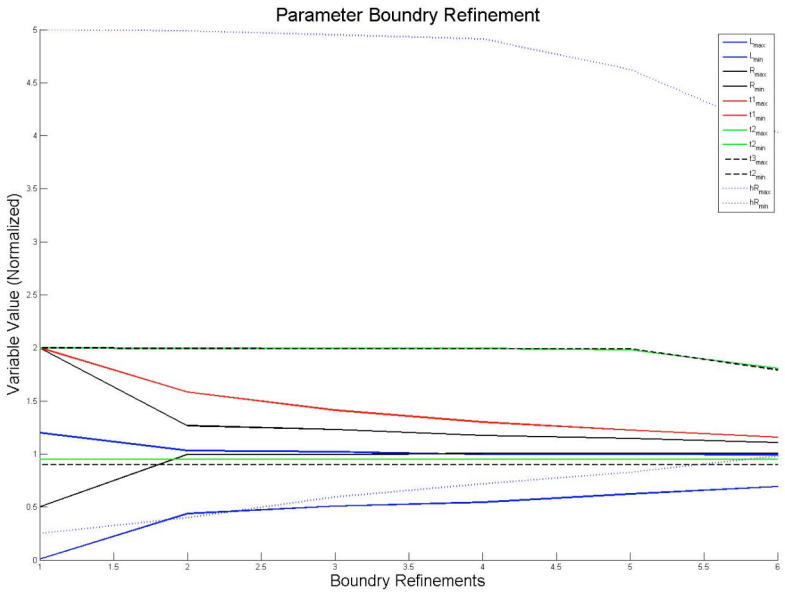


Figure 1: Parameter Boundary Refinement

Analysis

The advantage of this boundary refinement is that computation time is more efficient then without boundary refinement and there are higher quality designs generated. From Figure 1, it is clear that increasing the number of generated designs within the design space increases computation time at an exponential rate.

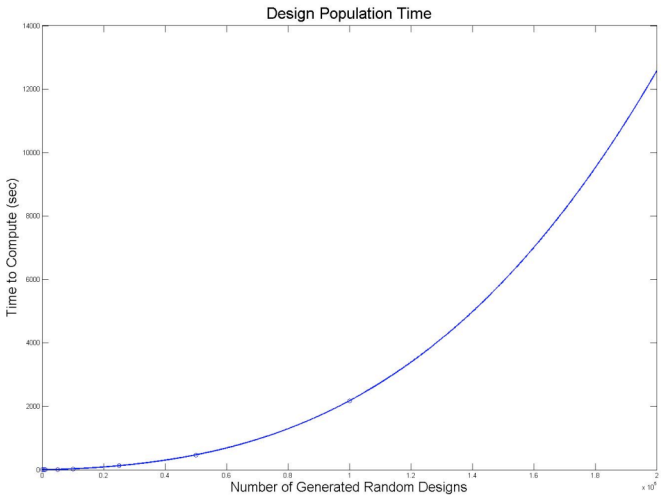


Figure 1: Design Population Time

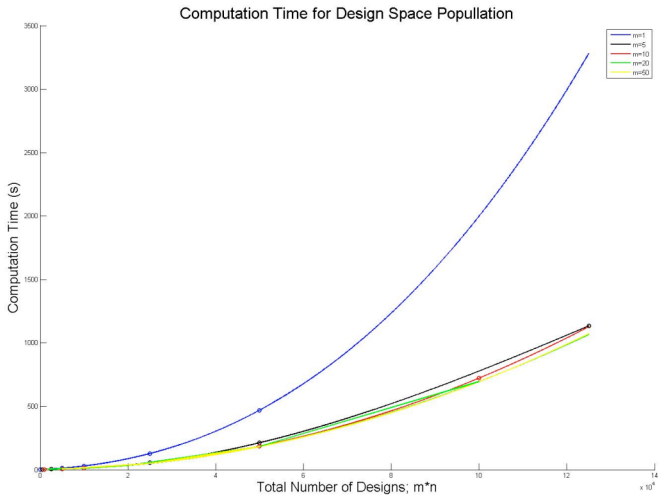


Figure 1: Computation Time for Design Space Population

So by generating smaller sets of designs and grouping them into the design space it can be more computationally efficient than just generating one large set of designs. This is shown in Figure 1 where computation time for any quantity of designs decreases as m increases though computation time decreases at a decreasing rate.

This increase in efficiency is because the computer rejects fewer designs due to "better" boundary ranges. For single iterations, m=1, the pass rate for designs is about 2.4%, when m=5, the pass rate increases to about 6.8%. This means the computer has to throw out fewer designs and therefore does not have to crunch numbers as long in order to get n number of usable designs.

Another advantage to boundary refinements is that a greater number of designs generated have a higher ROI value. For instance in Figure 1, 50,000 designs were generated without boundary refinement. It can be seen from the histogram that the average of the distribution is approximately -.55 with >1% of the distribution above zero ROI. In Figure 1, 50,000 designs were generated with 5 boundary refinements. Under this condition the average of the distribution is approximately -.2 with approximately 5% of the distribution above zero ROI.

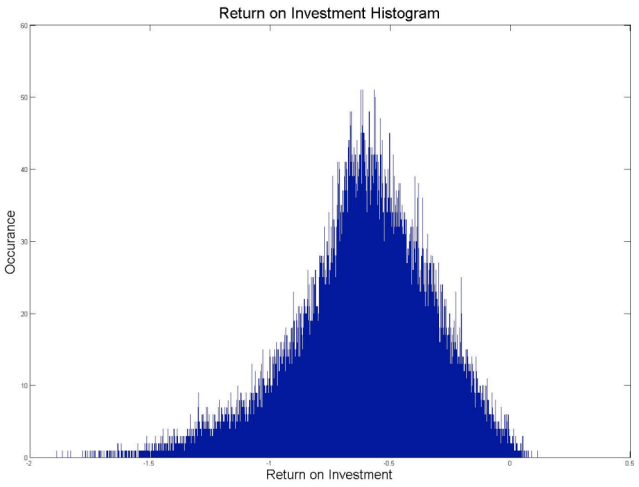


Figure 1: ROI Histogram w/o Boundary Refinement

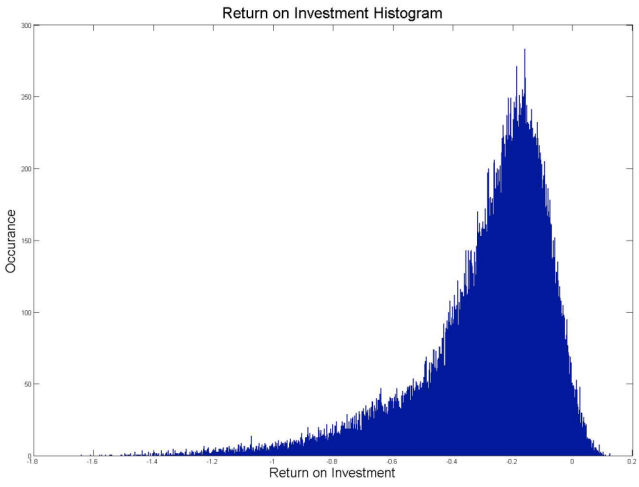


Figure 1: ROI Histogram with Boundary Refinement

ATSV

On the first attempt at using ATSV 150 points were generated randomly. Then the brush settings were adjusted to maximize ROI and using the preference section of the exploration engine about 300 more points were generated. It wasn't until this time that the volume, stress and vibration constraints were found. After changing the brush settings to only include the designs that meet the constraints, it was found that there were no feasible designs in this region. A target was then created at TankVolconst=0, SphereStressConst=0 and Cyl.str.const=0. This was run for about an hour and it generated a total of 18 feasible points. This session took about 2 hours.

For the second attempt the exploration engine was run longer to produce more points. First it generate 5,000 points blindly (4 of which met all constraints). After observing the dataset, the range of L_n was changed to .01-1.5 and the exploration engine generated 5,000 more points (9 of which met all constraints). The brush preferences settings were then changed to minimize the volume constraint (100% preference), minimize the value of $t1_n$ (50% preference), and minimize t3n (50% preference) based on trend observation. The preference function was looped for about 15 minutes generating 5,000 points for a total of 15,000 points. Only 2 feasible points were

generated during this run. The $t1_n$ and $t3_n$ brush preferences were then removed and 5,250 more points were generated. This yielded 28 more feasible designs for a total of 43 designs that met all the problem constraints. Based on this second attempt at using the program the highest ROI was 0.151.

Based on the results from the second trial it was decided to minimize the tank volume constraint and let the exploration engine run for a very long time so that it might yield a generous number of feasible designs. After the preference function based population was set, it was run overnight. It ran for a total of 20 hours 2.5 minutes and generated 302,556 points with only 571 feasible points. That is a pass rate of less than 0.2%. When the run was terminated it was impossible to view all the points (ATSV kept giving a memory error) even in the list view. So the log was saved and the visible points (feasible points) were also saved to a file. Rather than having to save the log it would have been useful to have a "save all data" button that would create a .txt file of all of the points. Notepad was then used to count the number of log entries to get the total amount of points generated and was able to open the visible points file with ATSV after restarting the program. The biggest trend observed with the feasible points was that as $t1_n$ goes down ROI goes up. Most other values seemed to be pretty random. Based on this attempt the highest ROI was 0.1924.

While running these trials the low yield rate of feasible designs (<0.2%) was discouraging. So an attempt was made to change the Java code of the exploration engine. The goal was to eliminate one of the major contributors to the failure of design points, the tank volume constraint. This was done by removing the h-R ratio from the independent variables and making it dependant on the others through the volume constraint. With this reprogrammed there should be no points generated that do not meet the tank volume constraint. After the code was reprogrammed an initial test was run. Since there were no limits applied to the h-R ratio some of the values for the ROI and other calculations were negative infinity or 0. Based on the trend shown by the previous trial the limits of the variables were adjusted. The ranges were changed to $0.5<Ln<3$, $0.5<Rn<1$, $0.25<t1n<1$, $0.25<t2n<2$ and $0.25<t3n<2$. This was then used to generate 406,000 points. When the brush settings were applied, only 19 feasible points were created. This was largely accredited to the brush settings for the h-R ratio. Since the h-R ratio needed to be within the specified range, most of the design points became infeasible. So, by eliminating one aggressive constraint, a new aggressive constraint was created.

The design point created by the third approach (ROI of 0.1924) was the best that ATSV produced. The values of this design point were then plugged into Excel. By minimizing each value one by one while meeting all the constraints a new design point with a higher ROI (ROI of .306579) was produced. The order in which the input variables were minimized was R_n, L_n, h/R, $t1_n$, $t2_n$ and lastly $t3_n$.

IV. Results

Excel

Excel Solver

The Generalized Reduced Gradient algorithm used in the Excel solver function is method of finding the optimum by "stepping" away from an initial point until the slope of the ROI surface becomes zero or negative. This method is robust in that it tends to give a result even with nonlinear problems; however there is no way of knowing if the resulting point is a local or global maximum. The optimum found using this method is dependent upon the initial point. As can be seen in Table 1 the Excel solver found two maxima: a local maximum, ROI = 0.23, and a possible global maximum, ROI = 0.32. The Excel solver is very fast, nearly instantaneous.

Table 1: Excel Solver results for ten trials

	Initial	**Optimum**	Initial	**Optimum**	Initial	**Optimum**	Initial	**Optimum**	Initial	**Optimum**
ROI	0.05	**0.23**	1.02	**0.32**	-8.98	**0.32**	1.33	**0.32**	0.99	**0.23**
Ln	1	**1.2**	2.08	**1.18**	2	**1.18**	0.25	**1.18**	2.08	**1.2**
Rn	1	**0.92**	0.7	**0.9**	2	**0.9**	0.25	**0.9**	0.72	**0.92**
t1n	1	**0.9**	0.25	**0.88**	2	**0.88**	0.25	**0.88**	0.25	**0.9**
t2n	1	**0.91**	0.25	**0.89**	2	**0.89**	0.25	**0.89**	0.25	**0.91**
t3n	1	**0.84**	0.25	**0.82**	2	**0.82**	0.25	**0.82**	0.25	**0.84**
h/Rn	1	**2.35**	3.5	**2.39**	2	**2.39**	0.25	**2.39**	3.6	**2.35**

	Initial	**Optimum**	Initial	**Optimum**	Initial	**Optimum**	Initial	**Optimum**	Initial	**Optimum**
ROI	1.11	**0.32**	-368.51	**0.23**	1.19	**0.32**	-66.41	**0.23**	1.19	**0.23**
Ln	2.19	**1.18**	10	**1.2**	0.01	**1.18**	10	**1.2**	0.01	**1.2**
Rn	0.55	**0.9**	10	**0.92**	0.01	**0.9**	10	**0.92**	0.01	**0.92**
t1n	0.35	**0.88**	10	**0.9**	0.01	**0.88**	0.01	**0.9**	10	**0.9**
t2n	0.38	**0.89**	10	**0.91**	0.01	**0.89**	0.01	**0.91**	10	**0.91**
t3n	0.37	**0.82**	10	**0.84**	0.01	**0.82**	0.01	**0.84**	10	**0.84**
h/Rn	2.22	**2.39**	10	**2.35**	0.01	**2.39**	1	**2.35**	0.01	**2.35**

CrystalBall

Since the Excel solver does not look at the entire tradespace, CrystalBall was used to randomly generate points in the hopes of either finding a global maximum or finding a design to use as an initial point in Excel. As can be seen in Table 2, all but one of the maxima found using CrystalBall have smaller ROI's than the best design found using the solver. This may be partly due to the way the solver allowed the constraints to be very, very small positive numbers ($1*10^{11}$) while all the constraints in CrystalBall were less than or equal to zero.

Table 2: Crystal Ball Results

	1	2	3	4	5	6	7	8	9	10
ROI	**0.235**	0.006	0.002	0.023	-0.007	0.031	0.162	0.049	0.049	0.14
Ln	**1.179**	0.905	1.149	1.146	0.881	0.947	1.073	0.839	0.839	1.268
Rn	**0.908**	1.008	0.923	0.932	1.009	0.992	0.944	1.045	1.045	0.896
t1n	**0.91**	1.182	1.165	0.953	1.062	0.995	0.947	1.064	1.064	0.933
t2n	**1.048**	1.002	1.264	1.435	1.941	1.492	1.14	1.191	1.191	1.255
t3n	**1.116**	0.997	0.955	1.995	1.025	1.777	1.292	1.409	1.409	0.877
h/Rn	**1.721**	3.272	2.175	1.014	3.728	2.477	2.655	1.933	1.933	1.247

Comparison

Table 3 shows the best design points created by all four methods along with the two design points created by making minor adjustments to the MatLab and ATSV results.

Table 3: Results Comparison

	Nominal	Solver	Crystal Ball	MatLab	MatLab (Excel)	ATSV	ATSV (Excel)
ROI	0.05	0.3219	0.2348	0.1954	0.1964	0.1924	0.3066
Ln	1	1.1808	1.1786	0.8664	0.8664	1.1	1.0995
Rn	1	0.8994	0.9077	1.0198	1.0198	0.9215	0.9172
t1n	1	0.8762	0.9098	0.9945	0.9934	0.9827	0.8936
t2n	1	0.8853	1.0478	1.0045	1.0039	1.0429	0.9029
t3n	1	0.8188	1.1163	0.9290	0.9284	0.9792	0.8343
h/Rn	1	2.3919	1.7209	2.4202	2.4202	3.2078	3.2078

V. Conclusion

Analysis

By using several different programs and methods to find optimal solutions to the tightly constrained problem a deep understanding for the relationships was created. Although Excel Solver produced the maximum ROI, other local maximums were found using the other programs.

Based on the experience of the group, it was decided that Excel was the easiest way to solve this specific problem. The problem was a tightly constrained single objective optimization problem that, due to its nature, made Excel the most powerful tool for solving it. If there were multi-objectives, ATSV would most definitely come out on top.

For problem understanding, Excel was scored at a 3 (out of 5) due to its simplified outputs. MatLab was scored at a 5 in this area since the code had to be programmed by the user and the visual capabilities of the program really helped. ATSV was scored at 4 because of its great ability to show, visually, the relationships between the variables and the objective.

For visualization capabilities of the software, Excel was scored at a 1 because its output was limited to a single design point. MatLab was scored at a 3 and ATSV was scored at a 5. ATSV really shines when it comes to visualizing the problem once there are enough points generated in the feasible design space.

The ATSV training was great for the time that was provided. If it was possible for a follow up session or workshop the additional experience and knowledge gained would have been valuable. That's why the training was rated at a 3.

The problem description was rated at a 4 only because there wasn't a complete optimization problem statement that included all the equations used. This made it difficult for MatLab programming and Java reprogramming.

ATSV would be a very useful product for examining multi-objective design problems. It also would be useful for comparing existing products on the market for purchase. The design by shopping method is interesting and is probably very useful in many fields.

Recommendations

ATSV does not have a "save all data" or a "save project" button. This would be very useful. If it was possible to simply save all of the design points in a .txt file to open it later, or with a different program without having to reset all of my brushes would be very useful. Also, if I was able to save the entire project that would be wicked awesome! If I were able to save my project (including brush settings, exploration engine settings, data sets and all visible windows with their settings) I would be able to pick up exactly where I left off at a later time without having to reset everything.

Another useful function for ATSV would be a way to set the limits on the color scale. Using the options to change the range of the axis is great, but being able to change the range on the color scale would be very beneficial. As it is, it doesn't matter what your brushes are set to, the color scale stays the same so all of your points might appear red which does not help.

I liked being able to select design points on the different displays and have their information come up. What I didn't like was the ability to select non-visible design points that were not included in the brush preferences. This made it difficult to select a feasible point that was surrounded by invisible non-feasible points. An option to only select visible points would be appreciated.

One of the greatest features that would make ATSV a fuller application would be the addition of an exploration engine creation tool, which does not require any manual code generation. Since most exploration engines are going to have the same layout (optimization problem layout) it should be relatively easy to create a "wizard" type tool or template. First, it would ask for the number of independent variables (inputs), their names, and their ranges. Then the number of dependant variables (outputs), their names and how they are calculated in relation to the inputs. The last step would be to include the problem constraints in the exploration engine. When the code for the engine is generated it would have an auto-check to verify if the design point meets all the constraints, if not, then the point would be trashed and not fully generated. The exploration engine would then display the number of trashed points after the run was complete. This would greatly conserve time and memory of the computing system. By having an exploration engine that is easily changed through the creation tool, you could relax your constraints on the fly, or change any other parameter when needed without having to search through lines of code.

VI. □ Appendix A

MatLab Code

```
% Design Optimization of External Fuel Tank for Space Shuttle
    % Objective is to maximize return on investment (ROI) through design
    % optimization of tank geometry.

clc; clear all; close all;
tic

%Operator Controls
n=10000;         %Number of designs (needs to be an integer)
m=20;            %Number of bound refinements (needs to be an integer)
x_max=10;         %Variable Bound Redefinition Width, min and max
x_min=1;

%Model Variables
k=6;                            %Material cost per unit mass ($/kg)
lamda=12;                       %Seam cost per unit length ($/m)
P=70;                           %Tank pressure (N/cm^2)
P_nom=30000;                    %Nominal payload (kg)
sigma_y=40000;                  %Yeild stress for tank material (N/cm^2)
rho=.0028;                      %Material density (kg/cm^3)
Pi=3.1416;
t=m+1;                          %Count Down Indicator

%Nominal Tank Geometry Varibles
L_nom=4150;                     %Cylinder length (cm)
R_nom=450;                      %Tank radius (cm)
t1_nom=0.7;                     %Cylinder thinkness (cm)
t2_nom=0.8;                     %Sphere thickness (cm)
t3_nom=0.75;                    %Cone thickness (cm)
hR_nom=1;                       %h/R cone height to radius ratio
j=0;                            %j is equal to # of rejected designs

%Variable Bounds
a_L=.01;                    %Minimum value for tank Legnth
b_L=1.2;                    %Maximum value for tank Legnth
a_R=.5;                     %Minimum value for tank Radius
b_R=2;                      %Maximum value for tank Radius
a_hR=.25;                   %Minimum value for Cone height radius ratio
b_hR=5;                     %Maximum value for Cone height radius ratio
a_t1=.95;                   %Minimum value for cylinder thickness
b_t1=2;                     %Maximum value for cylinder thickness
a_t2=.95;                   %Minimum value for sphere thickness
b_t2=2;                     %Maximum value for sphere thickness
a_t3=.9;                    %Minimum value for cone thickness
b_t3=2;                     %Maximum value for cone thickness
Bounds=[a_L,b_L,a_R,b_R,a_t1,b_t1,...
    a_t2,b_t2,a_t3,b_t3,a_hR,b_hR];

for q=1:m;
    if m>1;
        x=round(x_min+(m-q)/(m-1)*(x_max-x_min));          %variable boundry correction window
width
    end
    for i=1:n;

        %User changes normalized values
        %Tank Geometry Variables...Non dimenstionalized (input varibales)
        L_non(i,1)=a_L+rand(1)*(b_L-a_L);        %Cylinder length (unitless)
        R_non(i,1)=a_R+rand(1)*(b_R-a_R);        %Tank radius (unitless)

        %Actual Tank Geometry Variables
        L(i,1)=L_non(i,1)*L_nom;            %Cylinder length (cm)
        R(i,1)=R_non(i,1)*R_nom;            %Tank radius (cm)
```

```
        h(i,1)=(2826.4*100^3-2/3*Pi*R(i,1)^3-Pi*R(i,1)^2*L(i,1))/(1/3*Pi*R(i,1)^2); %Cone height
dependent on a volume of 3000 m^3
        hR(i,1)=h(i,1)/R(i,1);                  %h/R cone height to radius ratio

        hR_non(i,1)=hR(i,1);                    %h/R cone height to radius ratio (unitless)

        t1_non(i,1)=a_t1+rand(1)*(b_t1-a_t1);     %Cylinder thinkness (unitless)
        t2_non(i,1)=a_t2+rand(1)*(b_t2-a_t2);     %Sphere thickness (unitless)
        t3_non(i,1)=a_t3+rand(1)*(b_t3-a_t3);     %Cone thickness (unitless)
        t1(i,1)=t1_non(i,1)*t1_nom;               %Cylinder thinkness (cm)
        t2(i,1)=t2_non(i,1)*t2_nom;               %Sphere thickness (cm)
        t3(i,1)=t3_non(i,1)*t3_nom;               %Cone thickness (cm)
        I(i,1)=sqrt(R(i,1)^2+(hR(i,1)*R(i,1))^2); %Cone slant length

        %Stress Analysis
        sigma_hoop_cyl=P*R(i,1)/t1(i,1);
        sigma_long_cyl=P*R(i,1)/(2*t1(i,1));
        sigma_cyl=sqrt(sigma_hoop_cyl^2+sigma_long_cyl^2-sigma_hoop_cyl*sigma_long_cyl);
        sigma_sph=P*R(i,1)/t2(i,1);
        sigma_hoop_con=P*R(i,1)/t3(i,1);
        sigma_long_con=P*R(i,1)/(2*t3(i,1))*I(i,1)/h(i,1);
        sigma_con=sqrt(sigma_hoop_con^2+sigma_long_con^2-sigma_hoop_con*sigma_long_con);

        %Vibration Analysis
        Cyl_area=2*Pi*R(i,1)*L(i,1);
        Sph_area=2*Pi*R(i,1)^2;
        Con_area=Pi*R(i,1)*I(i,1);
        W=(Cyl_area*t1(i,1)+Sph_area*t2(i,1)+Con_area*t3(i,1))*rho;
        Xi=10000*sqrt(R(i,1)^3*t1(i,1)/(W*(L(i,1)+R(i,1)+h(i,1))^3));        %Vibration factor
        Xi_nom=1.33627;                %Nominal vibraiton factor

            while hR(i,1)<=a_hR || hR(i,1)>=b_hR || sigma_cyl>=sigma_y || sigma_sph>=sigma_y ||
sigma_con>=sigma_y || Xi<=Xi_nom;
                L_non(i,1)=a_L+rand(1)*(b_L-a_L);       %Cylinder length (unitless)
                R_non(i,1)=a_R+rand(1)*(b_R-a_R);       %Tank radius (unitless)
                L(i,1)=L_non(i,1)*L_nom;                %Cylinder length (cm)
                R(i,1)=R_non(i,1)*R_nom;                %Tank radius (cm)
                h(i,1)=(2826.4*100^3-2/3*Pi*R(i,1)^3-Pi*R(i,1)^2*L(i,1))/(1/3*Pi*R(i,1)^2); %Cone
height dependent on a volume of 3000 m^3
                hR(i,1)=h(i,1)/R(i,1);                  %h/R cone height to radius ratio
                hR_non(i,1)=hR(i,1);                    %h/R cone height to radius ratio
(unitless)
                t1_non(i,1)=a_t1+rand(1)*(b_t1-a_t1);     %Cylinder thinkness (unitless)
                t2_non(i,1)=a_t2+rand(1)*(b_t2-a_t2);     %Sphere thickness (unitless)
                t3_non(i,1)=a_t3+rand(1)*(b_t3-a_t3);     %Cone thickness (unitless)
                t1(i,1)=t1_non(i,1)*t1_nom;               %Cylinder thinkness (cm)
                t2(i,1)=t2_non(i,1)*t2_nom;               %Sphere thickness (cm)
                t3(i,1)=t3_non(i,1)*t3_nom;               %Cone thickness (cm)
                I(i,1)=sqrt(R(i,1)^2+(hR(i,1)*R(i,1))^2); %Cone slant length
                sigma_hoop_cyl=P*R(i,1)/t1(i,1);
                sigma_long_cyl=P*R(i,1)/(2*t1(i,1));
                sigma_cyl=sqrt(sigma_hoop_cyl^2+sigma_long_cyl^2-sigma_hoop_cyl*sigma_long_cyl);
                sigma_sph=P*R(i,1)/t2(i,1);
                sigma_hoop_con=P*R(i,1)/t3(i,1);
                sigma_long_con=P*R(i,1)/(2*t3(i,1))*I(i,1)/h(i,1);
                sigma_con=sqrt(sigma_hoop_con^2+sigma_long_con^2-sigma_hoop_con*sigma_long_con);
                Cyl_area=2*Pi*R(i,1)*L(i,1);
                Sph_area=2*Pi*R(i,1)^2;
                Con_area=Pi*R(i,1)*I(i,1);
                W=(Cyl_area*t1(i,1)+Sph_area*t2(i,1)+Con_area*t3(i,1))*rho;
                Xi=10000*sqrt(R(i,1)^3*t1(i,1)/(W*(L(i,1)+R(i,1)+h(i,1))^3));
                j=j+1;                              %Counter for # of rejected designs
            end

        % Surface and Volume Analysis
        Sph_area=2*Pi*R(i,1)^2;
        Sph_vol=2/3*Pi*R(i,1)^3;
        Cyl_area=2*Pi*R(i,1)*L(i,1);
        Cyl_vol=Pi*R(i,1)^2*L(i,1);
```

```
        Con_area=Pi*R(i,1)*I(i,1);
        Con_vol=1/3*Pi*R(i,1)^2*h(i,1);

        Tank_area=Sph_area+Cyl_area+Con_area;
        Tank_vol=Sph_vol+Cyl_vol+Con_vol;

        Tank_area_nom=13905910;         %Nominal tank surface area (cm^2)
        Tank_vol_nom=2926400400;        %Nominal tank volume (cm^3)

        %Weld Seam Lengths
        l_cyl=4*L(i,1);
        l_sph=2*Pi*R(i,1);
        l_con=4*I(i,1);
        l_cyl_sph=2*Pi*R(i,1);
        l_cyl_con=2*Pi*R(i,1);
        l=l_cyl + l_sph + l_cyl_sph...
            + l_cyl_con + l_con;  %Total length of weld on tank (cm)

        %Tank Weight
        W=(Cyl_area*t1(i,1)+Sph_area*t2(i,1)+Con_area*t3(i,1))*rho;
        W_nom=27737;                                        %Nominal tank weight (kg)

        %Material Costs
        f1=1.15-.33*(t1(i,1)-.1)+.165*(t1(i,1)-.1)^2;     %Weight-Cost function for material of
cylinder
        f2=1.15-.33*(t2(i,1)-.1)+.165*(t2(i,1)-.1)^2;     %Weight-Cost function for material of
sphere
        f3=1.15-.33*(t3(i,1)-.1)+.165*(t3(i,1)-.1)^2;     %Weight-Cost function for material of
cone
        MC_cyl=Cyl_area*t1(i,1)*rho*k*f1;                  %Material cost of cylinder
        MC_sph=Sph_area*t2(i,1)*rho*k*f2;                  %Material cost of sphere
        MC_con=Con_area*t3(i,1)*rho*k*f3;                  %Material cost of cone
        MC=MC_cyl+MC_sph+MC_con;                           %Total Material cost

        %Seam Cost
        t_avg_cyl_sph=(t1(i,1)+t2(i,1))/2;                 %Average of cylinder and sphere
thickness
        t_avg_cyl_con=(t1(i,1)+t3(i,1))/2;                 %Average of cylinder and cone thickness
        f4=1.2-.42*(t1(i,1)-.1)+.25*(t1(i,1)-.1)^2;        %Seam length cost function for cylinder
        f5=1.2-.42*(t2(i,1)-.1)+.25*(t2(i,1)-.1)^2;        %Seam length cost function for sphere
        f6=1.2-.42*(t3(i,1)-.1)+.25*(t3(i,1)-.1)^2;        %Seam length cost function for cone
        f7=1.2-.42*(t_avg_cyl_sph-.1)+.25*(t_avg_cyl_sph-.1)^2; %Seam length cost function for
cylinder and sphere
        f8=1.2-.42*(t_avg_cyl_con-.1)+.25*(t_avg_cyl_con-.1)^2; %Seam length cost function for
cylinder and cone
        SC_cyl=lamda*l_cyl*f4;
        SC_sph=lamda*l_sph*f5;
        SC_con=lamda*l_con*f6;
        SC_cyl_sph=lamda*l_cyl_sph*f7;
        SC_cyl_con=lamda*l_cyl_con*f8;
        SC=SC_cyl+SC_sph+SC_con+SC_cyl_sph+SC_cyl_con;  %Total seam cost

        %Total Cost
        C=MC+SC;
        C_nom=511424.2;

        %Aerodynamic Analysis
        A=Pi*R(i,1)^2;                 %Tank cross sectional area
        P1=14300;                      %Given parameter
        P2=5000;                       %Given parameter
        Ao=636174;                     %Given parameter
        D=.25+1.4*exp(1-1.6*(1+(hR(i,1)-1)/1)); %Cone drag fucntion
        Delta_P=-P1*(A-Ao)/Ao-P2*(D-1.018336)/1.018336-P2*(Tank_area-
Tank_area_nom)/Tank_area_nom; %Change in payload capacity

        %Return on Investment
        C_launch=599488576+C;                       %True launch cost: fixed cost + tank cost
        P_launch=P_nom-(W-W_nom)+Delta_P;           %Launch payload
        C_customer=21000*P_launch;                  %Customer cost to launch payload
```

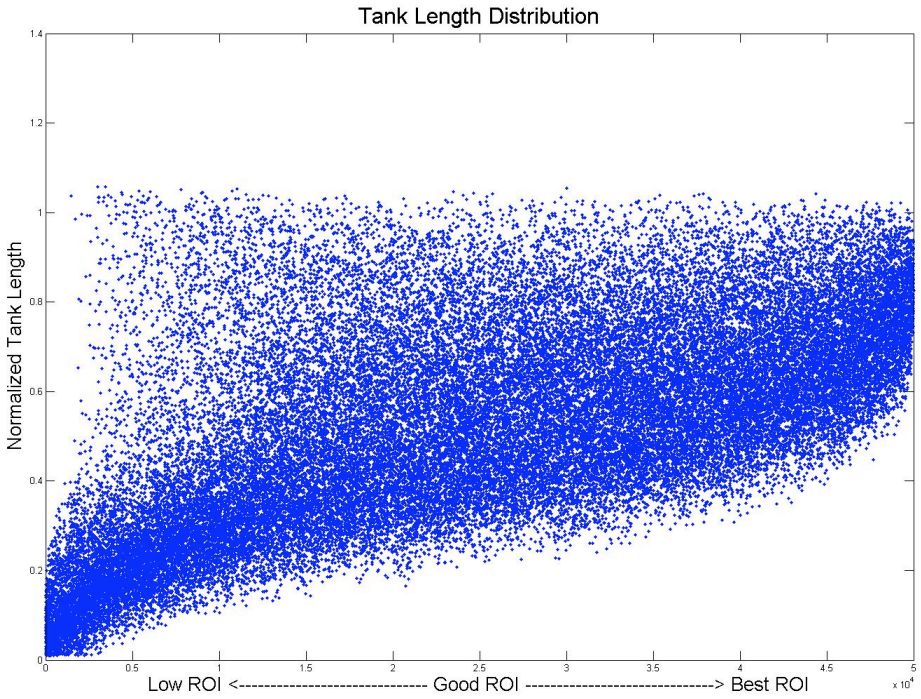
```
        roi(i,1)=(C_customer-C_launch)/C_launch;    %RETURN ON INVESTMENT!!!

        ROI(i+(q-1)*n,1)=roi(i);
        ROI(i+(q-1)*n,2)=L_non(i);
        ROI(i+(q-1)*n,3)=R_non(i);
        ROI(i+(q-1)*n,4)=t1_non(i);
        ROI(i+(q-1)*n,5)=t2_non(i);
        ROI(i+(q-1)*n,6)=t3_non(i);
        ROI(i+(q-1)*n,7)=hR(i);
    end

    %Sort ROI for best design
    ROI_sorted=sortrows(ROI,1);
    if m>1;
    %Redifine variable bounds
    a_L=min(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),2));
%Minimum value for tank Legnth
    b_L=max(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),2));
%Maximum value for tank Legnth
    a_R=min(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),3));
%Minimum value for tank Radius
    b_R=max(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),3));
%Maximum value for tank Radius
    %a_t1=min(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),4));
%Minimum value for cylinder thickness
    b_t1=max(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),4));
%Maximum value for cylinder thickness
    %a_t2=min(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),5));
%Minimum value for sphere thickness
    b_t2=max(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),5));
%Maximum value for sphere thickness
    %a_t3=min(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),6));
%Minimum value for cone thickness
    b_t3=max(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),6));
%Maximum value for cone thickness
    a_hR=min(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),7));
%Minimum value for Cone height radius ratio
    b_hR=max(ROI_sorted((1-x/100)*length(ROI_sorted):length(ROI_sorted),7));
%Maximum value for Cone height radius ratio
    Bounds(q+1,:)=[a_L,b_L,a_R,b_R,a_t1,b_t1,...
        a_t2,b_t2,a_t3,b_t3,a_hR,b_hR];
    end
t=t-1
end

%Best Design
[ROI_best,s]=max(ROI(:,1));

L_best=ROI(s,2);
R_best=ROI(s,3);
t1_best=ROI(s,4);
t2_best=ROI(s,5);
t3_best=ROI(s,6);
hR_best=ROI(s,7);
ROI_best

Best=[ROI_best;L_best;R_best;t1_best;t2_best;t3_best;hR_best];

%ROI histogram; number of bins is 10% of the number of designs (n)
hist(ROI(:,1),.1*n,'FontSize',18); set(gcf,'Color','w'); xlabel('Return on
Investment','FontSize',20); ylabel('Occurance','FontSize',20); title('Return on Investment
Histogram','FontSize',24)

%variable distribustion; 0 -> worst design, n -> best design

figure; plot(1:n*m,ROI_sorted(:,2),'.'); set(gcf,'Color','w'); title('Tank Length
Distribution','FontSize',24); xlabel('Low ROI <------------------------------ Good ROI ----------
--------------------> Best ROI','FontSize',20); ylabel('Normalized Tank Length','FontSize',20);
```

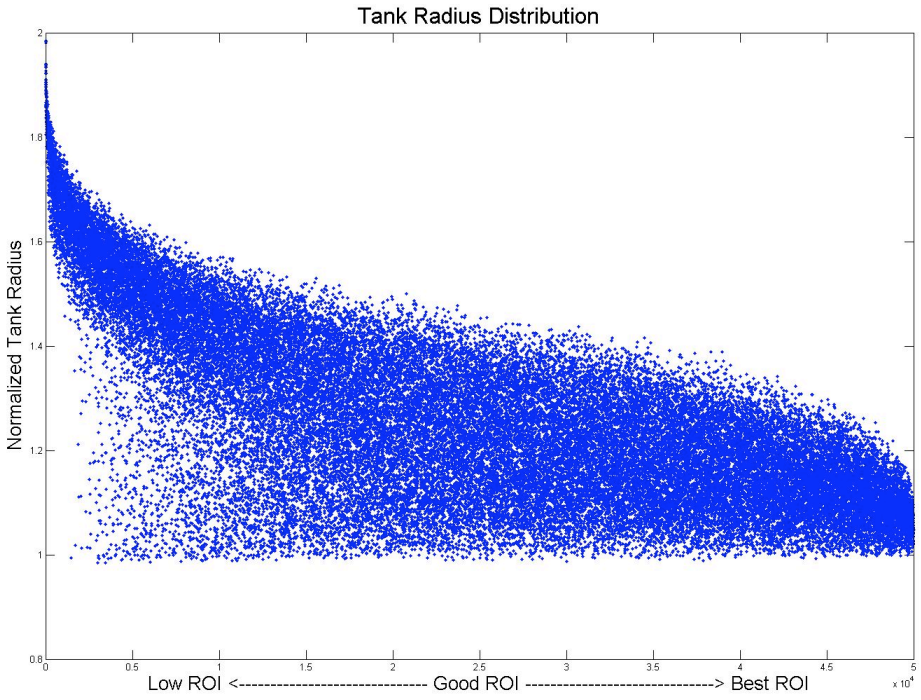
```
figure; plot(1:n*m,ROI_sorted(:,3),'.'); set(gcf,'Color','w'); title('Tank Radius
Distribution','FontSize',24); xlabel('Low ROI <------------------------------ Good ROI ----------
--------------------> Best ROI','FontSize',20); ylabel('Normalized Tank Radius','FontSize',20);
figure; plot(1:n*m,ROI_sorted(:,4),'.'); set(gcf,'Color','w'); title('Cylinder Thickness
Distribution','FontSize',24); xlabel('Low ROI <------------------------------ Good ROI ----------
--------------------> Best ROI','FontSize',20); ylabel('Normalized Cylinder
Thickness','FontSize',20);
figure; plot(1:n*m,ROI_sorted(:,5),'.'); set(gcf,'Color','w'); title('Sphere Thickness
Distribution','FontSize',24); xlabel('Low ROI <------------------------------ Good ROI ----------
--------------------> Best ROI','FontSize',20); ylabel('Normalized Sphere
Thickness','FontSize',20);
figure; plot(1:n*m,ROI_sorted(:,6),'.'); set(gcf,'Color','w'); title('Cone Thickness
Distribution','FontSize',24); xlabel('Low ROI <------------------------------ Good ROI ----------
--------------------> Best ROI','FontSize',20); ylabel('Normalized Cone
Thickness','FontSize',20);
figure; plot(1:n*m,ROI_sorted(:,7),'.'); set(gcf,'Color','w'); title('Cone Height Radius Ratio
Distribution','FontSize',24); xlabel('Low ROI <------------------------------ Good ROI ----------
--------------------> Best ROI','FontSize',20); ylabel('Cone Height Radius Ratio','FontSize',20);

%Change in Variable Boundary
if m>1;
figure;
hold; plot(1:m+1,Bounds(:,1),'LineWidth',2);    plot(1:m+1,Bounds(:,2),'LineWidth',2); %Length
plot(1:m+1,Bounds(:,3),'k','LineWidth',2);      plot(1:m+1,Bounds(:,4),'k','LineWidth',2); %
plot(1:m+1,Bounds(:,5),'r','LineWidth',2);      plot(1:m+1,Bounds(:,6),'r','LineWidth',2); %
plot(1:m+1,Bounds(:,7),'g','LineWidth',2);      plot(1:m+1,Bounds(:,8),'g','LineWidth',2); %
plot(1:m+1,Bounds(:,9),'-- k','LineWidth',2);   plot(1:m+1,Bounds(:,10),'-- k','LineWidth',2); %
plot(1:m+1,Bounds(:,11),':','LineWidth',2);     plot(1:m+1,Bounds(:,12),':','LineWidth',2); %
set(gcf,'Color','w');
title('Parameter Boundary Refinement','FontSize',24); xlabel('Boundary
Refinements','FontSize',20); ylabel('Variable Value (Normalized)','FontSize',20)
legend('L_m_a_x','L_m_i_n','R_m_a_x','R_m_i_n','t1_m_a_x','t1_m_i_n','t2_m_a_x','t2_m_i_n','t3_m_
a_x','t2_m_i_n','hR_m_a_x','hR_m_i_n');
end

%Percent of passed designs wrt rejected designs
J=n*m/(n*m+j)*100

toc

%Computation Time (Collected Data through multiple iteration of this program)
T=interp1([100 500 1000 5000 10000 25000 50000 100000],[2.13 2.78 3.66 12.51 29.68 127.8 469.7
2175],0:100:200000,'spline');
figure; plot([100 500 1000 5000 10000 25000 50000 100000],[2.13 2.78 3.66 12.51 29.68 127.8 469.7
2175],'o');
hold; plot(0:100:200000,T,'LineWidth',2); set(gcf,'Color','w');
title('Design Population Time','FontSize',24); xlabel('Number of Generated Random
Designs','FontSize',20); ylabel('Time to Compute (sec)','FontSize',20);

figure; hold on;

        T=interp1([100 500 1000 5000 10000 25000 50000],[2.13 2.78 3.66 12.5 29.68 127.8
469.7],0:100:125000,'spline');
        plot(0:100:125000,T,'LineWidth',2);
        T=interp1([100*5 500*5 1000*5 5000*5 10000*5 25000*5],[3.16 4.2 6.2 55.1 212.9
1134.9],0:100:125000,'spline');
        plot(0:100:125000,T,'k','LineWidth',2);
        T=interp1([100*10 500*10 1000*10 5000*10 10000*10],[3.4 6 12 185.5
722.9],0:100:125000,'spline');
        plot(0:100:125000,T,'r','LineWidth',2);
        T=interp1([100*20 500*20 1000*20 2500*20 5000*20],[4.05 11.78 33.5 182.5
695],0:100:125000,'spline');
        plot(0:100:125000,T,'g','LineWidth',2);
        T=interp1([100*50 500*50 1000*50 5000*50],[5.84 50.6 181.9 4183],0:100:125000,'spline');
        plot(0:100:125000,T,'y','LineWidth',2);

        plot([100 500 1000 5000 10000 25000 50000],[2.13 2.78 3.66 12.5 29.68 127.8
469.7],'o','LineWidth',2);
```

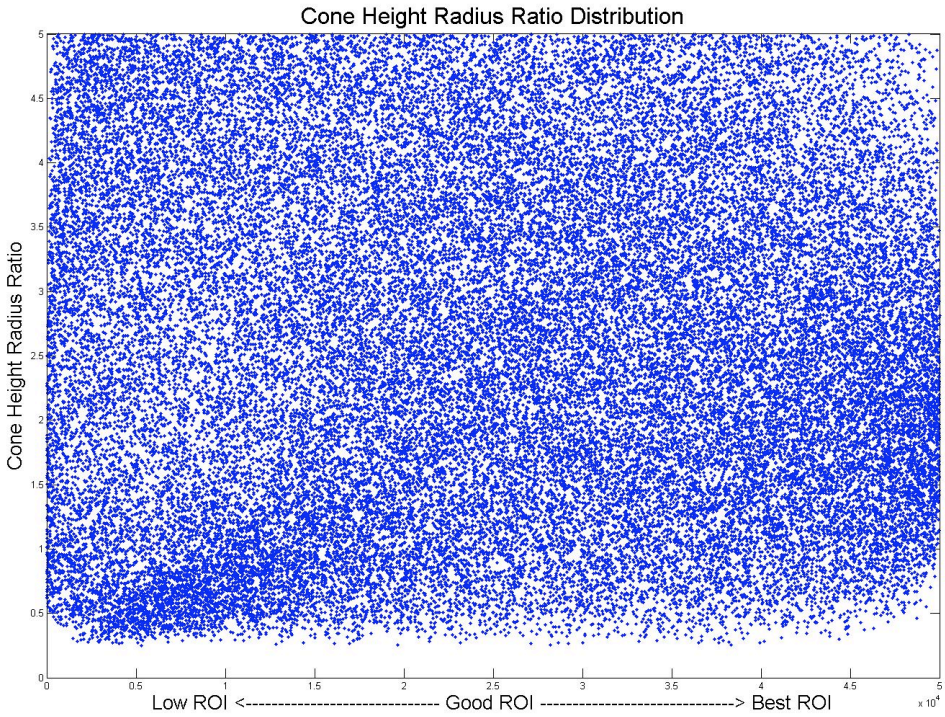
```
        plot([100*5 500*5 1000*5 5000*5 10000*5 25000*5],[3.16 4.2 6.2 55.1 212.9 1134.9],'k
o','LineWidth',2);
        plot([100*10 500*10 1000*10 5000*10 10000*10],[3.4 6 12 185.5 722.9],'r
o','LineWidth',2);
        plot([100*20 500*20 1000*20 2500*20 5000*20],[4.05 11.78 33.5 182.5
695],'g','LineWidth',2);
        plot([100*50 500*50 1000*50],[5.84 50.6 181.9],'y','LineWidth',2); set(gcf,'Color','w');

        xlabel('Total Number of Designs; m*n','FontSize',20); ylabel('Computation Time
(s)','FontSize',20); title('Computation Time for Design Space Popullation','FontSize',24)
        legend('m=1','m=5','m=10','m=20','m=50');
```

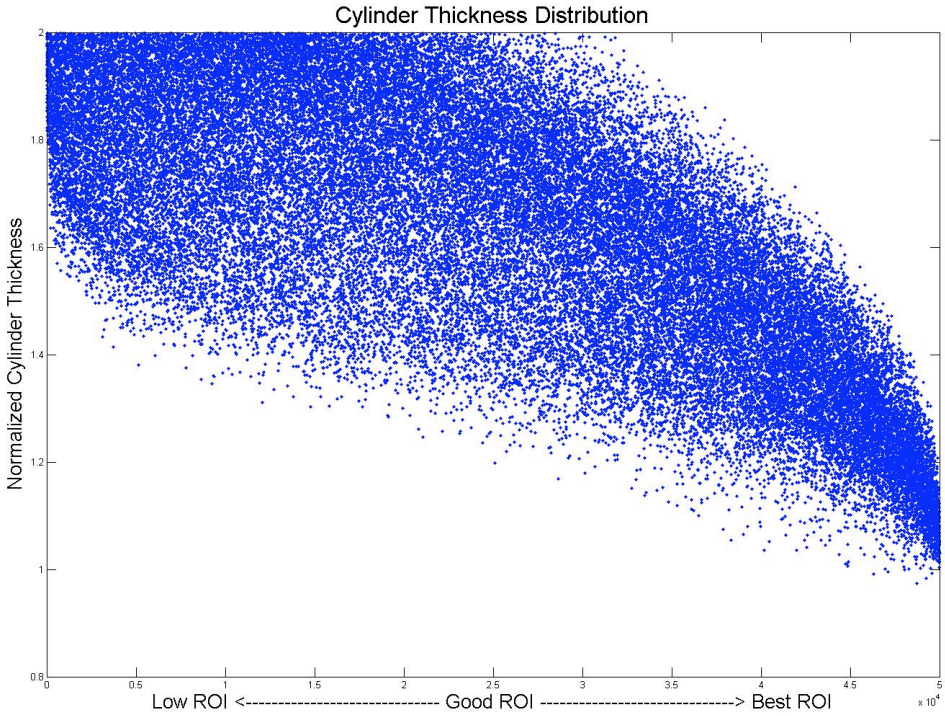
VII. Appendix B

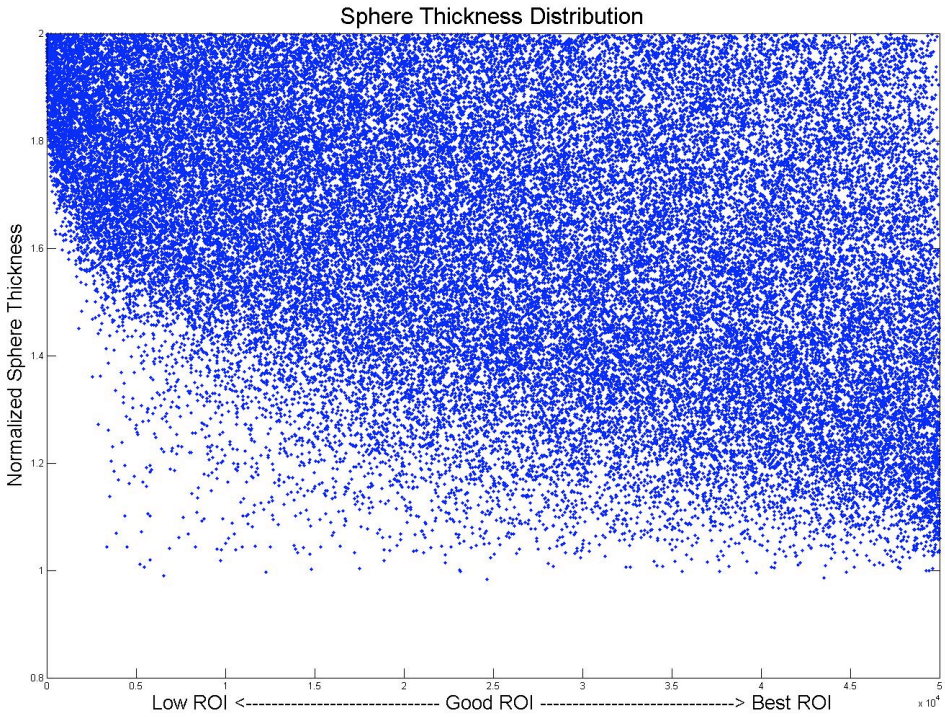
Variable parameters plotted WRT Return on Investment for a design population of 50000 with no boundary refinements.

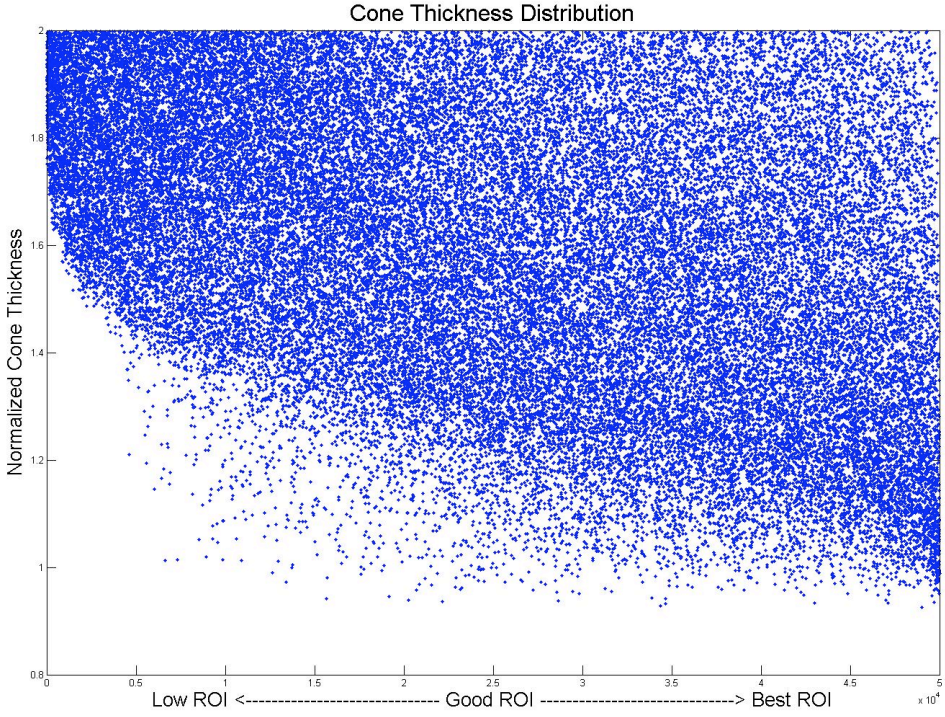





Cone Height Radius Ratio Distribution
Cone Height Radius Ratio
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4

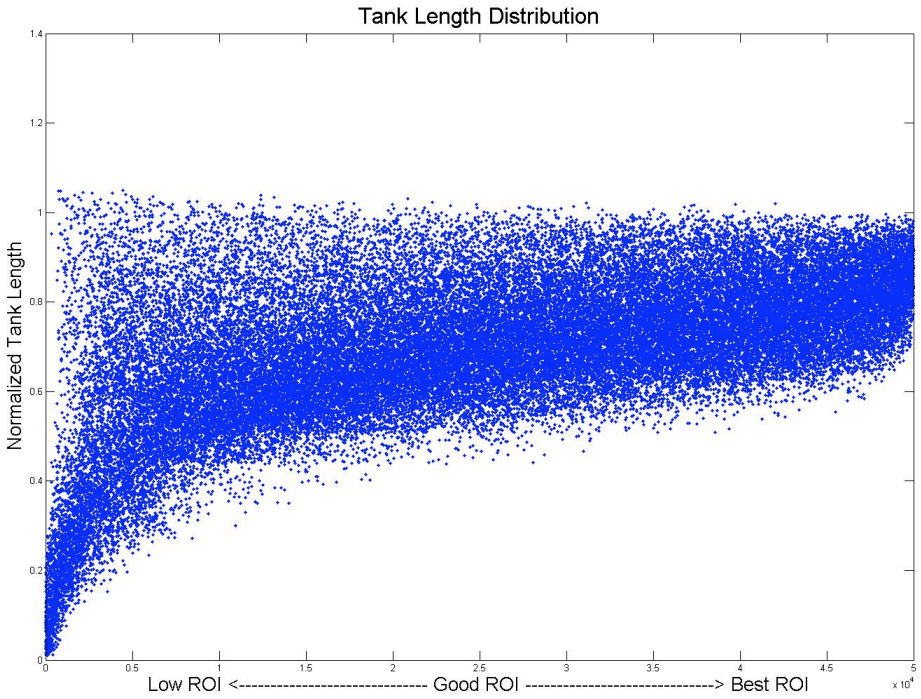

Cylinder Thickness Distribution
Normalized Cylinder Thickness
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4

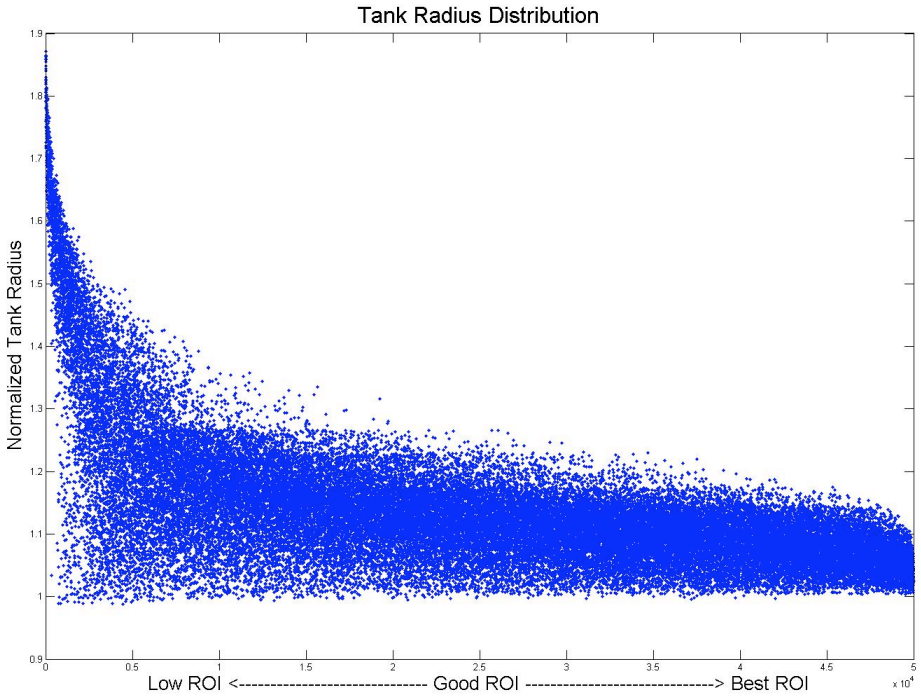

Sphere Thickness Distribution
Normalized Sphere Thickness
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4

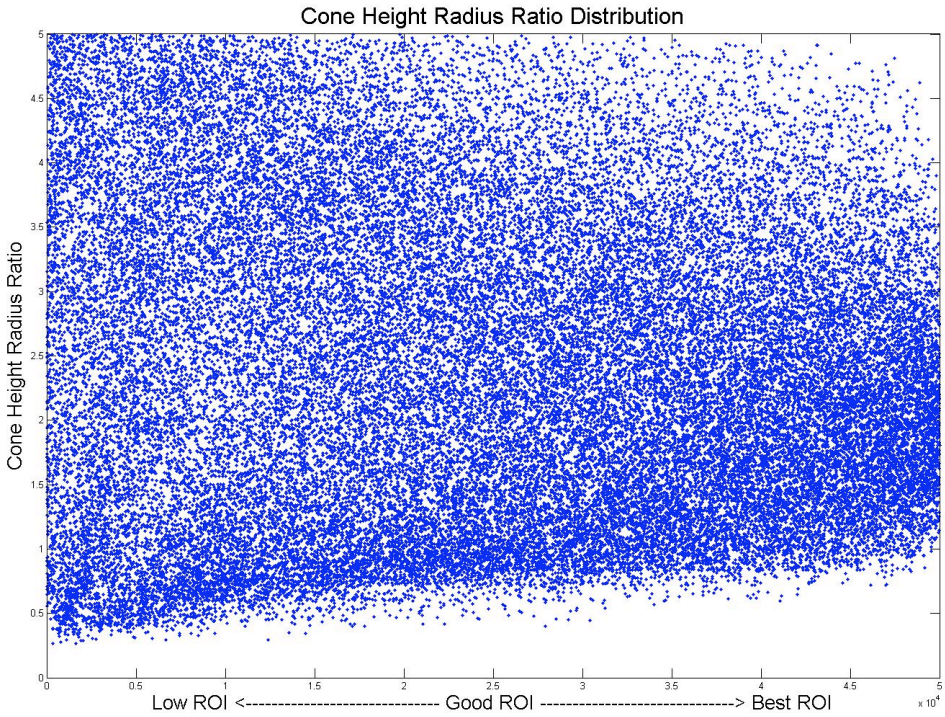

Cone Thickness Distribution
Normalized Cone Thickness
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4

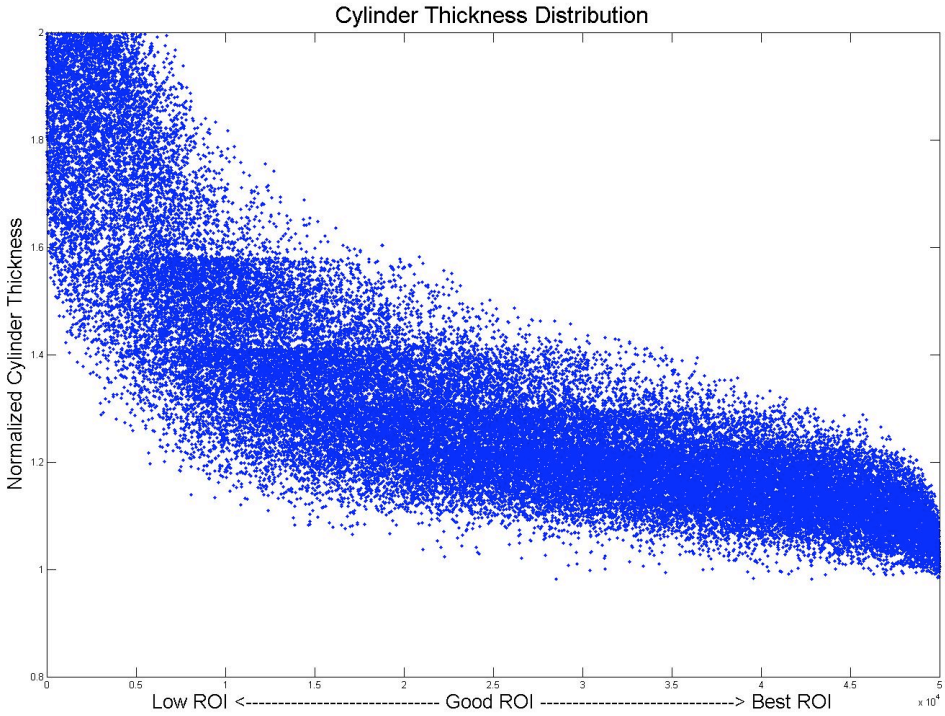
VIII. Appendix C

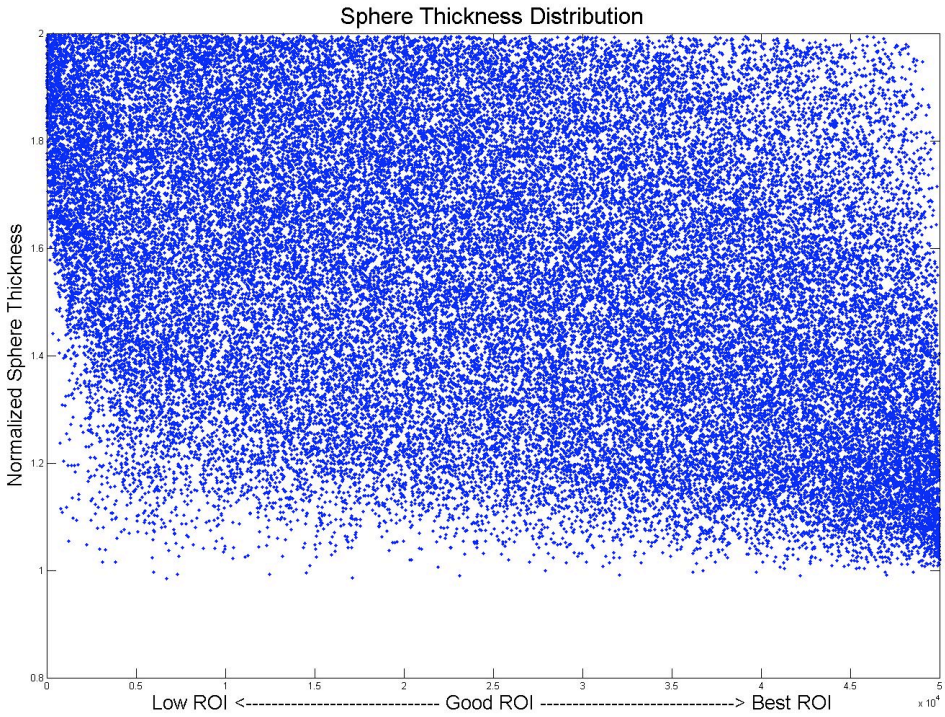
Variable parameters plotted WRT Return on Investment for a design population of 50000 with 5 boundary refinements.

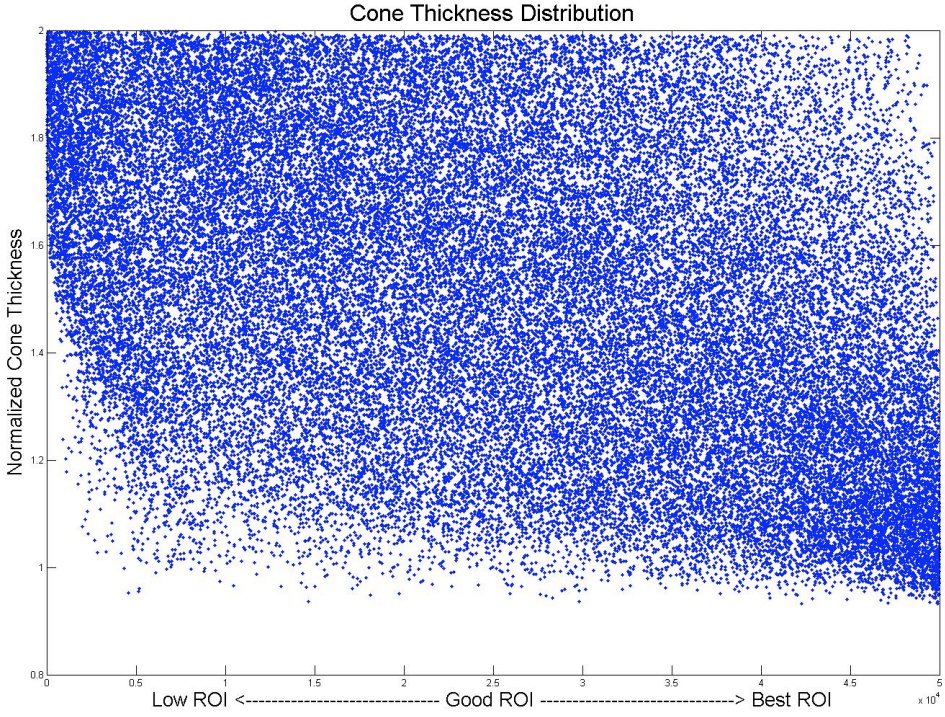





Cone Height Radius Ratio Distribution
Cone Height Radius Ratio
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4


Cylinder Thickness Distribution
Normalized Cylinder Thickness
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4


Sphere Thickness Distribution
Normalized Sphere Thickness
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4


Cone Thickness Distribution
Normalized Cone Thickness
Low ROI <---------- Good ROI ----------> Best ROI
x 10^4