Rocket Fin Shape Optimization

Alan Yu, Alexander Hu, Daniel Zeng
alanyu@mit.edu | alexhu@mit.edu | dzeng@mit.edu

1 Background

For their upcoming launch, the aerodynamics branch of MIT Rocket Team (MRT) wants to find parameters optimizing fin can geometry. Using trapezoidal fins, they hope to find the configuration that is both stable and leads to a high peak altitude on launch day. In this project, we tackle this constrained optimization problem by modeling via regression and neural networks. We also explore Bayesian Optimization as an alternative in Appendix A.1. Our work suggests improved rocket configurations and outlines a systematic optimization procedure that MRT can follow under different settings (e.g. different rocket engine, fin shape, etc.).

2 Problem Formulation & Related Work

In this problem, we are working with a two-stage rocket. The two components, sustainer and booster, are fired concurrently at launch and separate after some predetermined time (6 seconds) in flight (see 3a). There are four sets of fins on each component. For each component, we can control parameters corresponding to the fins (chord, span, sweep distance, and tipchord; see 3b), which we denote as $s = (s_0, \ldots, s_3)$ and $b = (b_0, \ldots, b_3)$ for the sustainer and booster, respectively. Each set of these parameters defines a fin. Hence, we are able to adjust 8 parameters in total.

We use a fixed engine file which specifies the thrust over time. Then, each rocket configuration is given by a `.CDX1` file, in which we can adjust each of the parameters to create a rocket with some desired fin configuration. We are also able to modify the weight freely, but we use density estimates of the material 6061 aluminum alloy to compute the weight $w(s, b)$.

We use RASAERO II, a software that accepts a rocket configuration and engine and simulates the altitude over time. For a given rocket r_i, we collect $A(t; r_i(s, b))$ and record $a(s, b) = \max_t A(t; r_i(s, b))$ as the peak altitude for r_i. In many cases, rockets will fail (4a) to complete a full flight (ascent to apogee followed by descent to the ground). This is caused by a low stability margin at some point in the flight, causing the rocket to terminate midair and very little altitude information. We also impose some reasonable physical constraints on each parameter, forcing them to take on values between 2 and 10 inches. Letting stability be denoted as the indicator $f(S(t; r_i(s, b)))$ and the physical constraints be described as $M = \{(s, b) : s_i, b_i \in [2, 10] \; \forall i \in [0, 3]\}$. we seek to solve the constrained optimization (COP)

$$\begin{aligned} &\max_{s,b} a(s, b) \\ \text{s.t.} &f(S(t; r(s, b))) = 1 \\ &(s, b) \in M. \end{aligned}$$

There is no closed form solution for stability and altitude, so we turn to regression to model the relationship between altitude, stability, and (s, b).

From our research, most related work has focused on determining the best shape of fins (ellipsoidal) rather than computing parameters for a fixed shape. We also came across a work

exploring the effect of different material and parameters on altitude and stability, but the study relies on sampling and data analysis rather than applying methods to predict the effect of parameters on altitude and stability (Bunkley et al., 2022).

3 Data Collection

In our setting, data is very expensive to collect. Because there is no API or batch-mode available for RASAERO II, we must collect data by manually uploading the rocket configuration, running the simulation, and downloading the resulting CSV file. In order to represent the region M well, we uniformly and independently sampled each parameter of (s, b) for a total of 400 samples and saved time, altitude, and stability observations at 1-second increments. Since each rocket gave a sequence of observations, we decided to represent each sequence with a singular value. For time and altitude, this was trivially the maximum observation time and altitude, respectively. Because the simulation would terminate immediately when the stability was not as desired, we noticed that the observation time (T) was a perfect predictor of how stable a rocket was. In particular, the data could be perfectly partitioned into sets with $T < 30$ and $T > 90$, with roughly half in each set. Within the dataset, the maximum altitude attained is 87,504.62 feet. We take this as the baseline solution for our problem.

4 Method and Results

In our approach, we seek to approximate the functions a and f by $\hat{a}$ and $\hat{f}$ and then try to solve the COP using these approximations. To make the model more precise, we define the domain of $\hat{a}$ as only consisting of stable rockets (i.e. $f(r) = 1$).

4.1 Stability

To model stability, we try various approaches to classify whether $T > 90$: logistic regression on (s, b) (Linear), logistic regression on linear and quadratic terms of (s, b) (Quadratic) with L^1/L^2 penalty (Lasso and Ridge), and neural network on linear and quadratic terms of (s, b) (NN). We found that including higher order terms does not improve the model. We evaluate the models using 4-fold CV (see Table 1) and CV within each fold for hyperparameters. We pick NN due to its marginally-higher score. We then perform CV again to select the final NN hyperparameters. For information about the NN architecture, see A.2.

4.2 Altitude

We use the same linear models but on continuous space. However, we also exclude the neural network approach. To justify this, recall that we only wish to regress on stable configurations, which comprise half of our dataset (≈ 200). After including quadratic terms, there are already 45 input parameters for the model, so we concluded that the amount of data available was insufficient for applying the neural network. This contrasts with the stability problem, where we had twice the amount of data available. After running 4-Fold CV, we get the performance in Table 1 and select the Ridge regressor. Tuning gives $\alpha_{L2} = 0.6$.

Fold # / Model	Linear	Quadratic	NN	Quadratic + Lasso	Quadratic + Ridge
1	0.89	0.88	0.86	0.89	0.88
2	0.94	0.90	0.89	0.94	0.93
3	0.94	0.94	0.97	0.95	0.94
4	0.83	0.88	0.89	0.80	0.85

Fold # / Model	Linear	Quadratic	NN	Quadratic + Lasso	Quadratic + Ridge
1	6026.60	1815.42	-	1815.02	2056.53
2	4918.04	2198.37	-	1648.13	1577.10
3	4558.18	3868.14	-	2055.45	1899.43
4	3984.86	1798.55	-	1794.28	1674.27

Task / Model	Linear	Quadratic	NN	Quadratic + Lasso	Quadratic + Ridge
Stability (Accuracy)	0.9	0.9	0.9025	0.895	0.9
Altitude (RMSE)	4871.92	2420.112	-	1828.21	1801.83

Table 1: **4-Fold Cross Validation**. For stability (first), the linear, quadratic, and regularized quadratic models refer to logistic regression with linear/quadratic terms. For altitude (second), these refer to linear regression with respective terms. Third table shows average over folds.

4.3 Solving COP

Taking into account the model validation results, we proceed taking $\hat{f}$ to be NN model and $\hat{a}$ to be the ridge-regularized quadratic model. We then solve the problem

$$\begin{aligned} &\max_{s,b} \hat{a}(s,b) \\ &\text{s.t.} \hat{f}(s,b) \geq 0.6 \\ &(s,b) \in M. \end{aligned}$$

using the COBYLA (Constrained Optimization by Linear Approximation) solver from SciPy on randomly selected stable points from the dataset, where the hyperparameter 0.6 is chosen as a confidence threshold for $\hat{f}$. Upon convergence, we immediately notice that most parameters are pushed to the boundary, and running the simulation on this point fails due to lack of stability. Upon reducing the number of iterations from 100 to 10, we have the same stability issue, even though the points are not pushed to the boundary. We observe that a possible cause of this issue is that COBYLA adjusts the parameters too much and $\hat{f}$ fails to generalize in those regions. To combat these issues, we establish a more local optimization algorithm.

Algorithm 1 Local Optimization

$X \leftarrow X_0$
$i \leftarrow 0$
while i < num_iterations **do**
  $Y \leftarrow X$
  $g \leftarrow \nabla_X \hat{a}(X)$
  $Y \leftarrow Y + g \odot \epsilon$
  if $Y \in M$ and $\hat{f}(Y) > \gamma$ and $\hat{a}(Y) \geq \hat{a}(X)$ **then**
    $X \leftarrow Y$
  else ▷ No more feasible directions
    break
  end if
end while
end

The idea of Algorithm 1 is to make small gradient updates in a neighborhood of X_0 while always improving $\hat{a}$ and remaining stable. By applying the algorithm to randomly sampled stable configurations, we get the best result with $\hat{a} = 92,480.15$ and $a = 93,142$ (see 1). Unlike the result from COBYLA, this is indeed stable in the simulation. Interestingly, we notice that a majority of the configurations the algorithm finds are stable in simulation, allowing us to extract insights for what optimal fins look like. To establish some intuition for the success of this algorithm, we have the following ascent lemma.

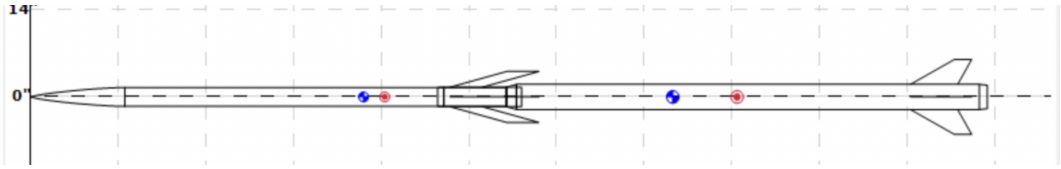
Lemma 1 *Suppose $f > 0$ in the ball $B_\delta(X_t)$ with $\nabla \hat{a}(X_t) \neq 0$. Then, $\exists \epsilon > 0$ sufficiently small such that $\hat{a}(X_t + \epsilon \nabla \hat{a}(X_t)) > \hat{a}(X_t)$ with $f(X_t + \epsilon \nabla \hat{a}(X_t)) = 1$.*

Proof. We mimic the steps justifying gradient descent from Bertsekas (1999). Since $f \in \{0,1\}$, we have $f(X) = 1$ for $X \in B_\delta(X_t)$. Taking the first-order Taylor expansion of $\hat{a}$ at X_t, we get $\hat{a}(X) = \hat{a}(X_t) + \nabla a(X_t)^T (X - X_t) + o(||X_t - X||)$, so

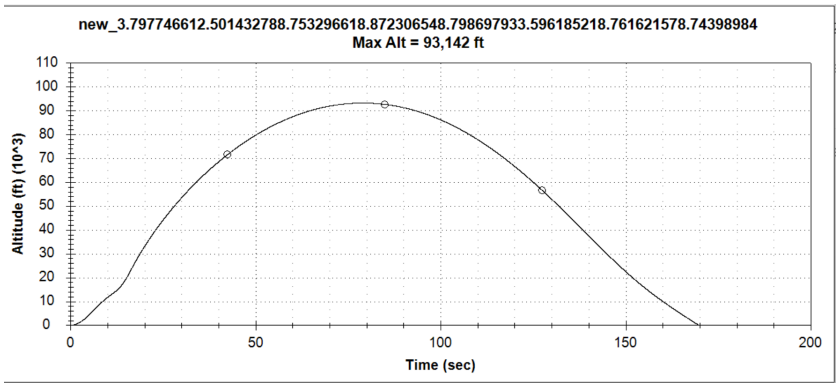
$$\begin{aligned} \hat{a}(X_t + \epsilon \nabla \hat{a}(X_t)) &= \hat{a}(X_t) + \nabla \hat{a}(X_t)^T (\epsilon \nabla \hat{a}(X_t)) + o(\epsilon ||\nabla \hat{a}(X_t)||) \\ &= \hat{a}(X_t) + \epsilon ||\nabla \hat{a}(X_t)||^2 + o(\epsilon), \end{aligned}$$

where $\limsup_{\epsilon \to 0} o(\epsilon) = 0$. *Hence, we can pick sufficiently small* $\epsilon_t \leq \frac{\delta}{||\nabla \hat{a}(X_t)||}$ *so that we have* $\epsilon||\nabla \hat{a}(X_t)||^2 + o(\epsilon) > 0 \implies \hat{a}(X_t + \epsilon_t \nabla \hat{a}(X_t)) > \hat{a}(X_t)$, *with* $d(X_t + \epsilon_t \nabla \hat{a}(X_t), X_t) \leq \delta \implies f(X_t + \epsilon_t \nabla \hat{a}(X_t)) = 1$.

The lemma suggests that if $\gamma \leq \hat{f} \leq f$ in a small neighborhood of X_t, using a sufficiently small ϵ in the update rule gives an improvement on $\hat{a}$.



(a) Fin Results



(b) Optimal Altitude Trajectory

Figure 1: Results from applying Algorithm I. (a) shows resulting parameters $(s, b) = ((3.80, 2.50, 8.75, 8.87), (8.80, 3.60, 8.76, 8.74))$ with $\hat{a} = 92,480.15$, and $\hat{f} = 0.99$. Computed using initial point $(s_0, b_0) = ((3.84, 3.12, 8.63, 8.89), (8.82, 3.70, 8.75, 8.74))$ with $\hat{a}_0 = 82,869.64$ and $\hat{f}_0 = 1$. (b) shows trajectory of (s, b) with $a = 93,142$.

5 Discussion & Conclusion

In the proposed optimization framework, we found that $\hat{a}$ did very well at generalization. However, $\hat{f}$ was less accurate when it deviated from the sample points, despite having average 0.9 out of sample accuracy. Algorithm 1 allowed us to take advantage of the approximation of $\hat{f}$ to f around the sampled points. One alternative to using Algorithm 1 would be to characterize stability as $\int_0^T S(t; r(s, b))dt \approx \sum_{i=0}^T S(i; r(s, b))$ and define $f := \mathbb{1}(\int_0^T S(t; r(s, b))dt > \xi)$, where ξ is some threshold hyperparameter. It is not clear whether this characterization would be better than $f := \mathbb{1}(T > 90)$, but it is worth considering in future work. In terms of physical observations, we noticed that our best rockets had fins in a "swept-back" shape (see 1), commonly found in many fighter jets (e.g. Nighthawk (3c)). It would be interesting to compare these results to those obtained from Bayesian optimization (A.1) if an easier way of online sampling were available.

Overall, we have outlined and demonstrated the effectiveness of a systematic regression process for optimizing fin parameters. The process allowed us to discover multiple stable fins that achieve superior peak altitudes than the best derived from the sample dataset, along with observations of optimal fin structures that can be used to guide future investigation. The process can be easily adapted to other settings involving different engine files and base fin shapes.

References

[1] Bunkley, Justyn, Caballes, Marc J.L., Ajuwon, Margaret & Chen, Guangming. (2022). Design Analysis of Rocket Tail Fins Aimed at Higher Apogee by Computer Simulation. *American Society for Engineering Education*.

[2] Bertsekas, Dimitri (1999). Nonlinear Programming (2nd Edition). *Athena Scientific*, pp. 23-24.

[3] Rasmussen, C. E. & Williams, C. K. I. (2006). Gaussian Processes for Machine Learning. *MIT Press*.

A Appendix

A.1 Bayesian Optimization

Since each sampling of a parameter vector is expensive in our case, we also tried Bayesian optimization. It selects the most promising data points to sample given our current knowledge of data points sampled.

We define our objective function for a particular rocket r parameterized by s and b as $g(s,b) \equiv a(s,b) + \alpha \min_t S(t; r(s,b)) \equiv a(s,b) - \alpha S(s,b)$, where $S(s,b)$ is the minimum stability over the entire flight. This function penalizes instability and rewards altitude. The α hyperparameter can be fine-tuned, and in our case we set $\alpha = 10000$ so that negative stability will significantly penalize the objective function, while positive stability, on the order of 1, will only be a little significant in affecting the objective function. We now solve $\max_{s,b \in M} g(s,b)$.

We assume a conditional distribution $P(g(x)|g(x_1) = y_1, \dots g(x_n) = y_n)$, known as the surrogate model, where (x_1, y_1) through (x_n, y_n) are the current data points sampled. We use the Gaussian process surrogate prior, which assumes that our data can be modeled as a multivariate-Gaussian distribution with covariance matrix given by kernel function $K(x_i, x_j)$. We use a radial basis kernel $K(x_i, x_j) = \exp\left(-\frac{d(x_i,x_j)^2}{2\ell^2}\right)$, where our length scale ℓ is fixed to be 1 and $d(x_i, x_j)$ stands for Euclidean distance metric. Our conditional distribution rewrites as

$$P \sim \mathcal{N}\left(k\Sigma y, K(x,x) - k^T \Sigma^{-1} k\right),$$

where $k \equiv [K(x_i, x)]$, $y = [y_i]$, and $\Sigma \equiv [K(x_i, x_j)]$.

We then select an acquisition function $u(x|x_1, \dots x_d)$ which represents how much we should select the point x next. We use the *Probability of Improvement (PI)* metric, which is $x^* = \text{argmax}_x P(g(x) \geq y^* | g(x_1) = y_1, \dots, g(x_n) = y_n)$, where $y^* = \max_i y_i$. When selecting the next point to sample, we consider a random sample of $N = 100$ data points $x = (s,b) \in [2,10]^8$ and evaluate u on each x, taking the x with the largest value.

We note that hypothetically our y values should be modeled as $g(x) + \epsilon$ where ϵ is some Gaussian noise. This would be appropriate for our situation since our rocket could be affected by random variables like wind speed in real life. However, our simulator cannot simulate randomness, so we do not need to introduce ϵ.

Since it is expensive to do online sampling, we implement a small-scale version of our algorithm on 7 datapoints, two of which are randomly sampled to initialize our algorithm, and the rest of which are selected according to the PI acquisition function. We are able to obtain $y^* = 76626.86$ and $a(s,b) = 63453.27$. We notice that uncertainties for each of the y_i are quite low ($\approx 10^{-3}$), suggesting we did not sample enough points for our space to become saturated. Considering our regression used hundreds of data points, this suggests that a faster online sampling technique would be necessary to evaluate this method.

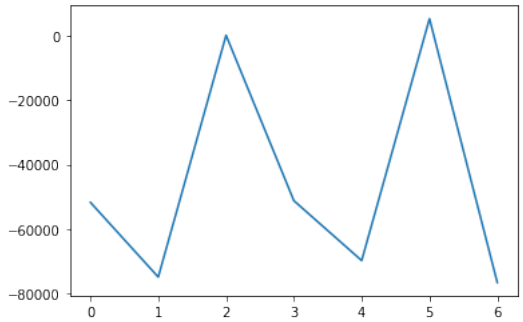


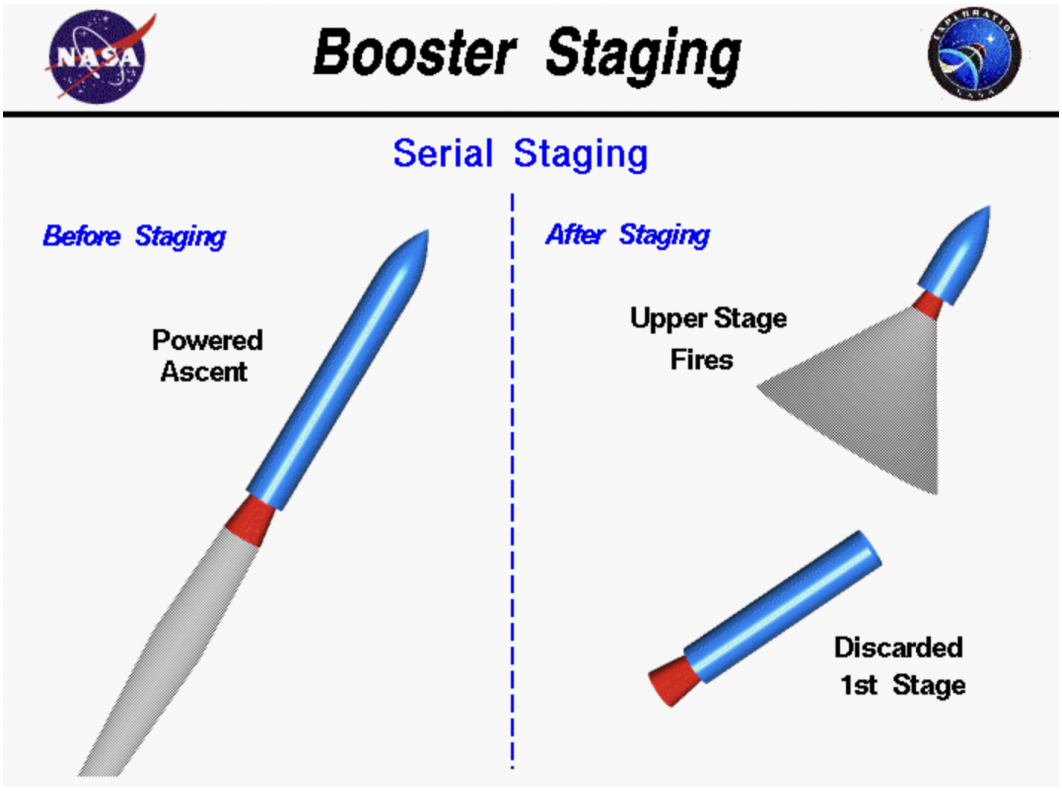
Figure 2: Evolution of the objective function $-g(s,b)$ over the samples.

Notice from the figure that we do not yet see any consistent trend in the objective function. However, it is promising that there are multiple points that are at least stable, though this could also be due to the initialization. Nevertheless, this plot supports our claim that a faster online sampling method would be preferred for better evaluation of the method. It would be interesting to see how this approach would do if we could generate more samples.

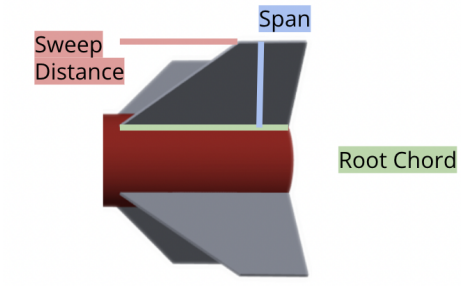
A.2 Neural Network Model

The Neural Network used contains 4 hidden layers. The input is 45-dimensional and mapped through fully connected layers of 90, 60, 30, 10 dimensions with ReLU activation. The final layer is then combined into the final output with Sigmoid activation. We train with binary cross entropy loss and SGD with learning rate $\alpha = 0.01$ for 250 epochs.

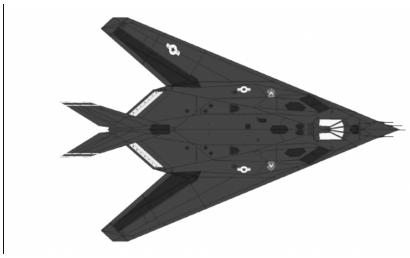
A.3 Supporting Figures



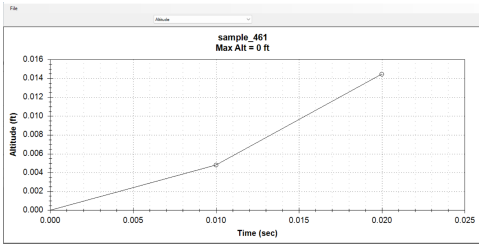
(a) Two Stage Rocket.



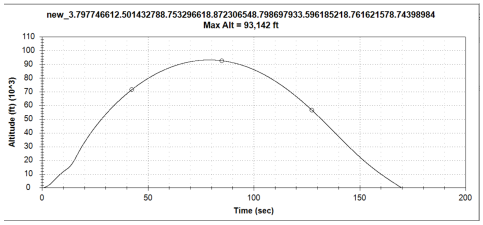
(b) Labeled Fin Parts



(c) Nighthawk Fighter Jet



(a) Failed Rocket



(b) Stable Rocket

A.4 Code Repository

All code can be found at https://github.com/alex-t-hu/rocket.