Optimal placement of reaction control thrusters conforming to a spacecraft’s surface

Table of contents

1 Table of contents

$\overrightarrow{M}$	Moments exserted by the thruster system
$\vec{F}$	Forces exserted by the thruster system
$\vec{T}$	Thrusts of each thruster (can be negative)
$\vec{r}_i$	Position of thruster i
$\vec{d}_i$	Normalized direction of thruster i
$\vec{u}$	x-position of the thrusters on the 2d map
$\vec{v}$	y-position of the thrusters on the 2d map
$\vec{a}$	Angle of the thrusters on the surface
A	Thruster matrix
$\vec{x}$	Design variables
$f(\vec{x})$	Objective function
$g(\vec{x})$	Inequality constraint
$\vec{\phi}(x)$	2D Path representing the surface

2 Introduction

In order to control attitude and translation, spacecraft can make use of reaction control thrusters. Reaction control thrusters are small thrusters that can expel gas in a certain direction to impart a controlled force at a specific point of the vehicle into a specific direction. Ideally, a combination of a sufficient amount of these thrusters on specific points of the craft can provide any combination of torque and force within a certain magnitude. Some thruster arrangements may be more or less efficient at converting their thrust into certain torques or moments than others.

The purpose of this report is to find the optimal arrangement of thrusters on a certain spacecraft. The spacecraft in question is shown in Figure 2. This is a spacecraft design I came up casually some years ago. It is chosen as it provides a geometry which is not to complex, but where the optimal solution is not immediately obvious. The geometry also contains holes, and some sharp angles, which might make the optimization more interesting. The thruster design used is as shown in Figure 1. The reason for this design is the bi-directional thrust, which allows the thrust to be negative in the mathematical model. This significantly simplifies the calculations. The challenge of the optimization is mostly the restrictions of the thruster placement. The thrusters need to lie on the surface of the vehicle and the thrust direction needs to be parallel to the surface.

A more rigorous formulation of the optimization problem and the model is given in Section 3. Some of the mathematical properties of the objective function, like bounds/continuity and noise are discussed in Section 4. The optimization of a simplified 2D problem is discussed Section 5 while the actual optimization is discussed in Section 6. In Section 7, the results of the optimization are commented on and their sensitivity is analyzed. The final conclusion is presented in Section 8.

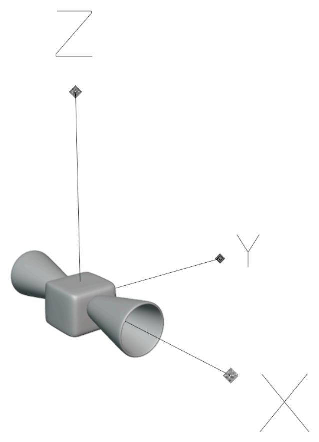

Figure 1: The concept of the thruster design, thrust can be applied in the positive or negative x-direction

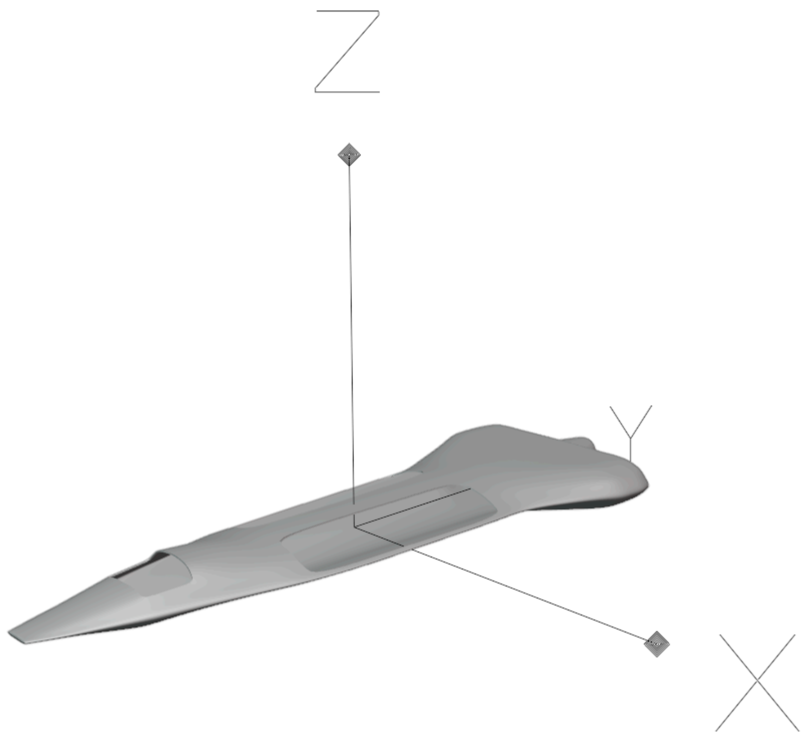

Figure 2: The spacecraft in question. Only the surface where thruster placement is allowable is visible

3 Problem formulation

The moment (M_i) and force (F_i) a single thruster imparts on the spacecraft can be written as:

$$\overrightarrow{M_i} = T_i\left(\vec{r}_i \times \vec{d}_i\right)$$
$$\vec{F}_i = T_i\vec{d}_i$$

where $\vec{r}_i$ is the position of the thruster, $\vec{d}_i$ is the direction and T_i the thrust, that thruster exerts. The final forces and moments are the sum of the forces and moments from the individual thrusters.

$$\begin{aligned}\begin{bmatrix}\overrightarrow{M}\\ \vec{F}\end{bmatrix} &= \begin{bmatrix}\vec{r}_1 \times \vec{d}_1 & \cdots & \vec{r}_n \times \vec{d}_n \\ \vec{d}_1 & \cdots & \vec{d}_n\end{bmatrix}\begin{bmatrix}T_1\\ \vdots \\ T_n\end{bmatrix}\\ &= A\vec{T}\end{aligned}$$

If $n = 6$, A is a square matrix and, if A is invertible, the thrust of the thrusters can be calculated as

$$\vec{T} = A^{-1}\begin{bmatrix}\overrightarrow{M}\\ \vec{F}\end{bmatrix}$$

If A is not invertible, there are forces/moments that the design cannot provide. The goal of this project is to find a combination of thrusters on a specific spacecraft that most 'efficiently' provide the thrusts for certain input thrusts/torques, i.e. where the average magnitude of $\vec{T}$ is minimal.

3.1 Simplifications

Several assumptions are made in the problem statement that are be stated below.

- The geometry of the spacecraft does not influence the thrust of the thrusters. Thruster are considered to always give constant thrust no matter where they are placed
- Thrusters can be placed anywhere on the surface. Some of the surface has been omitted from the model to indicate places where no placement is possible. The remaining surface is all considered an equally valid destination. In reality, the placement of some thrusters might restrict the valid placement of others.
- All forces and torques are weighed equally. For the purpose of this optimization, 1 unit of force has the same utility as one unit of torque. Since all points in this problem lie within $[-1, 1]^3$, force and torque have similar scales. The importance of different forces/moments can easily be weighted by scaling the rows of the matrix A to account for e.g. the mass moment of inertia.

3.2 Optimization problem

The initial attempt to represent the 'average magnitude of $\vec{T}$' mathematically is to solve $\vec{T}$ for multiple predefined values of $\left[\overrightarrow{M}, \vec{F}\right]^T$ and consider the (smooth) maximum absolute value between all the solutions the value to minimize. However, this approach brought multiple problems:

- Only a finite subset of all possible inputs can ever be represented
- The maximum thrust is considered, where in practice the average thrust is more important as it relates to fuel consumption
- The function evaluation involved solving multiple 6x6 system of equations, which is rather slow.
- There are many cases where A is not invertible.

Instead, I settled on minimizing of $\left|\det(A^{-1})\right|$, since the determinant is representative of how much a vector is scaled. Minimizing $\left|\det(A^{-1})\right|$ is equivalent to maximizing $|\det(A)|$, since $\det(A^{-1}) = \det(A)^{-1}$. To archive a smoother version of the absolute value, the final optimization function is.

$$f(x) = -\sqrt{\det(A)^2 + 0.1^2}$$

$$A = \begin{bmatrix}\vec{r}_1 \times \vec{d}_1 & \cdots & \vec{r}_6 \times \vec{d}_6 \\ \vec{d}_1 & \cdots & \vec{d}_6\end{bmatrix}$$

3.3 Modelling aspects

Since the matrix A is constructed from the vectors $\vec{r}_1 ... \vec{r}_6$ and $\vec{d}_1 ... \vec{d}_6$, it is easy to conclude to model the problem as a 36-dimensional problem with equality constraints that force $\vec{r}_i$ to conform to the surface of the vehicle and $\vec{d}_i$ to be normalized and perpendicular to the the surface. However, there are 2 ways to reduce the dimensionality of the problem while simultaneously reduce the amount of constraints.

Firstly, since each thruster has to be confined to the surface, the position can be defined by a 2d vector, which has the components u_i and v_i. To retrieve the 3d position form a 2d vector, an image as shown in Figure 3 can be used. Each point on the image represent a vector on the surface of the object. This is a technique common in computer graphics, called UV-mapping . The red, green and blue components of the color in the image correspond to the x, y and z components in 3d space.

A similar mapping process can be done with the surface normal vector. This map is split up into the positive and negative components of the normal vector (Figure 5 and Figure 6), because of the limitations of the images and the tools to create them. Subtracting Figure 6 from Figure 5 gives the true normal vector. Since the direction is orthogonal to the surface normal, its rotation can now be expressed by a single variable, called a_i. The direction is retrieved by building an orthonormal basis around the normal vector via the Gram-Schmidt process, where $\pi \cdot a_i$ defines a rotation in the plane orthogonal to the normal vector.

These two simplifications allow to reduce the problem from 6 variables per thruster to 3, yielding 18 variables in total. In addition the large amount of complicated constraints has been reduced to a single constraint: the 2d-coordinates of all thrusters must fall within the area for which a surface point is defined. For this, a signed distance field is defined, as seen in Figure 4. The signed distance field defines the distance of each point to a surface , with points within the object being negative. Therefore the signed distance field can be used directly as a constraint for a single thruster. The final inequality constraint is the maximum sdf of all thrusters.

$$g(\vec{u}, \vec{v}, \vec{a}) = \operatorname*{Max}_i(\mathrm{sdf}(u_i, v_i)) \leq 0$$

$\vec{u}$, $\vec{v}$ and $\vec{a}$ can take any value, but repeat outside of the (0, 1) domain.

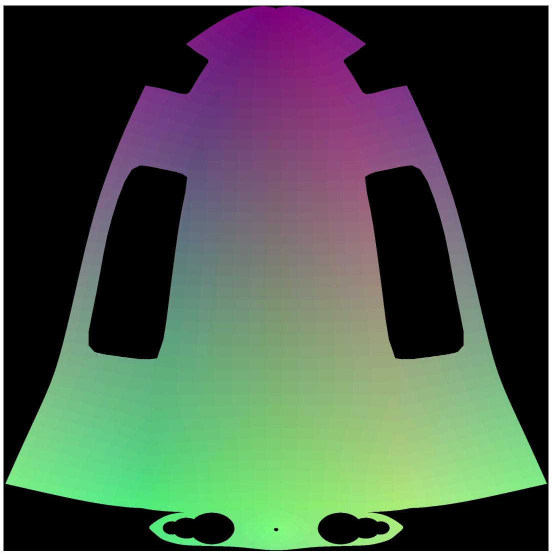
Figure 3: Position map. Image is remapped from (0, 1) to (-1, 1)

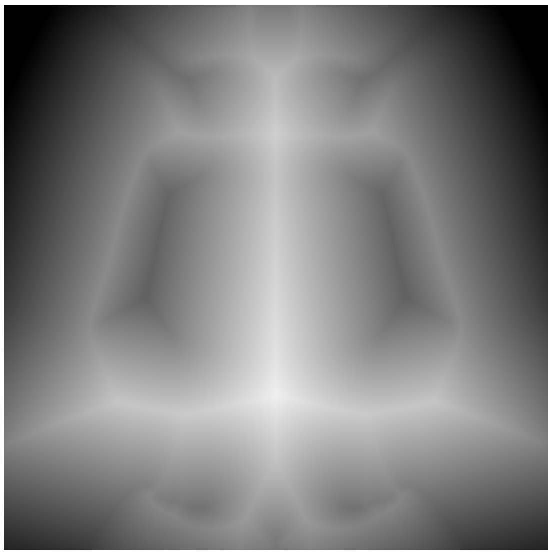
Figure 4: Inverse SDF map. Image is remapped from (0, 1) to (1, −1) before use

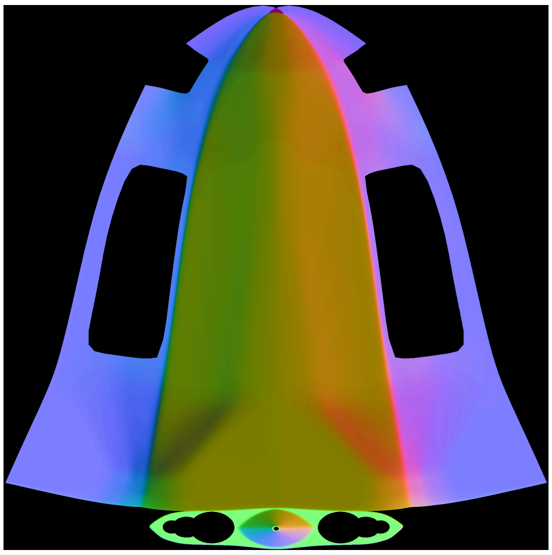
Figure 5: Positive normal map

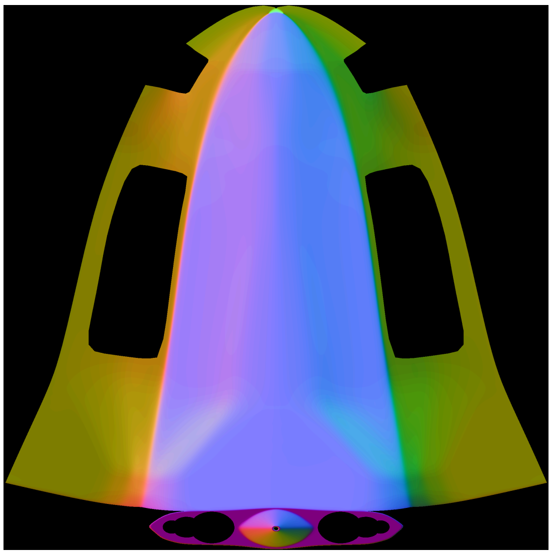
Figure 6: Negative normal map

Lastly, the symmetry of the spacecraft can be exploited to place pairs of thrusters instead of individual thrusters and maintain the symmetrical design. This has the risk of eliminating potential optimal solutions that are either non-symmetrical or has thrusters at the boundary, but reduces the number of design variables to 9. This is performed by representing the symmetry of the 3d model in the symmetry of the UV-map and setting the the state of the last 3 elements as written below.

$$\begin{aligned} u_{4,5,6} &= 1 - u_{1,2,3} \\ v_{4,5,6} &= v_{1,2,3} \\ a_{4,5,6} &= -a_{1,2,3} \end{aligned}$$

4 Initial problem investigation

There is no reason to assume that the objective function is either monotonous or convex and it will be clear in the simplified problem that it isn't. It is also clear from Figure 3 that the domain is not convex as well.

4.1 Bounds

The optimization function is

$$f(\vec{x}) = -\sqrt{\det(A)^2 + 0.1^2}$$

Whose upper bound occurs as $\det(A) = 0$. So $f(x) \leq -0.1$. To find the lower bound, the bounds of $|\det(A)|$ need to be known. Hadamards inequality states that $|\det(A)| \leq \prod_{i=1}^{n} \|\vec{w}_i\|$, where $\vec{w}_i$ are the column vectors of A [1]. In this case,

$$\begin{aligned} \vec{w}_i &= \left[\vec{r}_i \times \vec{d}_i \vec{d}_i\right] \\ \Leftrightarrow \|\vec{w}_i\| &= \sqrt{\left\|\vec{r}_i \times \vec{d}_i\right\|^2 + \left\|\vec{d}_i\right\|^2} \\ &\leq \sqrt{\|\vec{r}_i\|^2 + 1} \end{aligned}$$

The maximum value of $\|\vec{r}_i\|$ across the whole surface is around 1.087. Therefore:

$$\det(A)^2 \leq \left(1.087^2 + 1\right)^3 \cong 10.383$$
$$-10.383 \leq f(\vec{x}) \leq -0.1$$

This gives a lower bound for the optimum.

4.2 Smoothness & Continuity

The determinant varies smoothly with respect to the components of the matrix, so $f(\vec{x})$ is smooth with respect to the positions and directions of the thrusters $\vec{r}_i$ and $\vec{d}_i$. The direction also varies smoothly with the angle a_i. Thus the question of smoothness and continuity relies on the smoothness of the position/normal maps. The position map is continuous because of the way it is generated: adjacent, valid points on the 2d map are also adjacent in 3d space. Since the surface normal is related to the gradient of the surface, the normal map represents the derivatives of the position map. Therefore the normal map is discontinuities if and only if the position map is non-smooth. This corresponds with sharp angles in the geometry of the mesh. Even though the shape is mostly smooth, there are some sharp corners in the mesh and some very tight bevels (only a few pixels wide), which are represented as very steep gradients in the normal map. Therefore there are some discontinuities or near discontinuities in the objective functions.

4.3 Noise and precision

Storing data in images comes with two fundamental restriction in resolution. Both the resolution of the data itself, as well as the spatial resolution are limited by the image format. The position and normal maps are 2048×2048 pixel maps with a color depth of 16 bits per channel. The image is sampled by linearly interpolating between the closest pixels and can be considered, on a small scale, to consist of facets of size $5 \cdot 10^{-4}$. This can influence finite difference calculations, especially around sharp corners. The 16 bit color depth makes the normals accurate to the order ~$1.5 \cdot 10^{-5}$ and positions accurate to the order ~$3 \cdot 10^{-5}$. This is sufficient for the problem, and a much less dominant restriction than the image resolution.

I could not get an SDF image in 16 bit per channel color depth. Therefore it is also limited to 1024 × 1024 resolution as an SDF that decreases by less than 1 unit per pixel can lead to problems. This means that he constraint has significantly lower resolution than the optimization function. Because of the type of optimizer chosen and the way the constraint is implemented, however, the noise in the constraint is much less relevant.

5 Initial optimization on simplified problem

5.1 Simplified problem model

Before solving the full problem, a simplified version of the problem is investigated in order to gage the complexity and nature of the optimization function. An easy way to do this is to look at the 2-dimensional equivalent of the original problem.

In the 2-dimensional case, there are 2 degrees of translational motion and a single degree of rotational motion. Therefore only 3 thrusters are necessary. Instead of defining the position of the thrusters as 2d point that maps to a 3d surface, it can be defined as a single value that maps to a 2d position via evaluating a path representing the surface. The constraint that the thrust direction needs to be orthogonal to the surface allows to get the direction directly from the position, as the normalized derivative of the the path. This reduces the number of variables in the 2D case to only 3. Similarly to the actual problem, the use of symmetry allows to reduce the problem to 2 variables by assuming that $x_3 = 1 - x_1$.

The path $\vec{\phi}(x)$is represented by interpolating between 20 points with piecewise cubic hermite spline, as seen in Figure 9. This way, some of the flat surfaces and sharp corners of the shape stay conserved while still offering smooth interpolation in some places. The path starts at (0.5, 0), reaches (0.5, 1) at $x = 0.5$ and returns to (0.5, 0) at $x = 1$. It is symmetrical along $x = 0.5$.

$$\vec{r}_{i,\,\text{2D}} = \vec{\phi}(x_i)$$

$$\vec{d}_{i,\,\text{2D}} = \frac{\mathrm{d}\vec{\phi}(x_i)}{\mathrm{d}x_i} \cdot \left\| \frac{\mathrm{d}\vec{\phi}(x_i)}{\mathrm{d}x_i} \right\|^{-1}$$

$$A_{\text{2D}} = \begin{bmatrix} |\vec{r}_{1,\,\text{2D}} \;\; \vec{d}_{1,\,\text{2D}}| & |\vec{r}_{2,\,\text{2D}} \;\; \vec{d}_{2,\,\text{2D}}| & |\vec{r}_{3,\,\text{2D}} \;\; \vec{d}_{3,\,\text{2D}}| \\ \vec{d}_{1,\,\text{2D}} & \vec{d}_{2,\,\text{2D}} & \vec{d}_{3,\,\text{2D}} \end{bmatrix}$$

5.2 Algorithm selection

The reduction to two variables allows to plot the function across the whole domain at once, as seen in Figure 7. Furthermore, a cross-section along the $x_1 = 0.5$ is shown in Figure 8. These two figures confirm two of the difficulties that where suspected earlier: The objective function has many local optima and it contains a lot of plateaux and discontinuities.

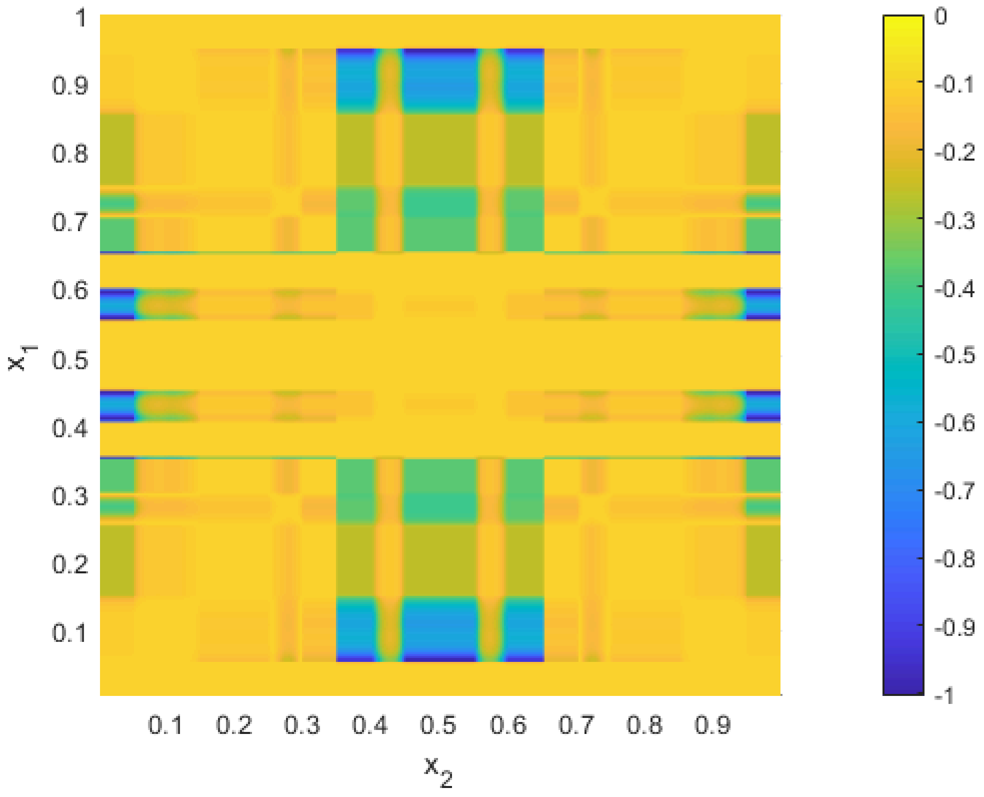

Figure 7: Plot of $f(\vec{x})$ for the simplified problem over its entire domain

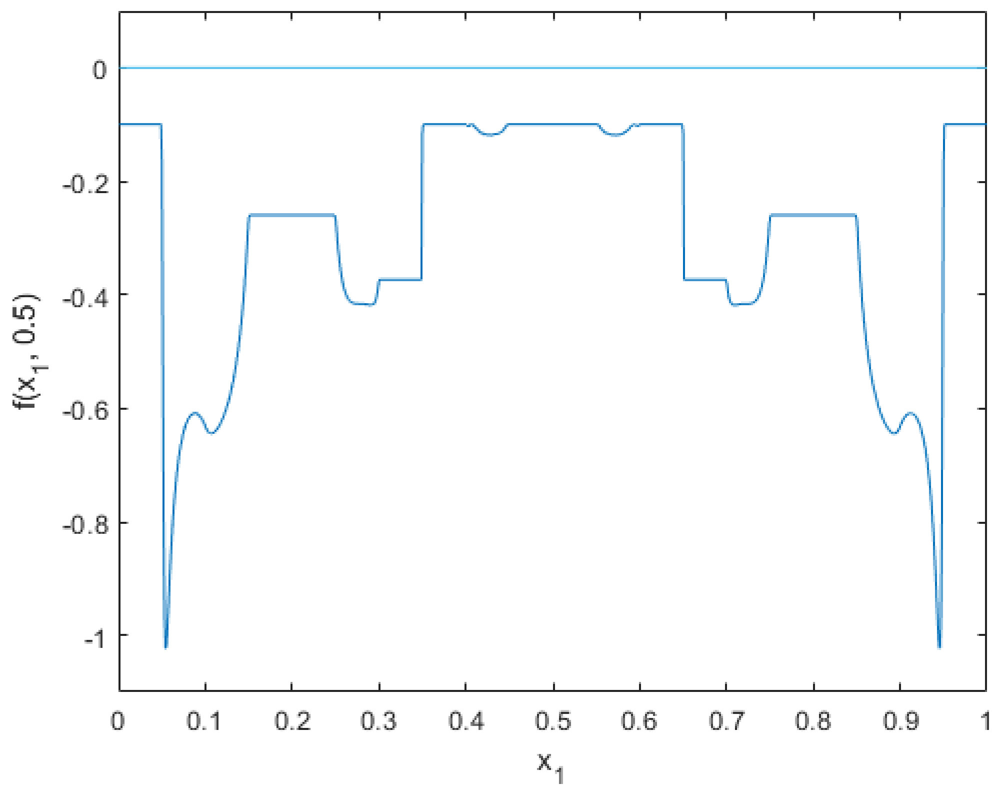

Figure 8: Plot of a cross-section of $f(\vec{x})$ at $x_1 = 0.5$ for the simplified problem

This leads to the following criteria to choose an optimization algorithm:

1. The optimizer must be able to escape local optima.
2. The optimizer must not rely on the first/second derivative.

The last condition eliminates almost any method that are not direct search methods. Direct search methods include stochastic methods like simulated annealing or genetic algorithms, as well as methods like cyclic coordinate search and the Nelder-mead simplex method. My initial thought was to use a stochastic method for their ability to escape local minima. I chose to use simulated annealing for its simplicity to implement. Indeed, for this relatively simple problem, simulated annealing performs relatively well and finds the global optimum in most runs. There are multiple global optima because of the symmetry in the objective function, but a solution is shown in Figure 9.

The thrusters are as far as possible from each other to maximize the angular momentum they can provide and are 'balancing' on the corners to be able to provide thrust in both x- and y- directions. Although this makes the solution technically very sensitive to the inputs, it doesn't make for an unreasonable design. This will be discussed more in the sensitivity analysis of the main problem.

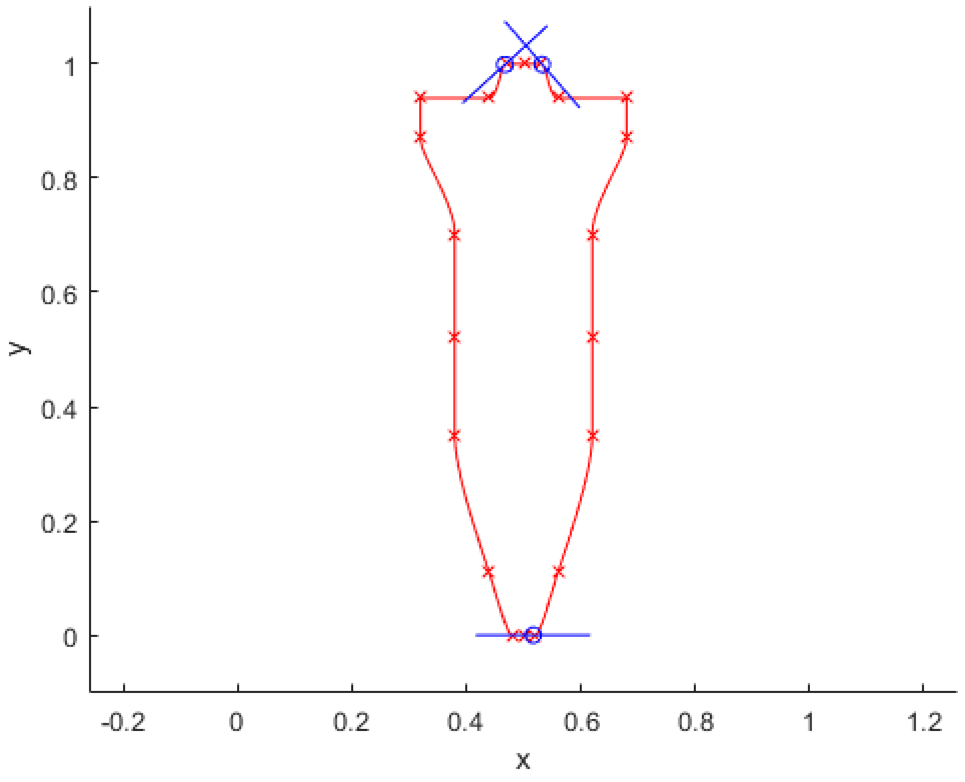

Figure 9: Plot of the path representing the 2d shape in red. Position of the thrusters in the solution plotted in blue.

6 Optimization of the actual problem

The actual 3D problem has 9 design variables, so it is not possible to construct a plot like Figure 7, but similar properties can be assumed, since flat surfaces and tight corners are also present in the 3D case. In this more complicated case, simulated annealing turned out to be very unreliable and often converged to results which were obviously not optimal.

The reason for this might be the ease for simulated annealing to get trapped at low temperature within shallow plateaux which are local optima. This can be mitigated by 'reheating' the solution after it is stuck for to long, but the optimizer is still very inefficient in finding the right solution. Since there are a lot of variations of simulated annealing and a lot of parameters that can be adjusted, there might be a more efficient version of simulated annealing that can deal with this problem, but there is also a simpler way.

An efficient algorithm, robust to both un-smooth functions and plateaux, can be run on many random points on the domain in hopes that it finds a good global optimum in the process. After some testing, Powell's conjugate directions (PCD) method was chosen [2]. Since PCD is based on line-searches that use golden-section search or quadratic interpolation internally [3], it does not rely on derivatives and is pretty robust.

A major benefit of PCD in this case is that the search radius can be defined exactly with the bounds of the line-searches. The optimization performs better and converges faster on a smaller search radius, centered around 0. By limiting the search radius of the algorithm, the optimizer needs to deal with multiple optima less often and is more likely to find the closest local optimum quickly. It is also less likely to cycle between different optima without converging, as less of the function is visible at a time. This comes with the disadvantage that less of the solution space is explored at a time, so more cycles might be needed to find the local minimum. A good compromise is to set the search radius to 0.25.

Since PCD, like most direct search methods, does not account explicitly for constraints, the objective function needs to be transformed to give a penalty outside the feasible domain, such that the optimal solution cannot lie outside the domain. The follow transformation is used.

$$\tilde{f} = f + p(\max(0, g))^2 \quad p \in \mathbb{R}^+$$

where p determines the magnitude of the penalty. p can be set quite high right away, since the optimizer needs to deal with very steep gradients (almost discontinuities) anyway. The value used is $p = 1000$.

To generate an initial value to use for the PCD, a random vector is generated with a uniform distribution in the domain $[0, 1]^9$ and re-generated until it conforms to the inequality constraint. After some initial tests, it was found that all the most promising results occur with 2 sets of thruster at $v_i \geq 0.5$ (the upper half of the UV map, corresponding to the rear of the spacecraft) and one set at $v_i \leq 0.5$. This makes physical sense, as the thrusters would benefit to be evenly distributed over the spacecraft to gain the maximum moment arm, with the rear of the spacecraft being wider and better for roll control. Ultimately, the domain for the initial value generation was constrained to accommodate these preferences and make the total search space of initial guesses smaller. Along with the restrictions in v_{i_0}, $u_{i_0} < 0.5$ can be enforced, as the $f(x)$ is symmetric with respect to u_i. This reduces the hypervolume of the search space by a factor of $2^6 = 64$.

The interval of convergence for the PCD is set to a rather large number, 0.01. The idea is to have an algorithm that converges quickly, such that it can be run many times to find a good candidate for the optimal solution, then find the precise optimum with that candidate as a starting point in a second step.

7 Interpretation of results

After running the program multiple times with different rng states, most come up with similar solutions. This is a good sign, as it is likely that this family of solutions are include the global optimum. It is impossible to say though if any solutions is the true global minimum unless a better solution is found. The result that is shown here is the best one found with a utility function of $f(x) = -2.8347$. The values in terms of map coordinates (u, v) and a are presented in Table 2, while the solutions in terms of position $(\vec{r})$ and direction $(\vec{d})$ are presented in Table 3. A 3d visualization of the solution is also visible in Figure 10.

Table 2: Solution in terms of map coordinates of first 3 thrusters

i	1	2	3
u	0.2355	0.4431	0.2622
v	0.1255	0.9431	0.1081
a	0.7742	0.0647	0.1260

Table 3: Solution in terms of 3d positions and directions

i	1	2	3	4	5	6
r_x	−0.3320	−0.0473	−0.3587	0.3327	0.0473	0.3584
r_y	0.8432	−0.9758	0.8344	0.8431	−0.9763	0.8342
r_z	0.0342	0.0036	−0.0050	0.0336	0.0033	−0.0060
d_x	0.6213	−0.6935	0.2543	0.6184	−0.5928	0.3233
d_y	0.6593	0.0775	0.3084	−0.6590	−0.0769	−0.3085
d_z	0.4234	−0.7032	−0.9165	−0.4281	0.7854	0.8945

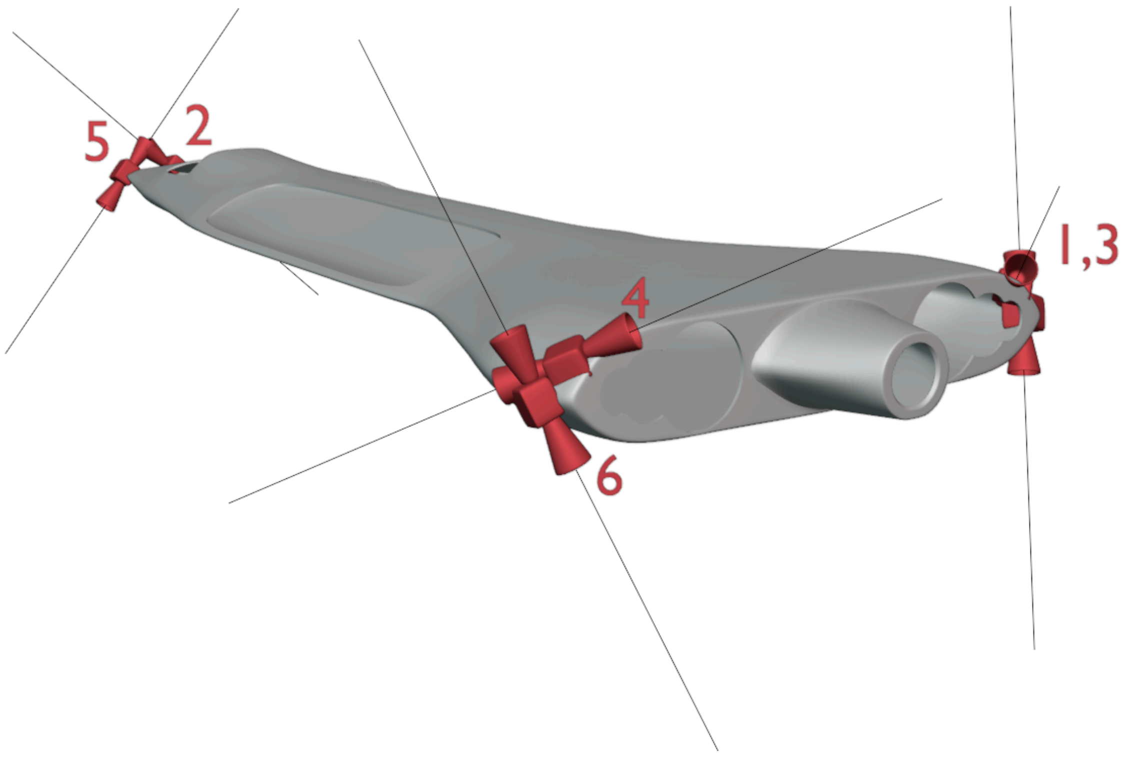

Figure 10: Solution rendered in 3d

It can be seen from Figure 10 that the thrusters from the solution form 3 pairs, one at each extremity of the spacecraft. Each pair forms a perpendicular cross. This solution maximizes both the distance of the thrusters from each other while making sure there is minimum alignment between any two thruster. It makes sense physically that this is one of the best solution that can be archived.

7.1 Sensitivity analysis

For this problem, a sensitivity analysis comes in two forms. One is the sensitivity with respect to the design parameters that have been optimized, which is what is typically done. These parameters, however have been chosen to reduce the dimensionality of the problem and do not correspond to any physical dimensions. Therefore, the sensitivity with respect to the position and direction is also calculated.

The single constraint was not an active constraint in the solution, so the only response that is relevant is the response of the objective function itself. Since the relevant point is a local optimum that does not lie on any constraint, the derivatives at that point with any value are either zero or the function is not smooth/discontinuous. Discontinuities can only come from the position/normal maps themselves. After looking at the discrete gradient of the normal map, it is found that points 2 and 4 (the pair in the front of the spacecraft) lie on a crease, so they have discontinuous derivatives and very large sensitivity. The other points lie on a smoother surface, so the derivative with respect to them is zero. The thruster pair is, in a way, exploiting the interpolation between pixels in the normal map to find a very narrow solution that results in a right angle for the thrusters. This makes the solution very sensitive with respect to the placement of thruster 2/4 on the 2d map, but this sensitivity does not have any physical meaning.

The sensitivity with respect to the positions and directions of the thrusters has a more direct physical meaning. It represents how much the objective function changes with manufacturing defects, design adjustments and physical deformation (due to e.g. temperature or stress).

The derivative with respect to each position $\vec{r}_i$ and direction $\vec{d}_i$ can be calculated analytically.

$$\frac{\mathrm{d}f}{\mathrm{d}[\vec{r} \;\; \vec{d}]^T} = \frac{\mathrm{d}f}{\mathrm{d}\det(A)} \frac{\mathrm{d}\det(A)}{\mathrm{d}A} \frac{\mathrm{d}A}{\mathrm{d}[\vec{r} \;\; \vec{d}]^T}$$

$$\frac{\mathrm{d}f}{\mathrm{d}\det(A)} = \frac{\det(A)}{f} \approx -1$$

For the second derivative, Jacobi's formula can be used

$$\frac{\mathrm{d}\det(A)}{\mathrm{d}A} = \operatorname{adj}(A) = \det(A)A^{-1}$$

To get $\frac{\partial A}{\partial[\vec{r} \;\; \vec{d}]^T}$, both matrices can be written in vector form.

$$\partial \begin{bmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{56} \\ A_{66} \end{bmatrix} = \begin{bmatrix} \Omega_{r_1} & \Omega_{d_1} & & \cdots & 0 \\ & I & & & \vdots \\ & & \ddots & & \\ \vdots & & & \Omega_{r_6} & \Omega_{d_6} \\ 0 & \cdots & & & I \end{bmatrix} \partial \begin{bmatrix} \vec{r}_1 \\ \vec{d}_1 \\ \vdots \\ \vec{r}_6 \\ \vec{d}_6 \end{bmatrix}$$

where

$$\vec{r}_i \times \vec{d}_i = \Omega_{r_i} \vec{r}_i = \Omega_{d_i} \vec{d}_i$$

Finally, it needs to be considered that the direction vectors $\vec{d}_i$ are always normalized. Therefore a change in one dimension of $\vec{d}$ will have indirect changes, as the other components adjust.

Since, for any pair of vector components

$$d_i^2 + d_j^2 + \ldots = 1$$

$$\Leftrightarrow \frac{\partial d_i}{\partial d_j} = -\frac{d_j}{d_i}$$

The total derivative can be calculated as:

$$\frac{\mathrm{d}f}{\mathrm{d}d_i} = \frac{\partial f}{\partial d_i} + \frac{\partial f}{\partial d_j}\frac{\partial d_j}{\partial d_i} + \frac{\partial f}{\partial d_k}\frac{\partial d_k}{\partial d_i}$$

$$= \frac{\partial f}{\partial d_i} - \frac{\partial f}{\partial d_j}\frac{d_i}{d_j} - \frac{\partial f}{\partial d_k}\frac{d_i}{d_k}$$

$$\frac{\mathrm{d}f}{\mathrm{d}\vec{d}} = 2I - \vec{d}\left(\frac{1}{\vec{d}}\right)^T \frac{\partial f}{\partial \vec{d}} \qquad \frac{1}{\vec{d}} \text{ refers here to the element-wise inversion}$$

The final sensitivities are presented in Table 4. Since both position and direction have a similar scaling (from −1 … 1), it makes sense to present the sensitivity in linear derivatives, instead of logarithmic ones. The highest sensitivity is in r_{x_3} with an absolute value of 3.2. For reference: If the vehicle has a length of 10m, then a displacement of r_{x_3} of 10 mm in the most sensitive direction results of a change in the utility function of 0.2 %, which I consider sufficiently stable.

Furthermore, it can be seen that some thrusters are more sensitive to design changes than other ones. Thruster pair 2/5, for example is much less sensitive than pair 3/6. The dimension at which the highest sensitivity for a thruster occurs can also give insight into the purpose of the unit. Since thrusters 2/5 are more sensitive to changes in r_y and d_z, it can be deduced that is most important in pitch maneuvers (torque around the x-axis). With this information, the solution can be altered manually to better integrate it with the rest of the design, while maintaining its performance as much as possible.

Table 4: The sensitivity with respect to different $\vec{r}_i$ and $\vec{d}_i$, colored by magnitude

i	1	2	3	4	5	6
r_x	−1.2418	−0.0696	−3.1924	1.266	0.0599	3.1779
r_y	0.6157	−1.3783	0.7497	0.6488	−1.4026	0.7667
r_z	0.8635	−0.0832	−0.6336	0.83	−0.0922	−0.8844
d_x	−1.9138	−0.0374	−0.6434	−2.005	0.0287	−1.0214
d_y	−1.2718	−0.3768	−0.8854	1.4866	0.3834	0.7276
d_z	−1.2939	1.1169	3.0216	1.1591	−1.3929	−2.3653

8 Conclusion & Recommendations

The objective of this optimization problem was to find the optimum thruster placement for a specific spacecraft. Nominally the 6 necessary thrusters define a 36-dimensional design space with many quite complex constraints, as the thrusters need to conform to the surface of the spacecraft. With usage of position/normal maps however, the problem can be restated with 18 design variables, and a single (non active) constraint. With the use of symmetry, the amount of design variables can be further reduced to 9. This restatement comes with the cost of introducing noise, discontinuities and plateaux into the objective function, which limits the choice of optimization function. Ultimately, running Powell's conjugate direction method with a limited search radius from multiple randomly chosen starting points yields a good result most of the time.

The best solution found is presented in Figure 10. The sensitivity within the restated problem is very high, as one thruster lies on a sharp edge, corresponding to a near discontinuity in the objective function. The solution is much less sensitive in the original problem, which is much more representative of the physical reality.

The success of this methodology relies heavily on the geometry and the UV-map that maps the 3d surface to a 2d plane. Geometry with less sharp corners and less flat planes will be easier to optimize. The choice of mapping also strongly affects the objective function. The reduction of design variables by employing symmetry required a specific mapping to work. For some geometries, the original 36-dimensional problem formulation might be more advantageous, since the objective function is much smoother, and can be looked into for future research. The main issue with that approach is to define the problem constraints mathematically.

Bibliography

[1] J. Hadamard, “Etude sur les propriétés des fonctions entières et en particulier d'une fonction considérée par Riemann,” *Journal de Mathématiques Pures et Appliquées*, pp. 171–216, 1893.

[2] “7.3 POWELL'S ALGORITHM,” in *Algorithms for Minimization Without Derivatives*, in Dover Books on Mathematics., Dover Publications, 2002.

[3] “'fminbnd' documentation.” [Online]. Available: https://nl.mathworks.com/help/matlab/ref/fminbnd.html

A Aknowledgments

normal/position maps (Figure 3 Figure 5 Figure 6), as well as the 3D renders where created in Blender (https://www.blender.org/download/)
signed distance field from Figure 4 was generated in SDF maker by job talle (https://jobtalle.com/SDFMaker/)
All computations made in matlab

B Source code

Source code can be found on github (https://github.com/bionick7/ME46060_Optimization)