NASA/TM—2014-218317

Polyhedral Interpolation for Optimal Reaction Control System Jet Selection

Leon P. Gefert and Theodore W. Wright
Glenn Research Center, Cleveland, Ohio

June 2014

NASA/TM—2014-218317

Polyhedral Interpolation for Optimal Reaction Control System Jet Selection

Leon P. Gefert and Theodore W. Wright
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

June 2014

Contents

Polyhedral Interpolation for Optimal Reaction Control System Jet Selection

Leon P. Gefert and Theodore W. Wright
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Abstract

An efficient algorithm is described for interpolating optimal values for spacecraft Reaction Control System jet firing duty cycles. The algorithm uses the symmetrical geometry of the optimal solution to reduce the number of calculations and data storage requirements to a level that enables implementation on the small real time flight control systems used in spacecraft. The process minimizes acceleration direction errors, maximizes control authority, and minimizes fuel consumption.

1 Introduction

Rotations and small translations of space vehicles such as NASA's *Orion Crew Exploration Vehicle* (CEV) are performed by firing Reaction Control System (RCS) jets. These jets have a fixed thrust, so their duty cycle is varied between 0 and 100 percent to effect control. The jets have fixed locations and orientations on the spacecraft body, requiring combinations of jets firing with particular duty cycles to perform rotation about a particular axis or translation in a particular direction.

The orientations of the RCS jets are typically not aligned with spacecraft axes, making finding the correct combinations of jets to fire for particular maneuvers non-intuitive. The jet firing solutions are also dependent on the spacecraft's mass, center of mass, inertia, and the currently enabled set of redundant thrusters. Because of these factors, an optimizing algorithm is needed to determine efficient duty cycles. The optimal set of duty cycles for a particular rotation or translation maneuver is the set that maximizes control authority (acceleration) while minimizing fuel consumption and axis direction error.

However, it is not practical to run the RCS jet selection optimizer on the spacecraft's embedded control computers. Ideally, the embedded control computers should just perform a table look up of a pre-calculated set of optimal RCS jet duty cycles for a desired rotation or translation axis. The dependence of the optimal solution on the spacecraft's current mass properties and enabled thruster set prohibits the creation of tables large enough to accurately incorporate all these inputs.

This paper describes a new approach to looking up optimal RCS jet duty cycles that does not require huge data tables for accuracy, and also has a computational complexity (measured by the number of run time arithmetic operations) that is comparable with simple table lookups and interpolation. Unlike a typical table lookup, the solutions provided by the new approach are an exact match for the optimizer's results for pure rotation maneuvers. Results for pure translation are also good, but have less than perfect fuel efficiency. In addition, the translation algorithm also demonstrates a technique for combining a commanded pure rotation and pure translation into one set of duty cycles that performs the compound maneuver.

2 Optimal Solutions for Pure Rotation

The commanded input for performing a pure rotation with the RCS jets is a vector in 3 dimensions specifying the desired axis of rotation. The right hand rule is used to determine the rotation orientation about the axis. The output of the optimization is a set of eight duty cycles, one for each of the enabled RCS jets. The optimized values also depend on the spacecraft's mass, center of mass and inertia, which are mostly functions of the mission timeline (for example, the mass and inertia at the scheduled time of the trans-lunar insertion burn for a Moon mission will be very different than the mass and inertia at Earth re-entry on the way back from the Moon).

For a given axis input, the optimizer finds the best combination of RCS duty cycles that satisfies

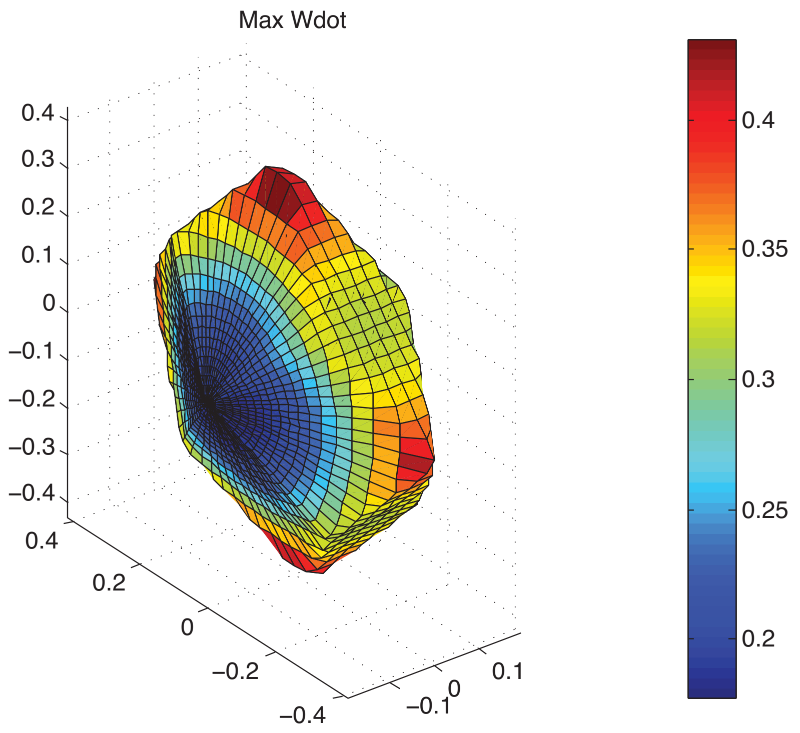

Figure 1: Optimized rotational acceleration magnitude as a function of rotation axis $[deg/s^2]$

three goals, in order of priority:

1. Accuracy - The direction of the resulting rotational acceleration must be purely around the commanded axis, with zero translation acceleration.
2. Control Authority - The magnitude of the resulting acceleration is maximized. If two duty cycle sets both cause rotation about the correct axis, the one with the greater acceleration is preferred.
3. Fuel Efficiency - The amount of fuel consumed (which is directly proportional to the sum of all duty cycles) is minimized. If two duty cycle sets both cause rotation about the correct axis with equal acceleration, the one with the smallest sum of duty cycles is preferred.

In spherical coordinates, the 3 dimensional rotation axis vector actually has only two degrees of freedom: inclination and azimuth angle (magnitude is not significant). Varying inclination and azimuth over all possible values (a 4π steradian sphere) will determine every possible input axis for the optimizer. The MATLAB LINPROG linear optimizer was used to generate optimal results for the Orion CEV 606d mass properties.

Figure 1 shows the result of plotting the magnitude of the acceleration due to the optimized rotation RCS duty cycles as a function of the input axis. This shape defines the overall rotational control authority (the maximum rotational acceleration that is possible about every axis).

Figure 2 shows the plot of all eight duty cycle values as a function of the input axis, with the values superimposed on the shape of the rotational acceleration magnitude.

Figure 3 shows the plot of just one of these RCS jet duty cycles as a function of the input axis, enlarged for clarity.

These optimal duty cycle values (as a function of the desired rotation axis) are the numbers that the flight control system needs to quickly determine for real time control.

3 Geometric Properties of the Optimal Rotation Solutions

A close examination of the geometry of the optimal rotation plots shows that the surface being plotted describes a shape called a *Rhombic Dodecahedron*: a symmetric shape with 12 sides, each of which consists of a kite shaped face. There are 14 vertices where 3 or 4 sides come together. Each of the kite shaped faces can be split into two coplanar triangular surfaces to allow for easy triangular interpolation on that surface.

The orientation and "stretching" of the rhombic dodecahedron varies based on the mass properties and enabled thruster set (thrusters come in pairs; only one thruster in each pair is enabled), but the basic symmetry and topology remains the same.

The duty cycle values of a particular thruster varies linearly across the faces of the rhombic dodecahedron. For a given thruster, typically three faces will be completely 0% (thruster never fires) at all locations, three faces will be 100% (thruster always on) at all locations, and six faces will have a linear variations between 0% and 100%.

The symmetric shape and linear variation across faces is the key to efficiently interpolating duty cycle values for any rotation axis from values given only at the 14 vertices.

The duty cycle values at the vertices are all either 0% or 100%, making it possible to calculate the vertex positions analytically. By visually inspecting the optimized results, an on/off thruster to vertex map is created. Multiplying this thruster map by the fixed thruster induced angular acceleration matrix determines the vertex locations. Once the vertex locations (which are the same for all eight thrusters) and vertex values (different for different thrusters) are known, values at the direction specified by any axis can be interpolated.

4 Rotational Rhombic Dodecahedron Interpolation

The algorithm for interpolating duty cycle values from data on the surface of a rhombic dodecahedron can be split into two parts to reduce calculation complexity:

1. Initialization - The initialization code is run whenever there are significant changes to the mass properties or enabled thruster set (e.g., before each major phase of flight). Initialization

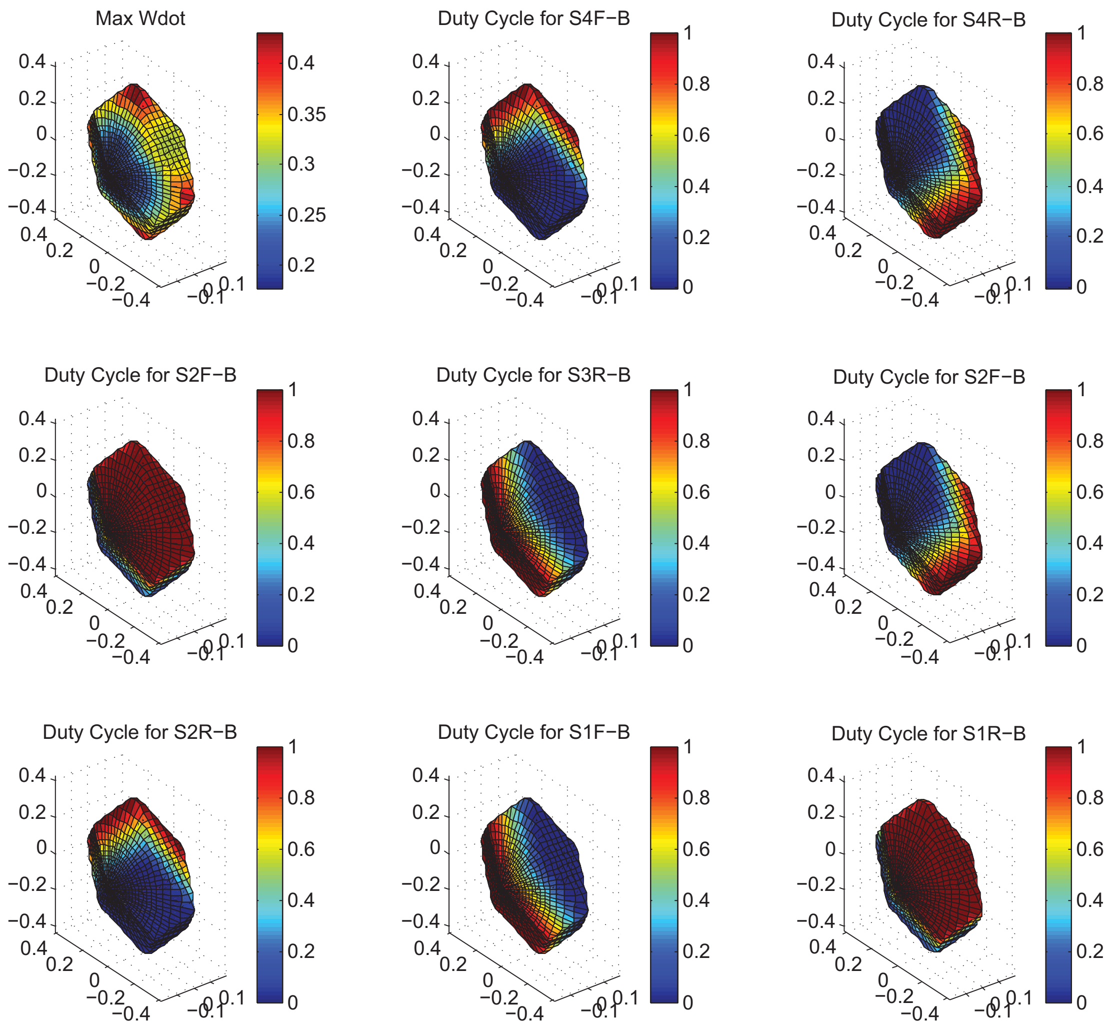

Figure 2: Pure rotation optimized duty cycle values as a function of the input axis [percent]

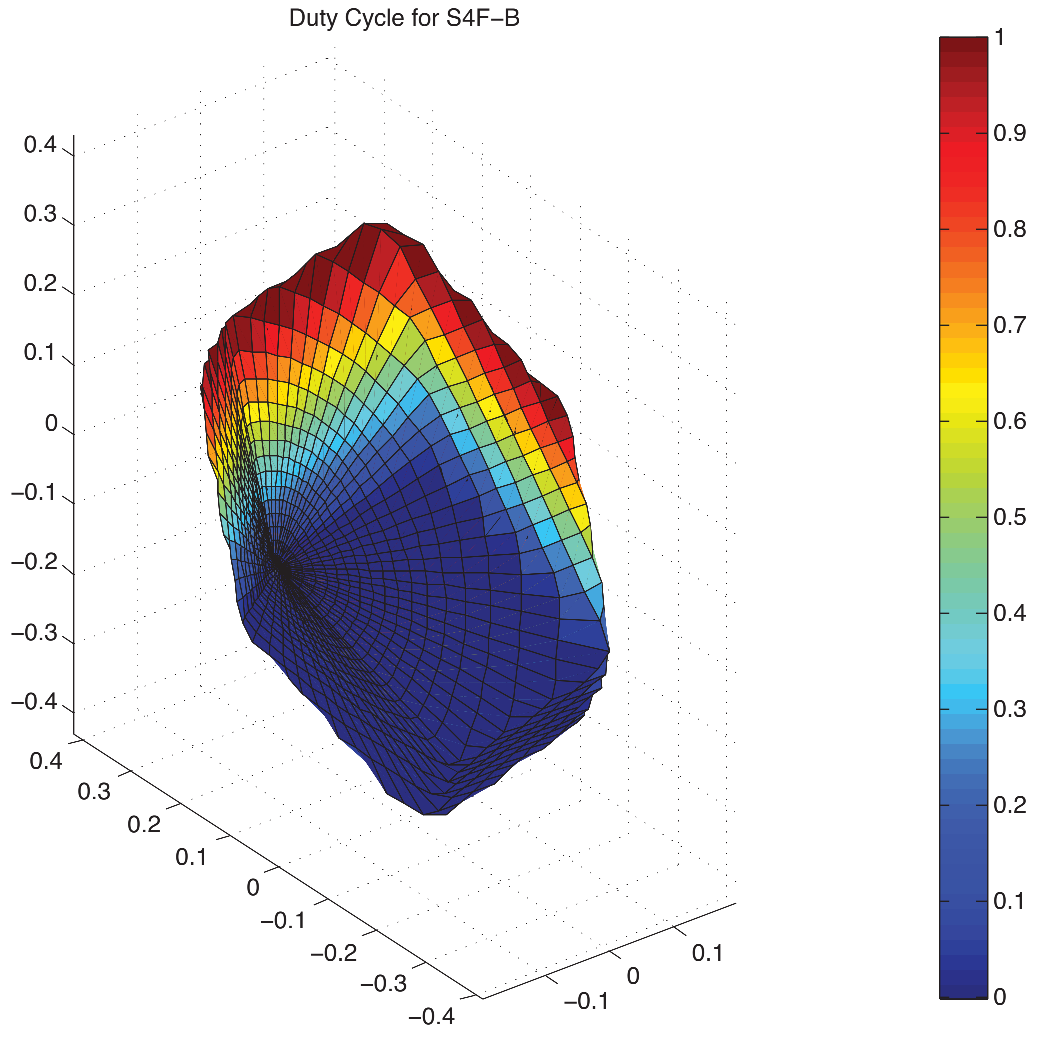

Figure 3: One pure rotation optimized RCS jet duty cycle as a function of the input axis [percent]

code calculates vertex positions, and computes plane fitting and barycentric interpolation coefficients for each triangular surface subdivision.

2. Control cycle - The control cycle code takes a rotation axis vector as input, finds the triangle on the polyhedron surface that intersects the rotation direction, and interpolates the duty cycles on that surface. Triangle selection and barycentric coordinate computation are common to all eight thrusters. Only the final interpolation calculation is unique for each thruster.

4.1 Initialization algorithm

The initialization portion of the rhombic dodecahedron interpolation algorithm collects all of the calculations that do not need to be repeated when only the direction axis input changes.

4.1.1 Rotational accelerations of current thruster set

The location vector $\bar{L}_i$ and force vector $\bar{F}_i$ of the 8 currently enabled thrusters are loaded. The location and force vector of each thruster are constant, but the set of enabled thrusters may change.

The mass m, center of mass $\bar{L}_{cm}$, and inertia $\bar{I}$ of the vehicle are loaded. These are relatively constant during a particular phase of flight.

The translational acceleration $\bar{a}_i$ due to each thruster i if it were firing at 100% duty cycle is found using equation 1.

$$\bar{a}_i = \bar{F}_i / m \tag{1}$$

The total rotational acceleration matrix $\bar{a}$ is the 3×8 matrix containing all the $\bar{a}_i$ vectors.

The torque τ_i due to each thruster is found using equation 2.

$$\tau_i = \left(\bar{L}_i - \bar{L}_{cm}\right) \times \bar{F}_i \tag{2}$$

The rotational acceleration $\bar{\dot{\omega}}_i$ due to each thruster is found using equation 3. Note that this requires inverting the inertia matrix.

$$\bar{\dot{\omega}}_i = \bar{I}^{-1} \cdot \tau_i \tag{3}$$

The total rotational acceleration matrix $\bar{\dot{\omega}}$ is the 3×8 matrix containing all the $\bar{\dot{\omega}}_i$ vectors.

4.1.2 Map accelerations to vertex positions

At each of the 14 vertices of the rhombic dodecahedron, a particular thruster always has a 0% or 100% contribution to the optimized results. The vertices are numbered, and each thruster's contribution at each vertex is found (by visual inspection) to be the constant array $DCsOfVerticiesR$, shown in equation 4.

$$DCsOfVerticiesR = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{4}$$

The location of each vertex $\bar{V}_i$ is found with equation 5.

$$\bar{V}_i = DCsOfVerticiesR \cdot \bar{\dot{\omega}}' \tag{5}$$

The vertices of the rhombic dodecahedron are connected to divide the surface into 24 triangles. These are described by their 3 vertex indices, which are found (by visual inspection) to be the constant array $trianglesR$, shown in equation 6 (this is one 24×3 matrix, split into two columns for easier reading).

The vertex vectors of a particular triangle i are found by extracting the ith row from $trianglesR$ to get the vertex indices for that triangle, and then using those indices to extract the corresponding vectors from the vertex matrix $\bar{V}$.

$$triangles R = \begin{bmatrix} 2 & 3 & 4 \\ 2 & 4 & 1 \\ 4 & 5 & 6 \\ 4 & 6 & 1 \\ 6 & 7 & 8 \\ 6 & 8 & 1 \\ 8 & 9 & 2 \\ 8 & 2 & 1 \\ 12 & 3 & 11 \\ 12 & 11 & 10 \\ 13 & 5 & 12 \\ 13 & 12 & 10 \end{bmatrix}, \begin{bmatrix} 14 & 7 & 13 \\ 14 & 13 & 10 \\ 11 & 9 & 14 \\ 11 & 14 & 10 \\ 12 & 5 & 4 \\ 12 & 4 & 3 \\ 13 & 7 & 6 \\ 13 & 6 & 5 \\ 14 & 9 & 8 \\ 14 & 8 & 7 \\ 11 & 3 & 2 \\ 11 & 2 & 9 \end{bmatrix} \tag{6}$$

4.1.3 Calculate plane coefficients for each triangle

To reduce repetitive calculations, it is helpful to find plane coefficients for each triangle in the initialization algorithm. For each triangle, the three vertices identified by vertex vectors $\bar{V}1, \bar{V}2, \bar{V}3$ determine a plane described by plane coefficients a, b, c, d where

$$ax + by + cz + d = 0 \tag{7}$$

The plane coefficients are calculated from the vertices using equations 8 and 9.

$$[a\ b\ c] = (\bar{V}2 - \bar{V}1) \times (\bar{V}3 - \bar{V}1) \tag{8}$$

$$d = \bar{V}1 \cdot [a\ b\ c] \tag{9}$$

4.1.4 Calculate barycentric coefficients for each triangle

Barycentric coefficients for each triangle can also be pre-calculated to reduce repetitive calculations. For each triangle, the three vertex vectors $\bar{V}1, \bar{V}2, \bar{V}3$ are used to compute the 9 barycentric coefficients $c1_x, c1_y, c1_z, c2_x, c2_y, c2_z, c3_x, c3_y, c3_z$ using equation 10.

$$\begin{bmatrix} c1_x & c1_y & c1_z \\ c2_x & c2_y & c2_z \\ c3_x & c3_y & c3_z \end{bmatrix} = \begin{bmatrix} V1_x & V1_y & V1_z \\ V2_x & V2_y & V2_z \\ V3_x & V3_y & V3_z \end{bmatrix}^{-1} \tag{10}$$

4.2 Control cycle algorithm

The control cycle portion of the rhombic dodecahedron interpolation algorithm contains only the calculations that are performed for new values of the direction axis input. All other inputs are assumed constant (until the next phase of flight, where new mass properties or enabled thruster sets trigger the run of the initialization routine).

4.2.1 Triangle selection using determinants

A rhombic dodecahedron can be split into triangles by the combination of six different planes that pass through the origin and five (carefully chosen, but constant) vertex pairs $\bar{V}1$, $\bar{V}2$, as shown in figure 4.

Given a direction vector $\bar{P}$ (or $[P_x\ P_y\ P_z]$) starting at the origin and a pair of vertices that define one of the splitting planes $[V1_x\ V1_y\ V1_z]$ and $[V2_x\ V2_y\ V2_z]$, the sign of a determinant will indicate which side of the plane includes the direction vector.

Equation 11 shows a determinant test for one of the planes. This expression will be true for all of the axis directions on one side of the plane, and false for all other axis directions.

$$\begin{vmatrix} V1_x & V2_x & P_x \\ V1_y & V2_y & P_y \\ V1_z & V2_z & P_z \end{vmatrix} \leq 0 \tag{11}$$

If this determinant test is performed with each of the six planes, the result is a vector of six Boolean values that are the same for every axis direction that passes through a particular triangle. For example, a result of [True, False, False, True, False, True] might mean the given direction vector passes through triangle 23.

A map of Boolean result vectors to triangle numbers can be created by calculating the Boolean results for the midpoint of each triangle. This map is constant, and can be implemented as a table lookup (by treating the Boolean array as a binary number, and using that number as a table index to return the corresponding triangle number).

The vertex pairs used in the tests are also constant during this calculation, so the only changing input is axis direction. Computing a single determinant requires 8 multiplications and 5 additions, and six determinants are required, so selecting the correct triangle given an axis direction requires a total of 54 multiplies and 30 additions.

4.2.2 Vector plane intersection

Once the correct triangle is found, the point of intersection between the direction vector and the triangular surface is computed. This is the target interpolation point $\bar{Q}$, or $[Q_x\ Q_y\ Q_z]$ as shown in

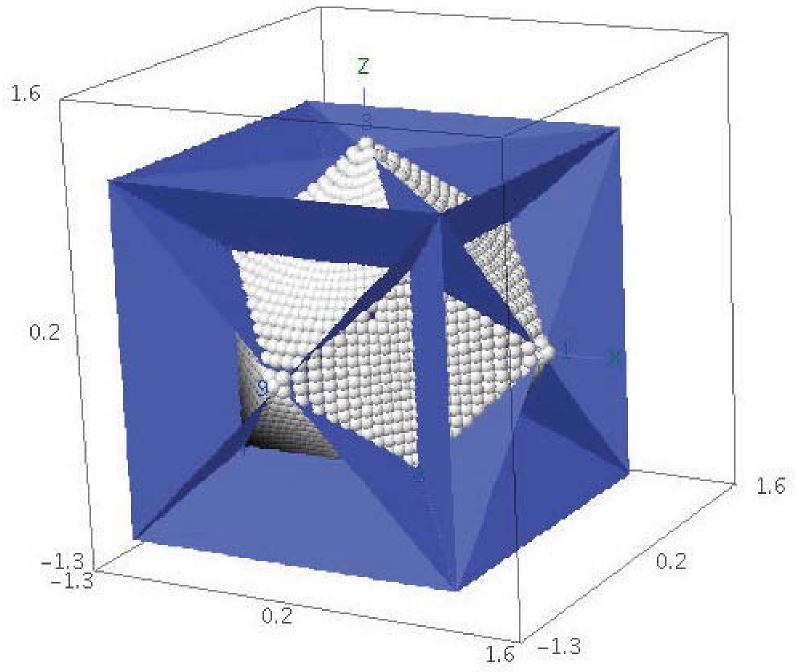

Figure 4: Six planes through the origin split a rhombic dodecahedron into triangles

figure 5.

The plane coefficients a, b, c, d for each triangle were found during the initialization algorithm. The plane coefficients are used to compute a scale factor t that determines $\bar{Q}$ from $\bar{P}$, as shown in equation 12 and equation 13.

$$t = \frac{-d}{a\,P_x + b\,P_y + c\,P_z} \tag{12}$$

$$\bar{Q} = t\,\bar{P} \tag{13}$$

Implementing equation 12 requires 3 multiplications, 2 additions and a division (The d coefficient is not used anywhere else and can be stored in its negated form). Equation 13 requires 3 multiplications. Finding the target interpolation point requires a total of 6 multiplications, 2 additions and 1 division operation.

4.2.3 Common barycentric coordinate calculation

Barycentric coordinates are the linear contribution of each vertex value to the interpolated value at the target interpolation point.

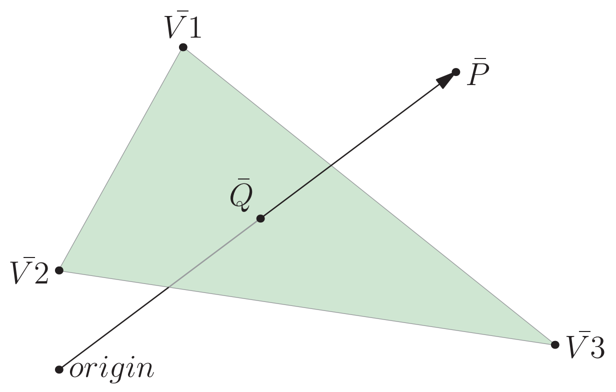

Figure 5: The target interpolation point is the intersection of the direction vector and the triangle

The barycentric coefficients $c1_x, c1_y, c1_z, c2_x, c2_y, c2_z, c3_x, c3_y, c3_z$ for each triangle were found during the initialization algorithm. The barycentric coefficients are used to compute the barycentric coordinates $b1, b2, b3$ as shown in equations 14 through 16.

$$b1 = [Q_x\ Q_y\ Q_z] \cdot [c1_x\ c2_x\ c3_x] \tag{14}$$
$$b2 = [Q_x\ Q_y\ Q_z] \cdot [c1_y\ c2_y\ c3_y] \tag{15}$$
$$b3 = [Q_x\ Q_y\ Q_z] \cdot [c1_z\ c2_z\ c3_z] \tag{16}$$

Each of these dot products requires 3 multiplications and 2 additions, so finding the barycentric coordinates requires a total of 9 multiplications and 6 additions.

4.2.4 Thruster unique triangular interpolation

The interpolated duty cycle value for one thruster is the dot product of the duty cycle values at the triangle vertices and the barycentric coordinates, as shown in equation 17.

$$interpolated_{DC} = [V1_{DC}\ V2_{DC}\ V3_{DC}] \cdot [b1\ b2\ b3] \tag{17}$$

This calculation is performed for each of the eight RCS jets (the barycentric coordinates are the same, but the duty cycle values at the triangle vertices vary). Each dot products requires 3 multiplications and 2 additions, so finding the interpolated duty cycles for all eight RCS jets requires a total of 24 multiplications and 16 additions.

5 Rotation Interpolation Accuracy

When the results from the rhombic dodecahedron interpolation are compared to the ideal results from the optimizer for all axis direction inputs, the worst case difference is less than $10^{-12}\%$. This is essentially a perfect match.

For pure rotations, rhombic dodecahedron interpolation is a non-iterative algorithm yielding optimized duty cycles without requiring an optimizer. The results have zero acceleration mis-alignment, maximum control authority, and minimum fuel consumption.

6 Rotation Interpolation Complexity

The total instruction counts for initializing the pure rotation rhombic dodecahedron interpolation algorithm are summarized in table 1.

Operation	Repetitions	Multiply	Add	Divide
Find rotational acceleration for current jets	1	195	110	9
Map rotational acceleration to vertices	1	336	294	0
Find plane coefficients for triangles	24	12	20	0
Find barycentric coefficients for triangles	24	27	14	9
Total		1467	1220	225

Table 1: Rotation instruction counts for initialization

The total instruction counts for one control cycle computation of the eight RCS jet optimal duty cycles for pure rotation are summarized in table 2.

Operation	Repetitions	Multiply	Add	Divide
Identify triangle (6 dets)	1	54	30	0
Point in plane	1	6	3	1
Barycentric coordinates	1	9	6	0
Interpolate	8	3	2	0
Total		93	55	1

Table 2: Rotation instruction counts for one control cycle

By pre-computing as many values as possible in the initialization code, the recurring control cycle part of the algorithm can be reduced to a relatively small number of instructions.

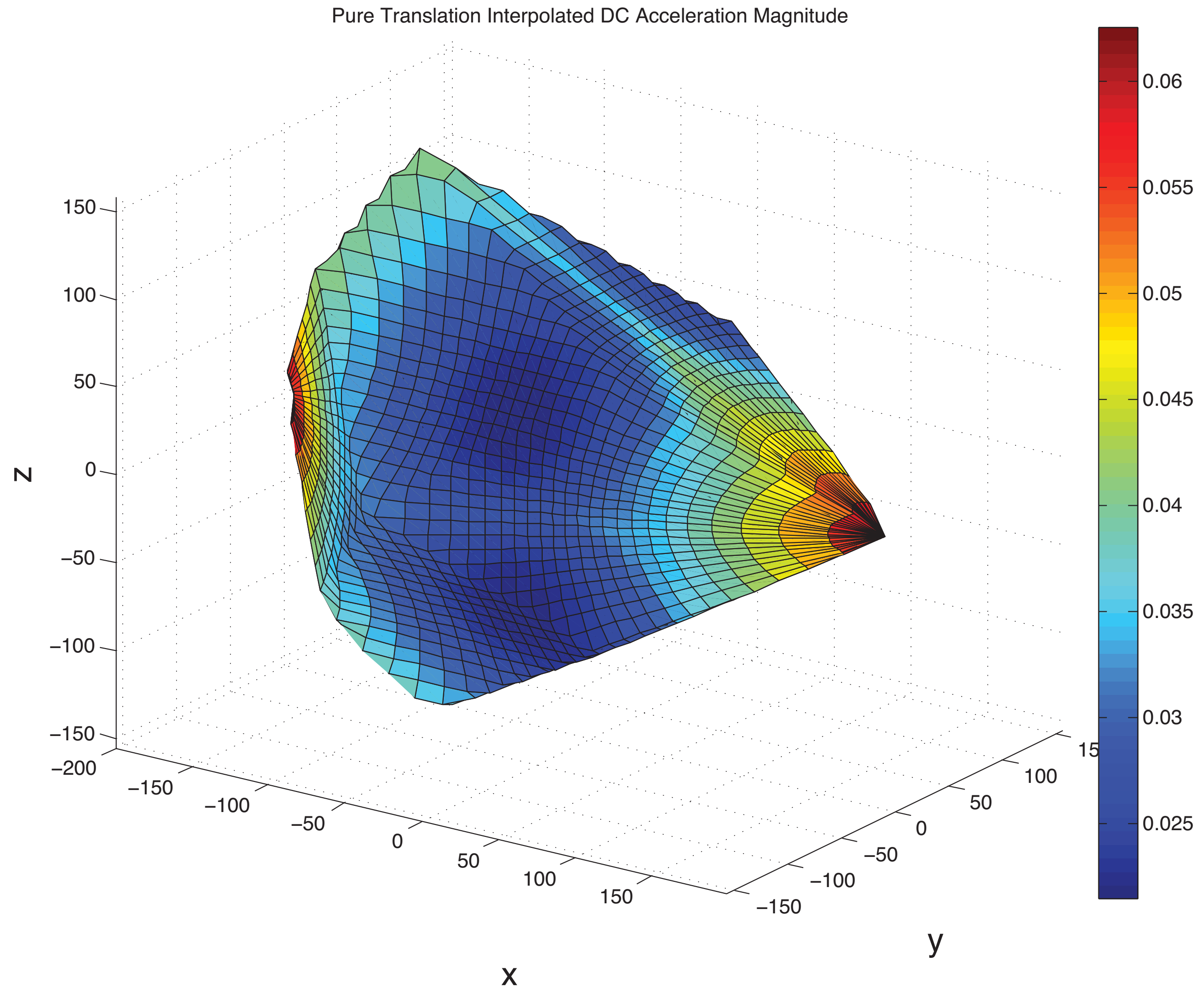

Figure 6: Optimized pure translation acceleration magnitude as a function of rotation axis $[ft/s^2]$

7 Optimal Solutions for Pure Translation

The commanded input for performing a pure translation with the RCS jets is a vector in 3 dimensions specifying the desired translation direction. Like the rotation case, only the inclination and azimuth of the vector are significant. The optimization criteria for pure translation require that the resulting acceleration is purely in the commanded direction, with zero rotational component. Then the magnitude of the resulting acceleration is maximized, and finally the amount of fuel consumed is minimized.

Figure 6 shows the optimized pure translation control authority (maximum linear acceleration as a function of commanded direction).

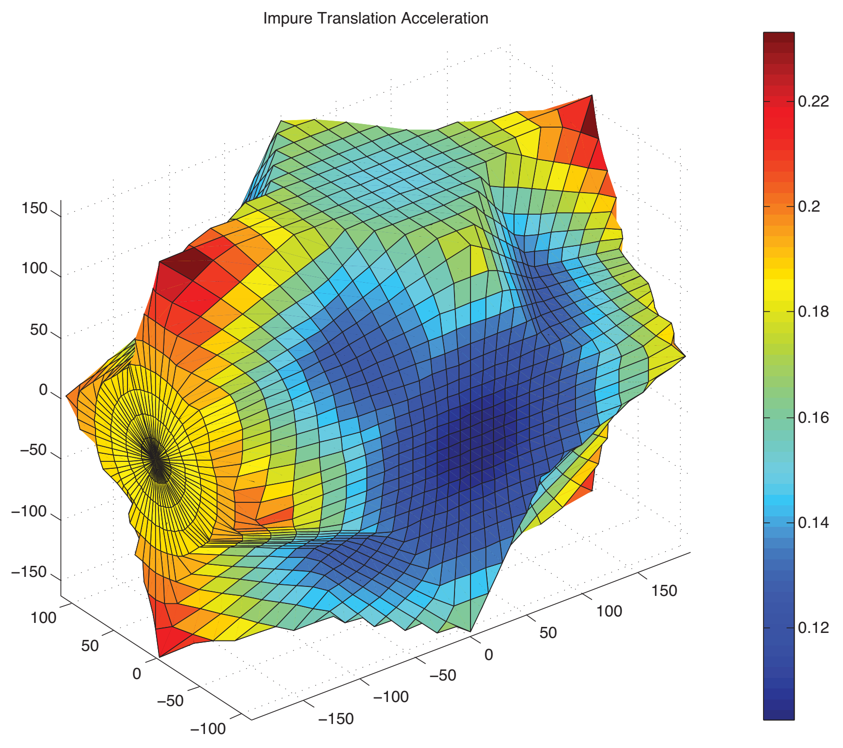

Figure 7: Optimized "impure" (with unintended rotation) translation control authority $[ft/s^2]$

Unfortunately, the shape of the pure translation control authority is too asymmetric to lend itself to decomposition by determinant tests, so it is not possible to use the same algorithm as the pure rotation case.

8 Approximating Pure Translation by Correcting Unintended Rotation

If the "zero rotational component" translation optimization criterion is relaxed, the shape of the optimal results allows a solution similar to the pure rotation algorithm (although further steps must be taken to compensate for the unwanted rotation). Figure 7 shows the optimized "impure" translation control authority (when unintended rotational accelerations are allowed).

Figure 8 shows the plot of all eight impure translation duty cycle values as a function of the input axis, with the values superimposed on the shape of the rotational acceleration magnitude. Figure 9 shows the plot of just one of these impure translation RCS jet duty cycles as a function of the input axis, enlarged for clarity.

The impure translation duty cycle shape is a compound polyhedron that combines the rhombic dodecahedron with a cuboctahedron. It has 96 sides and 50 vertices, but it shows the same geometric

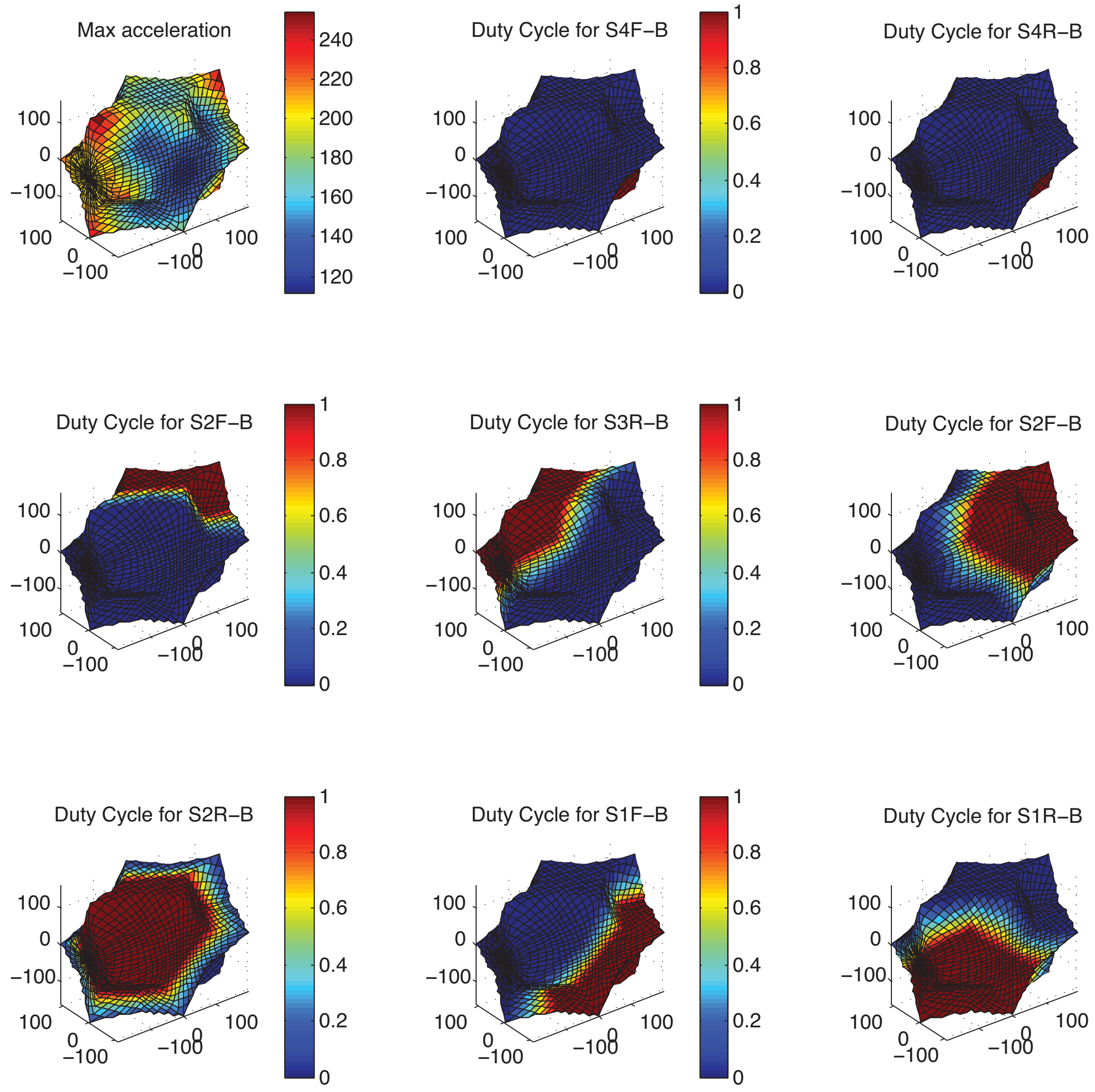

Figure 8: "Impure" translation optimized duty cycle values as a function of the input axis [percent]

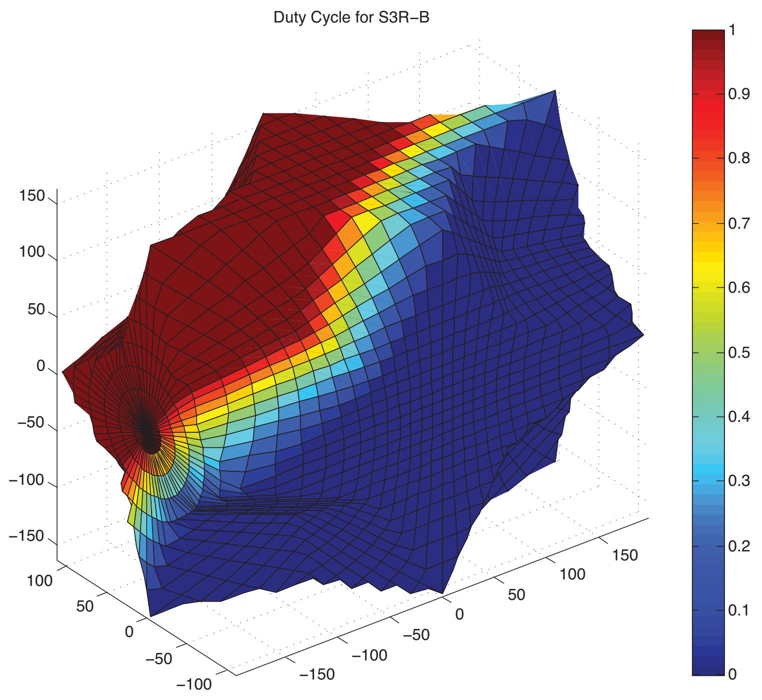

Figure 9: One "impure" translation optimized duty cycle as a function of the input axis [percent]

characteristics that make the efficient rotation algorithm possible:

- The shape is symmetric and can be decomposed with determinants
- Duty cycle values at the vertices are either 0% or 100%
- Duty cycle values vary linearly across the flat faces of the polyhedron

After the optimum impure duty cycle values are found, they can be used to quantify the unintended rotational acceleration. Duty cycle values for an offsetting rotational acceleration can then be generated using the rotation interpolation algorithm, and the impure translation and offseting rotation duty cycles can be combined into a single set of duty cycles that approximates pure translation (with some loss of optimality).

9 Translation Polyhedron Interpolation

Interpolating optimum duty cycle values on the surface of the impure translation polyhedron follows mostly the same steps as the rotation interpolation, so only the differences will be described.

9.1 Initialization algorithm

At each of the 50 vertices of the translation polyhedron, a particular thruster always has a 0% or 100% contribution to the optimized results. The vertices are numbered, and each thruster's contribution at each vertex is found (by visual inspection) to be the constant array $DCsOfVerticiesT$, shown in equation 18 (this is one 50×8 matrix, split into four columns for easier reading).

$$DCsOfVerticiesT = \begin{bmatrix} 10101010 \\ 00101110 \\ 00111100 \\ 10111000 \\ 11110000 \\ 11100010 \\ 11000011 \\ 10001011 \\ 00001111 \\ 01010101 \\ 00011101 \\ 01110100 \\ 11010001 \end{bmatrix}, \begin{bmatrix} 01000111 \\ 01010100 \\ 00010100 \\ 00010101 \\ 00000101 \\ 01000101 \\ 01000001 \\ 01010001 \\ 00101010 \\ 00101000 \\ 10101000 \\ 10100000 \\ 10100010 \end{bmatrix}, \begin{bmatrix} 10000010 \\ 10001010 \\ 00001010 \\ 00001100 \\ 00110000 \\ 11000000 \\ 00000011 \\ 01010000 \\ 10110000 \\ 01110000 \\ 11010000 \\ 11100000 \\ 00001110 \end{bmatrix}, \begin{bmatrix} 00001011 \\ 00000111 \\ 00001101 \\ 00111000 \\ 00101100 \\ 00011100 \\ 00110100 \\ 10000011 \\ 11000010 \\ 11000001 \\ 01000011 \end{bmatrix} \tag{18}$$

The location of each vertex $\bar{V}_i$ is found with equation 19.

$$\bar{V}_i = DCsOfVerticiesT \cdot \bar{a}' \tag{19}$$

The vertices of the translation polyhedron are connected to divide the surface into 96 triangles. These are described by their 3 vertex indices, which are found (by visual inspection) to be the constant array $trianglesT$, shown in equation 20 (this is one 96×3 matrix, split into four columns for easier reading).

$$triangles T = \begin{bmatrix} 1\ 22\ 23 \\ 1\ 23\ 24 \\ 1\ 24\ 25 \\ 1\ 25\ 26 \\ 1\ 26\ 27 \\ 1\ 27\ 28 \\ 1\ 28\ 29 \\ 1\ 29\ 22 \\ 10\ 15\ 16 \\ 10\ 16\ 17 \\ 10\ 17\ 18 \\ 10\ 18\ 19 \\ 10\ 19\ 20 \\ 10\ 20\ 21 \\ 10\ 21\ 34 \\ 10\ 34\ 15 \\ 9\ 39\ 29 \\ 9\ 29\ 40 \\ 9\ 40\ 33 \\ 9\ 33\ 41 \\ 9\ 41\ 18 \\ 9\ 18\ 42 \\ 9\ 42\ 30 \\ 9\ 30\ 39 \end{bmatrix}, \begin{bmatrix} 5\ 36\ 34 \\ 5\ 34\ 37 \\ 5\ 37\ 32 \\ 5\ 32\ 38 \\ 5\ 38\ 25 \\ 5\ 25\ 35 \\ 5\ 35\ 31 \\ 5\ 31\ 36 \\ 3\ 43\ 23 \\ 3\ 23\ 44 \\ 3\ 44\ 30 \\ 3\ 30\ 45 \\ 3\ 45\ 16 \\ 3\ 16\ 46 \\ 3\ 46\ 31 \\ 3\ 31\ 43 \\ 7\ 47\ 27 \\ 7\ 27\ 48 \\ 7\ 48\ 32 \\ 7\ 32\ 49 \\ 7\ 49\ 20 \\ 7\ 20\ 50 \\ 7\ 50\ 33 \\ 7\ 33\ 47 \end{bmatrix}, \begin{bmatrix} 11\ 17\ 16 \\ 11\ 16\ 45 \\ 11\ 45\ 30 \\ 11\ 30\ 42 \\ 11\ 42\ 18 \\ 11\ 18\ 17 \\ 12\ 46\ 16 \\ 12\ 16\ 15 \\ 12\ 15\ 34 \\ 12\ 34\ 36 \\ 12\ 36\ 31 \\ 12\ 31\ 46 \\ 4\ 43\ 31 \\ 4\ 31\ 35 \\ 4\ 35\ 25 \\ 4\ 25\ 24 \\ 4\ 24\ 23 \\ 4\ 23\ 43 \\ 2\ 44\ 23 \\ 2\ 23\ 22 \\ 2\ 22\ 29 \\ 2\ 29\ 39 \\ 2\ 39\ 30 \\ 2\ 30\ 44 \end{bmatrix}, \begin{bmatrix} 8\ 40\ 29 \\ 8\ 29\ 28 \\ 8\ 28\ 27 \\ 8\ 27\ 47 \\ 8\ 47\ 33 \\ 8\ 33\ 40 \\ 6\ 26\ 25 \\ 6\ 25\ 38 \\ 6\ 38\ 32 \\ 6\ 32\ 48 \\ 6\ 48\ 27 \\ 6\ 27\ 26 \\ 13\ 49\ 32 \\ 13\ 32\ 37 \\ 13\ 37\ 34 \\ 13\ 34\ 21 \\ 13\ 21\ 20 \\ 13\ 20\ 49 \\ 14\ 50\ 20 \\ 14\ 20\ 19 \\ 14\ 19\ 18 \\ 14\ 18\ 41 \\ 14\ 41\ 33 \\ 14\ 33\ 50 \end{bmatrix} \qquad (20)$$

9.2 Control cycle algorithm

The translation polyhedron can be split into triangle pairs by the combination of nine different planes that pass through the origin and eight (carefully chosen, but constant) vertex pairs $\bar{V}1, \bar{V}2$, as shown in figure 10. One final determinant using the common vertices of the triangle pair is sufficient to select between the two remaining triangles (to distinguish between the red and orange triangles in figure 10). In total, 10 determinants are required to select the triangular face on the surface of the polyhedron for a given translation direction.

9.3 Compute duty cycles to cancel unintended rotation

Once the duty cycles for impure translation DC_{it} are found, the unintended rotation it generates $\bar{\omega}_u$ can be found with equation 21.

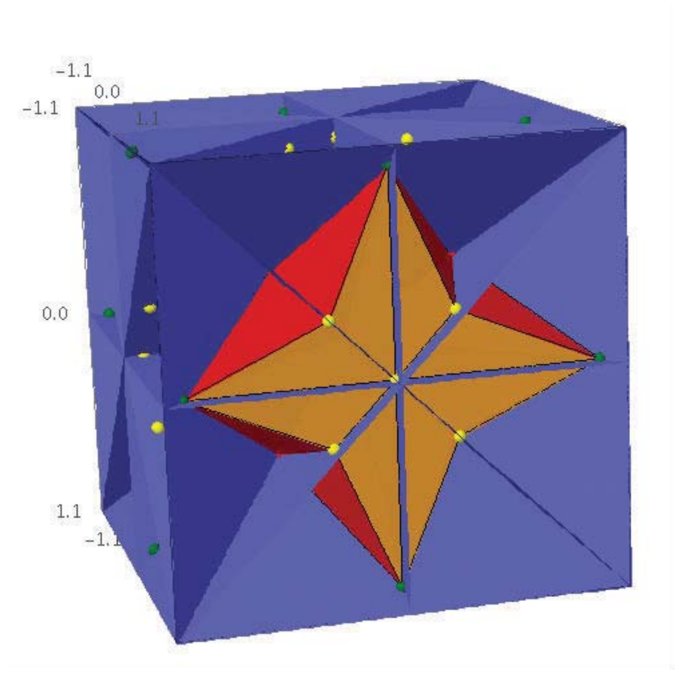

Figure 10: Nine planes through the origin split a translation polyhedron into triangle pairs

$$\bar{\dot{\omega}}_u = DC_{it} \cdot \bar{\dot{\omega}}' \tag{21}$$

Negating the unintended rotation $\bar{\dot{\omega}}_u$ yields the compensating rotation direction $\bar{P}_c$ needed for cancellation, which is used a the input for the rhombic dodecahedron interpolation algorithm to find the set of duty cycles for the rotating in the compensating direction: DC_c. Because the algorithm returns the duty cycles for maximum authority, these duty cycles will cause the maximum rotational acceleration $\bar{\dot{\omega}}_c$ about the compensating axis, as shown in equation 22.

$$\bar{\dot{\omega}}_c = DC_c \cdot \bar{\dot{\omega}}' \tag{22}$$

The compensating duty cycles need to be scaled down to the magnitude required to offset the unwanted rotation, resulting in DC_r as shown in equation 23.

$$DC_r = DC_c \cdot -\frac{1}{3}(\frac{\dot{\omega}_{ux}}{\dot{\omega}_{cx}} + \frac{\dot{\omega}_{uy}}{\dot{\omega}_{cz}} + \frac{\dot{\omega}_{uy}}{\dot{\omega}_{cz}}) \tag{23}$$

9.4 Combine translation duty cycles with rotational correction

To find the approximate pure translation duty cycles DC_t, the duty cycles for impure translation DC_{it} are combined with the duty cycles to correct unwanted rotation DC_r by simply adding them together and normalizing the result so that the maximum of the 8 duty cycles is 100%, as shown in equations 24 and 25.

$$DC_{sum} = DC_{it} + DC_r \tag{24}$$
$$DC_t = DC_{sum} / \max(DC_{sum}) \tag{25}$$

This "sum and normalize" technique can also be used to combine arbitrary translation and rotation commands into one set of duty cycles. It is equivalent to time multiplexing between the two commands.

10 Translation Interpolation Accuracy

As with the rotation interpolation algorithm, interpolating the impure translation duty cycles on the surface of the translation polyhedron results in a perfect match with ideal results from the optimizer for impure translation.

However, combining the impure translation duty cycles with an offsetting rotation correction to approximate pure translation duty cycles does not result in a perfect match with the pure translation results from the optimizer.

When all possible translation directions are considered, the mean difference in acceleration magnitude due to interpolated approximation verses optimized duty cycles for pure translation is 0.12%, with a worst case difference of 3.0%. The worst case direction error is less than 10^{-5} degrees. While interpolating values for pure acceleration using this technique is not perfect, it is fairly close to the optimum in terms of acceleration mis-alignment and control authority.

Pure translation interpolation is not as close to the optimum in terms of fuel efficiency. When all possible translation directions are considered, the mean difference in fuel efficiency (measured by the sum of the duty cycles) is 16% worse for the interpolated duty cycles, with a worst case increase in fuel consumption of 50%.

The difference between interpolated and optimal fuel efficiency for pure translation duty cycles as a function of translation direction is shown in figure 11. This plot shows that the fuel consumption differences reduce to zero along the axes, and are greatest at about 45 degrees off axis.

11 Translation Interpolation Complexity

The total instruction counts for one control cycle computation of the eight RCS jet optimal duty cycles for pure translation are summarized in table 3. This count includes computing both the impure translation and the offsetting rotation duty cycles, as well as combining them.

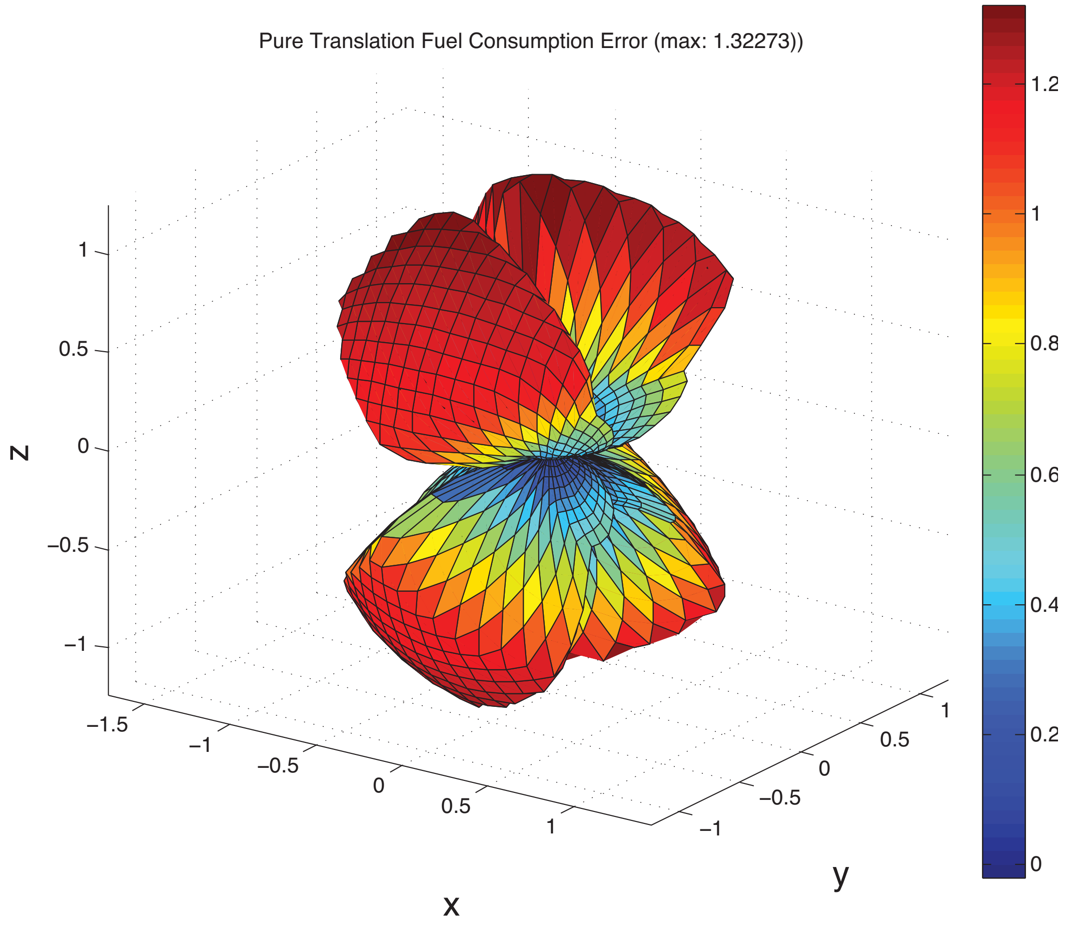

Figure 11: Pure translation fuel consumption difference between optimal and interpolated

Operation	Repetitions	Multiply	Add	Divide
Identify triangle (10 dets)	1	90	50	0
Point in plane	1	6	3	1
Barycentric coordinates	1	9	6	0
Interpolate	8	3	2	0
T acceleration calc	1	24	24	0
Rotation correction	1	93	55	1
R acceleration calc	1	24	21	0
Scaling and combining	1	8	8	9
Total		278	183	11

Table 3: Translation instruction counts for one control cycle

12 Summary

Optimal RCS jet selection is desirable on spacecraft because it minimizes acceleration direction errors, maximizes control authority, and minimizes fuel consumption. But on-orbit optimization calculations take too much time, and accurate table lookup of optimization results takes too much space. This paper describes a new efficient approach to interpolating optimization results that takes advantage of the geometric symmetries of the optimal solution.

For pure rotations, the new algorithm yields a perfect match with optimal results. It has zero acceleration mis-alignment, maximum control authority, minimum fuel consumption, and after initialization, requires a total of only 149 arithmetic operations to compute a result.

Pure translations are possible with the new algorithm, but require combining a translation with unintended rotation and a compensating rotation. In this case, the results are not a perfect match with the optimal, but are very close in terms of acceleration mis-alignment and control authority. However, there is an increase (16% mean, 50% max) in fuel consumption compared to the optimum.

The computational complexity of this new algorithm is fairly low, but it is non-intuitive due to its dependence on the symmetry of optimal solution rather than obvious physical properties of the spacecraft. The results presented here are based on the geometry of the Orion CEV spacecraft, and the applicability of the algorithm to other spacecraft is dependent on their geometry.

Traditional jet selection algorithm are very simple, but far from optimal. They sometimes have over 50 degrees of acceleration direction error, and they make no attempt to optimize fuel consumption. When compared to traditional jet selection, the added complexity of this new approach might be justified in situations where accuracy and efficiency are most important. For example, during proximity operation with other spacecraft, the ability to accelerate in the proper direction has increased significance. The ability to do accurate pure rotations and translations with predictable combinations of the two could be useful for more intuitive manual control of the spacecraft by astronauts. And finally, long duration station keeping operations increase the importance of optimized fuel consumption.