NASA/TM—2018-219913

Static Controls Performance Tool for Lunar Landers

Eliot D. Aretskin-Hariton, Calvin R. Robinson, and Drayton W. Munster
Glenn Research Center, Cleveland, Ohio

Mike R. Hannan
Marshall Space Flight Center, Huntsville, Alabama

An Erratum was added to this report March 2019.

October 2018

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Technical Report Server–Registered (NTRS Reg) and NASA Technical Report Server–Public (NTRS) thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers, but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., "quick-release" reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 757-864-6500

- Telephone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Program
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM—2018-219913

Static Controls Performance Tool for Lunar Landers

Eliot D. Aretskin-Hariton, Calvin R. Robinson, and Drayton W. Munster
Glenn Research Center, Cleveland, Ohio

Mike R. Hannan
Marshall Space Flight Center, Huntsville, Alabama

An Erratum was added to this report March 2019.

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

October 2018

Acknowledgments

This work was conducted as part of Lunar Cargo Transportation and Landing by Soft Touchdown (Lunar CATALYST) program in conjunction with Astrobotic Technology. NASA's Lunar CATALYST initiative is establishing multiple no-funds-exchanged Space Act Agreement (SAA) partnerships with U.S. private sector entities. The purpose of these SAAs is to encourage the development of robotic lunar landers that can be integrated with U.S. commercial launch capabilities to deliver payloads to the lunar surface. Editing support for this document was provided by Thomas F. Lavelle from Alcyon Technical Services.

Erratum

Issued March 2019 for

NASA/TM—2018-219913
Static Controls Performance Tool for Lunar Landers
Eliot D. Aretskin-Hariton, Calvin R. Robinson, and Drayton W. Munster
Glenn Research Center, Cleveland, Ohio
Mike R. Hannan
Marshall Space Flight Center, Huntsville, Alabama

October 2018

Page 10, Section 3, first paragraph: Remove the words "TBD Restore" from the first sentence so the sentence reads: After organizing the equality, inequality, and optimization matrices, the simplex method is then applied using the scipy.optimize.linprog() library in Python™.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

Available electronically at http://www.sti.nasa.gov/ and http://ntrs.nasa.gov/

Abstract

This paper presents a static analysis tool used to evaluate the controllability of Lunar landers. This was created as part of the NASA Lunar CATALYST program. This tool is capable of accepting typical design information such as location and direction of thrusters, maximum thruster forces, gravity vectors, and center of mass locations. The tool evaluates how far the center of gravity can move from its starting position while still maintaining control. This analysis is intended to support results produced by time domain simulations.

Nomenclature

Acronyms

ACS	attitude control system
CG	center of gravity
CoM	center of mass
Lunar CATALYST	Lunar Cargo Transportation and Landing by Soft Touchdown
PWM	pulse width modulation
SC	spacecraft

Symbols

$CG_{\Delta max}$	maximum allowable change in the location of the center of gravity
I_{sp}	Specific Impulse
M	number of attitude control system thrusters on the system
N	number of primary thrusters on the system

1 Introduction

This paper presents the design of a tool that can assess the controllability of lunar landers. This tool is compatible with Python^{TM}3.x and is designed to determine where the center of gravity (CG) of the spacecraft (SC) can be placed to ensure that the lander remains stable. This is performed by evaluating thruster locations, orientation, and thruster force through a static equilibrium approach where forces and moments are balanced. Solving this problem is accomplished by using linear programming and image processing techniques. The problem the tool was applied to was lunar landers, however, the tool can be changed to accept other landing scenarios. The tool can present a first cut at evaluating the controllability of a lander system during two phases of descent: vertical descent and hover (Figure 1). Both of these phases have different constraints on the static balance, which are inputs to the tool. The CG location is an output of this analysis because the CG can be altered by moving the location of installed equipment on the lander. This

analysis is meant to act as supporting information to dynamic, time domain analysis, which includes additional effects such as control algorithm design. Because static analysis is computationally inexpensive, it can also provide a useful component in parametric design for systems that are highly coupled. Inputs to the tool for static moment and force balancing and the construction of constraints will be described in Section 2. Calculations for linear programming and image processing techniques are described in Section 3. Simulation output information is detailed in Section 4. Additionally, a sample problem is provided to demonstrate the capability of the tool in Section 5. Lastly, conclusions are presented in Section 6.

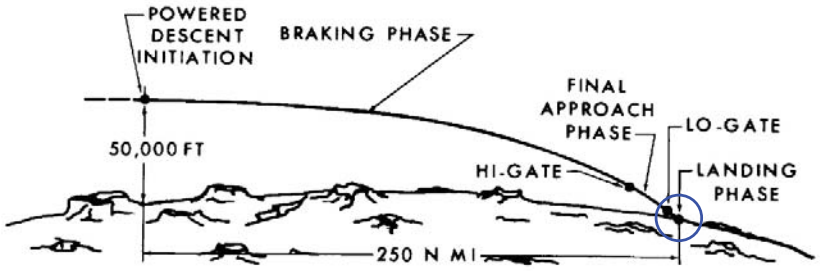


Figure 1: Key descent phases for lunar lander. Focus of tool is on last part of trajectory, vertical descent, and landing (blue circle). [1]

2 Simulation Inputs

This section discusses the types of inputs required for tool operation and the calculations that happen inside the tool. This is covered for two cases: hover and vertical descent. Hover has two subsections: analysis where the primary thrusters can also provide torque through off-pulsing (Section 2.2.2) and scenarios without off-pulsing (Section 2.2.3). The vertical descent phase (Section 2.2.4) does not have this distinction. The implementation of vertical descent for this study sets the primary thrusters to full power and prevents them from supplying control torque or off-pulsing.

The first case analyzed is hover with off-pulsing, meaning that the main engines can be used to supply control torque and balance the moments (Section 2.2.2). Hover without off-pulsing is where thrusters can be throttled higher and lower but they cannot be actuated individually for control torque creation (Section 2.2.3). In the vertical descent case, described in Section 2.2.4, the main thrusters are set to maximum power and cannot be used to balance the CG.

2.1 Variables and Coordinates

Required input parameters for the tool are listed in Table 1. Internal variables used by the tool and optional inputs are listed in Tables 2 and 3. The names of those parameters will be used throughout the rest of the text. The coordinate systems used in this paper include a SC local frame and a CG frame as shown in Figure 2. The SC frame is introduced because most SC will be designed without

knowing where the CG is located until later in the design process. Thus, the SC frame may be centered on the axis of symmetry, but not necessarily where the CG is located. The CG frame has the x-axis aligned with the gravity vector, which points along the negative x-axis. These frames are parallel to each other, but may have translational offset. An example of this translational offset is shown in Figure 2. There is no rotation between these coordinate systems, and thus this code is not currently designed for analysis at the start of powered descent, where the gravity vector and the CG frame are not aligned. If a future user wishes to inject a rotation between these coordinate systems, one additional input of a quaternion describing the rotation between the two coordinate frames would suffice.

The thrust vector input value for this analysis corresponds with the the direction of exhaust gases. This is 180°opposite of the force the thruster imparts on the SC. Inside the tool, the input value is rotated to become the force imparted on the SC.

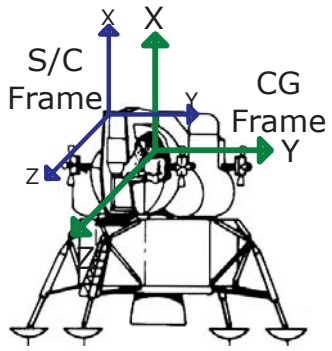


Figure 2: Example coordinate frame showing spacecraft (SC) frame and parallel center of gravity (CG) frame. Both frames aligned with gravitational attraction vector, which acts in negative x-axis. Frames are related with each other through translation but not rotation. [1]

Table 1: SIMULATION REQUIRED INPUTS

Variable name	Description	Size	Coordinates	Units
$ThrustPrimPos\_SC$	Position of primary thrusters	$N^a \times 3$	SC local	m
$ThrustPrimOri$	Orientation of primary thrusters	$N \times 3$	SC and CG	m
$ThrustACSPos\_SC$	Position of ACS thrusters	$M^b \times 3$	SC local	m
$ThrustACSOri$	Orientation of ACS thrusters	$M \times 3$	SC and CG	m
$GforcePos\_SC$	CoM[c] of the spacecraft (SC)	1×3	SC local	m
$MaxPrim$	Maximum thrust value of N primary thrusters	$N \times 1$	NA	N
$MaxACS$	Maximum thrust value of M ACS thrusters	$M \times 1$	NA	N
$SCMass$	Mass of the SC	1×1	NA	kg
$AccGrav$	Acceleration of the SC due to gravity	1×1	NA	m/s^2

[a]Number of primary thrusters.
[b]Number of attitude control system (ACS) thrusters.
[c]Center of mass of the SC.

Table 2: SIMULATION INTERMEDIATE VARIABLES

Variable name	Description	Size	Coordinates
$Thrusters\_SC$	Position and orientation of primary and ACS thrusters	$(N+M)\times 6$	SC local
$Thrusters\_CG$	Position and orientation of primary and ACS thrusters	$(N+M)\times 6$	CG frame
$ThrustPrim\_SC$	Position and orientation of primary thrusters	$N\times 6$	SC local
$ThrustPrim\_CG$	Position and orientation of primary thrusters	$N\times 6$	CG frame
$ThrustPrimPos\_SC$	Position of primary thrusters	$N\times 3$	SC local
$ThrustPrimPos\_CG$	Position of primary thrusters	$N\times 3$	CG frame
$ThrustPrimOri$	Orientation of primary thrusters	$N\times 3$	SC and CG
$ThrustACS\_SC$	Position and orientation of ACS thrusters	$M\times 6$	SC local
$ThrustACS\_CG$	Position and orientation of ACS thrusters	$M\times 6$	CG frame
$ThrustACSPos\_SC$	Position of ACS thrusters	$M\times 3$	SC local
$ThrustACSPos\_CG$	Position of ACS thrusters	$M\times 3$	CG frame
$ThrustACSOri$	Orientation of ACS thrusters	$M\times 3$	SC and CG
$Gforce\_SC$	CoM and orientation of the gravity force	1×6	SC local
$Gforce\_CG$	CoM and orientation of the gravity force	1×6	CG frame
$GforcePos\_SC$	CoM of the SC	1×3	SC local
$GforcePos\_CG$	CoM of the SC $=[0,0,0]$	1×3	CG frame
$GforceOri$	Orientation of the gravity force vector	1×3	NA
$ThrustMoment\_CG$	Unit vector moments of all thruster elements	$(N+M)\times 6$	CG frame
$CG_{\Delta max}$	Output: max. allowable change in CG location	1×1	NA

2.2 Hover With Off-Pulsing

Hover is the most restrictive of all the scenarios, and thus imposes the largest number of constraints on the system. In hover, all moments and forces must be zero. This keeps the craft in the same location without tilting or tumbling. However, the overall solution space on CG location will be the largest, due to the ability to use relatively large primary thrusters to create torque. Hover has the goal of minimizing primary thruster and ACS usage. The constraints used for hover are as follows:

$$\Sigma Forces = 0$$

$$\Sigma Moments = 0$$

Off-pulsing is a technique that can be used when an SC has several main engines. The thrust from each of the main engines is allowed to vary relative to each other. This allows the main engines to be used to contribute to torque balance as well as thrust. Allowing main engine differential thrusting allows for easier compensation when the thrusters are misaligned.

By using user input information on thruster location, direction, and craft weight (see: Table 1), off-pulsing is expressed as a series of linear equations, which can then be solved using the simplex method as described in Section 3.

2.2.1 Equality Constraints

The general equation describing the equality constraints is

$$A_{eq} \cdot x = B_{eq}$$

where A_{eq} describes the forces and moments acting on the SC, x is a vector of optimal thruster powers, and B_{eq} describes the constraints on the forces and moments, which is a function of the scenario. In the problems described in this paper, A_{eq} and B_{eq} are described as functions of the input values and we solve for x. Inputs to the function include a guess of where the CG is located. If there is no x value that solves the equation, then the given CG location is not feasible. The A_{eq} matrix is built from the individual thruster forces as follows:

$$Thrusters\_SC = \left[\begin{array}{c|c} ThrustACSPos\_SC & ThrustACSOri \\ \hline ThrustPrimPos\_SC & ThrustPrimOri \end{array}\right]$$

This information is translated into the CG frame. This affects the position part of the matrix, not the orientation section. The orientation section of the matrix is left as zeros:

$$Thrusters\_CG = Thrusters\_SC - \left[\begin{array}{c|c} GforcePos\_SC & 0 \end{array}\right]$$

The matrix that represents the moment vectors of each element in the thrust matrix is now constructed. This is represented as the cross product between the thruster location and the thruster orientation:*

$$\Sigma M_{CG} = Thrusters\_CG[:, 0:2] \times Thrusters\_CG[:, 3:5]$$

*The notation in this section uses standard Python™indexing nomenclature.

Table 3: SIMULATION OPERATIONAL INPUTS

Variable Name	Description	Size	Default
Resolution	The accuracy to which the minimum CG is determined	1×1	1×10^{-3} (meters)
GforceOri	Orientation of the gravity force unit vector (CG frame)	1×3	$[1, 0, 0]$
acs_thrust_%	Specific ACS thrust fractions to evaluate	$1 \times T^d$	[0.25]
missionPhase	Specifies mission phase, HOVER_NO_OFF_PULSE HOVER_OFF_PULSE, TERMINAL_DESCENT	String	TERMINAL_DESCENT

[d]Number of ACS thrust fractions to evaluate.

Next, create an expression for the sum of the forces in the three axes.

$$\Sigma F_{xyz} = Thrusters\_CG[:, 3:5]$$

These two elements combine to form the moment and force balance equation:

$$A_{eq} = \begin{bmatrix} \Sigma M_{CG} \\ \hline \Sigma F_{xyz} \end{bmatrix}$$

Then, form the B_{eq} matrix using information from the SC mass acceleration unit vector. For hover, the force and moment balance sum to zero. The gravitational acceleration on A_{eq} term is moved to the other side of the equation in this implementation B_{eq} , where

$$B_{eq} = \begin{bmatrix} 0_{3\times 1} \\ \hline -GforceOri \cdot SCMass \cdot AccGrav \end{bmatrix}$$

Substituting the A_{eq} and B_{eq} values this becomes:

$$\begin{bmatrix} \Sigma M_{CG} \\ \hline \Sigma F_{xyz} \end{bmatrix} \cdot x = \begin{bmatrix} 0_{3\times 1} \\ \hline -GforceOri \cdot SCMass \cdot AccGrav \end{bmatrix}$$

Where x is a column vector of size $[(M+N)\times 1]$ and represents the thrust level of the different thrusters. The number of ACS thrusters is given by M and the number of primary thrusters is indicated by N.

$$x = \begin{bmatrix} ACSForce \\ \hline PrimaryForce \end{bmatrix}$$

2.2.2 Inequality Constraints

Inequality constraints are built using maximum thruster values as follows:

$$A_{ineq} \cdot x <= B_{ineq}$$

where A_{ineq} is the identity matrix,

$$A_{ineq} = \mathbb{I}_{M+N\times M+N}$$

and

$$B_{ineq} = \begin{bmatrix} MaxACS \\ \hline MaxPrim \end{bmatrix}$$

2.2.3 Optimization Function

The generic optimization function seeks to find x such that $f = C \cdot x$ is minimized, where C is a vector of weights and x is column vector of thruster power. In the hover case, it is desirable to minimize fuel used to maintain hover. The Python™ `scipy.optimize.linprog()` library will be used to solve this problem.

This library requires that the input functions must be linear. Additionally, the initial starting point (e.g., the first point evaluated by the function) must be in the valid solution space. Thus, the initial CG guess provided must be in the solution space or the library will fail to solve.

$$FuelConsumed = \sum_{i=1}^{i=M+N} C_i \cdot x_i$$

where N is the number of primary thrusters and M is the number of ACS thrusters. The row vector C represents the coefficients of how the x values are weighted. In the simple case where all thrusters are equally efficient, the coefficient values are all ones to indicate that the thrusters have equal weight. This is because it does not matter if thruster A is used more than thruster B, just that overall all thrusters are being used as little as possible. For this case, C becomes

$$C = [1_1, 1_2 ... 1_{M+N}]$$

If the specific impulse (I_{SP}) of the thruster is supplied, this can be used as the coefficients for the C vector. This will give preference to using thrusters with a high I_{SP}. While the thrusters that will be used may change, the maximum allowable change in the CG $CG_{\Delta max}$ will stay the same. This is because $CG_{\Delta max}$ typically forces many of the thrusters to fire at 100 percent duty cycle. A system that includes I_{SP} as the weighting function would have the following C vector:

$$C = \left[\ \frac{1}{I_{SP,1}}, \frac{1}{I_{SP,2}} ..., \frac{1}{I_{SP,M+N}} \ \right]$$

The tool does not currently implement the I_{SP} option.

2.3 Hover Without Off-Pulsing

Additional equality constraints are created if the primary thrusters must all fire together at the same thrust level. The terminology used to describe this is called without off-pulsing, where off-pulsing indicates that the primary engines can fire independently at different thrust levels. Thus, without off-pulsing means that the primary engines cannot fire independently. The controllability suffers as a result since the main engines can no longer be used for torque balance. This causes the allowable CG offset to shrink when compared to cases with off-pulsing.

2.3.1 Equality Constraints

Implementation of systems without off-pulsing requires constructing additional equality constraints and combining with equality constraints already created in Section 2.2.2. This is performed by constructing a series of equations where each main thruster is set as equal to the other main thrusters.

$$\begin{bmatrix} T_{1main} = T_{2main} \\ T_{2main} = T_{3main} \\ \vdots \\ T_{N-1main} = T_{Nmain} \end{bmatrix}$$

None of the ACS thrusters have these types of constraints so they remain unchanged. This can be structured as an equality constraint, which will be added to A_{eq}:

$$Tmain_{eq} = \left[\begin{array}{c|cccccc} & 1 & -1 & 0 & 0 & \cdots & 0 \\ & 0 & 1 & -1 & 0 & \cdots & 0 \\ 0_{M-1\times N} & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ & 0 & 0 & 0 & \cdots & 1 & -1 \end{array}\right]$$

The corresponding B_{eq} constraints are all zero. These $Tmain_{eq}$ are added to A_{eq} and B_{eq} from Section 2.2.2 to create the whole equality matrix for this case:

$$\left[\begin{array}{c} ThrustMoment\_CG \\ \hline ThrustOri \\ \hline Tmain_{eq} \end{array}\right] \cdot x = \left[\begin{array}{c} 0_{3\times 1} \\ \hline -GforceOri \cdot SCMass \cdot AccGrav \\ \hline 0_{M+N} \end{array}\right]$$

2.3.2 Inequality Constraints

Inequality constraints are the same as those for hover with off-pulsing as described in Section 2.2.2.

2.3.3 Optimization Function

The optimization function is the same as those for hover with off-pulsing as described in Section 2.2.2. This is because of the continued desire to minimize fuel use.

2.4 Vertical Descent

Vertical descent is a less restrictive case as compared to hover. There is an allowed force imbalance along the x-axis (vertical axis). This enables the craft to accelerate (slow down) along this axis. This case is marked by the need to use primary thrusters at 100 percent thrust. This removes off-pulsing as a mechanism of control because no solutions are allowed where any primary thrusters are allowed to operate at less than 100 percent. Equations governing the x-axis are the only ones modified to remove constraints. To enable vertical descent, some of the equality constraints will be adjusted as compared to those in Section 2.2.2. Vertical descent has a goal of maximizing primary thruster usage while minimizing ACS thruster usage. Therefore, the cost function must also be modified. The constraints used for vertical descent are as follows:

$$\Sigma Forces_x \neq 0$$

$$\Sigma Forces_{y,z} = 0$$

$$\Sigma Moments = 0$$

2.4.1 Equality Constraints

The basic equality constraints are the same as those described in Section 2.2.2. The constraints for the hover case are adjusted slightly by removing the requirement of balancing forces in the x-axis (vertical axis). This requires removing a single line in the A_{eq} matrix and the corresponding row in the B_{eq} matrix.

$$A_{eq} = \left[\frac{\Sigma M_{CG}}{\Sigma F_{yz}} \right]$$

Additional equality constraints must be added to set the primary thrusters to their maximum thrust value:

$$A_{eq} = \left[\frac{\frac{\Sigma M_{CG}}{\Sigma F_{yz}}}{\left[\; 0_{M \times N} \; \middle| \; \mathbb{I}_{M \times M} \; \right]} \right]$$

Unlike the without off-pulsing case, additional equality constraints are not needed to set the primary thrusters equal to each other. This is complemented by additions in the B_{eq} matrix to set these primary thrusters to their maximum thrust:

$$B_{eq} = \left[\frac{0_{5 \times 1}}{MAXPrim} \right]$$

The completed equality constraints are then inserted in the familiar form of

$$A_{eq} \cdot x = B_{eq}$$

where x again represents the thrust level of the different thrusters.

2.4.2 Inequality Constraints

Inequality constraints are modified from the baseline presented in Section 2.2.2. The constraints related to the primary thrusters are removed because those thrusters have a fixed thrust value. The modification to the matrices is shown below:

$$A_{ineq} \cdot x <= B_{ineq}$$

$$A_{ineq} = \left[\; \mathbb{I}_{N \times N} \; \middle| \; 0_{N \times M} \; \right]$$

$$B_{ineq} = \left[\frac{MaxACS}{0_{M \times 1}} \right]$$

2.4.3 Optimization Function

The optimization function is the last item to be modified for this case. The dependence on the primary thrusters must be removed from the baseline case presented in section ref:hwop. These thrust values can no longer be optimized because they are being set to 100 percent. This is performed by summing only the values

for M ACS thrusters where previously $M + N$ ACS and primary thrusters were included.

$$FuelConsumed = \sum_{i=1}^{i=M} C_i \cdot x_i$$

This change is implemented by modifying the C vector to ensure that primary thrust values are not included:

$$C = \left[\; 1_{1\times M}, 0_{1\times N} \; \right]$$

3 Calculations

TBD Restore After organizing the equality, inequality, and optimization matrices, the simplex method is then applied using the `scipy.optimize.linprog()` library in Python™. The simplex method, as described by Dantzig [2], is a linear programming method that can be used with problems that can be described by a series of linear equations. This method solves the equality and inequality constraints and determines x and c if there is a viable solution. Where x is the thrust produced by each thruster and c is the fuel consumption.[†] An array of viable solutions is compiled for the final analysis.

To reduce the computational costs of searching the solution space, a boundary-following algorithm was implemented. This significantly reduced computation time. To illustrate the computational savings, consider the two examples in Figure 3. Both of these images trace out the boundary to the same resolution. However, the full factorial search requires 2,601 samples while the boundary following algorithm only required 323. Higher resolutions can show even further gains. For the example above, a 101×101 grid that would require 10,201 samples can be traced with only 642 samples.

The algorithm implemented here was created by Pavlidis [3]. While there are more advanced boundary-following algorithms available, the Pavlidis algorithm was chosen for its simplicity and performance on the shape of regions expected for this application.

The algorithm can be described by imagining a walker traversing the boundary, as shown in Figure 4. Darker squares correspond to examined pixels, green corresponds to the inside of the shape, and red corresponds to the outside. The walker starts on an interior point and (without loss of generality) facing north. The walker examines P1, P2, and P3, in that order, until they find a valid point (Figure 4a). If P1 is a valid point, the walker steps forward and to the left (resulting in a rotation to the left, Figure 4b). If P1 is invalid and P2 is valid, the walker steps forward (Figure 4c). If none of these points are valid, the walker rotates 90° to the right and tries again (Figure 4d). If both P1 and P2 are invalid and P3 is valid, the

[†]A note for bang-bang thrusters: since this is not a time domain analysis, the results of required percent of thruster firing can be thought of as the equivalent pulse width modulation (PWM) that happens when the thruster is pulsed over time. Thus a 10-percent thruster value indicates that the thruster should be fired in such a way that it achieves 10 percent of the maximum thrust which corresponds to a 10-percent duty cycle for a bang-bang thruster. While this analysis is not perfect, it does speak to the general capability of the thrusters to counteract force and moment constraints.

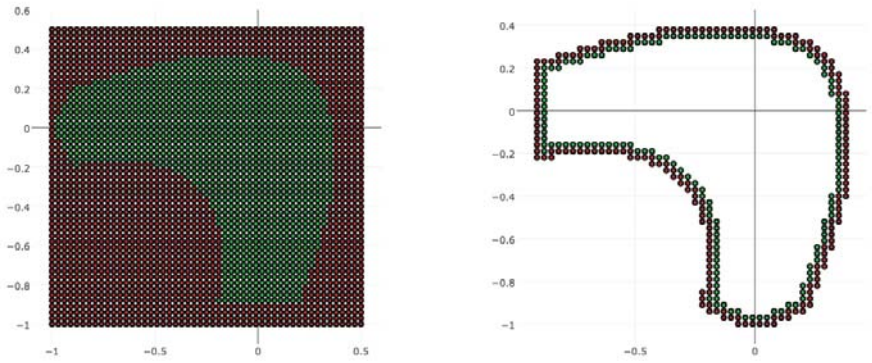


Figure 3: Simplex method evaluation. (a) Full-factorial search. (b) Border following. Each dot represents simplex method evaluation. Green dots are inside valid solution space (solution does converge). Red dots are outside solution space (solution does not converge).

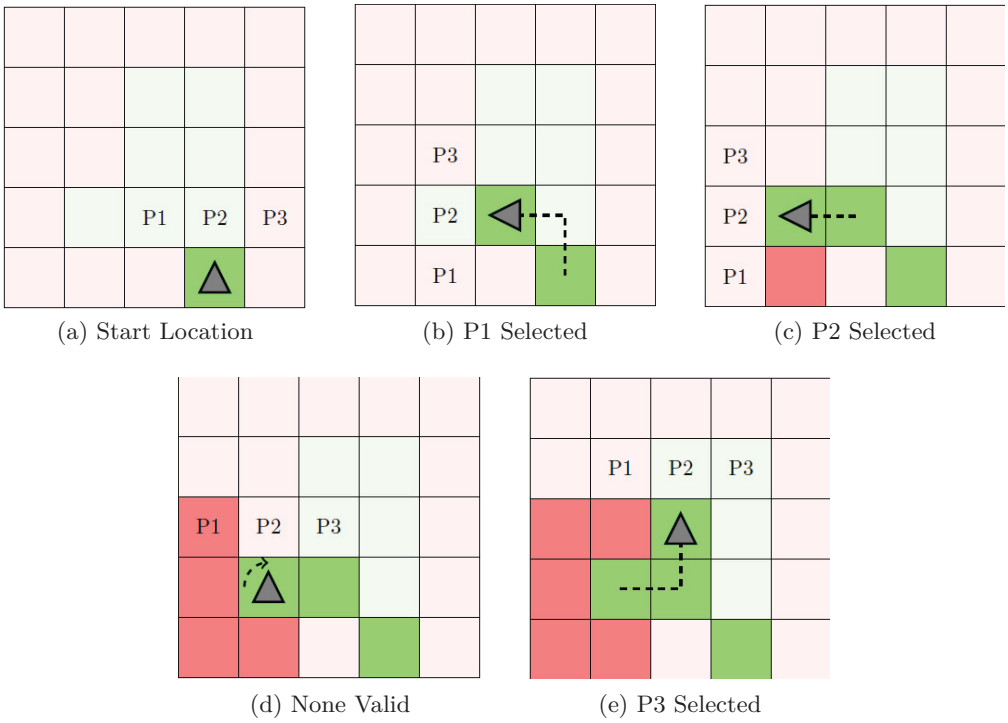
walker steps to the right and up (Figure 4e). The algorithm terminates when the starting location is reached. Different stopping criteria may be required for more complicated regions, but the regions involved in this application should not require more complicated criteria.

The values returned by the boundary-following algorithm are viable points along the boundary only. No information regarding the valid points inside of this boundary are returned. Thus, even though it may have taken several function evaluations to initially travel from the CG (the starting point which must be a valid point) to the boundary, those points are not reported because they are inside of the solution space.

4 Simulation Outputs

Summarizing the impact of the results from Figure 3 is done by determining the smallest amount that the CG can be moved before the system becomes unstable. This will help determine the requirements or bounds on acceptable CG locations during lander construction and design. Equipment on the lander may need to be moved to properly balance the craft. The geometric interpretation is achieved by fitting a circle into the shape of the solution space as shown in Figure 5. The diameter of that circle is $CG_{\Delta max}$. It is the equivalent of the largest circle that can be created when starting at the initial CG location and only contains stable conditions. Some changes in lander configuration may change the stable area, but do not change $CG_{\Delta max}$. This is because the radius of the circle is based on a constraint of the closest failure to the CG. In order to increase $CG_{\Delta max}$, the active constraint must be directly affected by changes in thruster location or initial CG placement.

A second graph can then be created by using this information by varying the



(a) Start Location (b) P1 Selected (c) P2 Selected

(d) None Valid (e) P3 Selected

Figure 4: First steps in Pavlidis' algorithm. (a) Start location. (b) P1 selected. (c) P2 selected. (d) None valid. (e) P3 selected.

$MaxACS$ from 0 to 100 percent and running the entire simulation (simplex method, boundary following, and $CG_{\Delta max}$ evaluation) for every value of maxACS. Then, maxACS is graphed against $CG_{\Delta max}$. An example of this is shown in Figure 6. This allows the user to assess how much ACS control authority they are willing to allocate to CG offset, with the rest of the control authority allowing for maneuvers.

A typical metric is to allocate 10 percent of ACS thruster capability to CG offset canceling. This is the default metric that is reported by the program. ACS thrusters at 10 percent is also the first value computed by the program when it is run in computation only mode (no human input). The target ACS fraction is set by the optional `targetACS` parameter.

5 Example Problem

The following hover with off-pulsing example shows the general capabilities of the tool. The craft mass is configured as 300 kg and the gravitation acceleration body is Luna. The initial CG is set to $[0, 0, 0]$. Table 4 describes thruster location, orientation, type, and maximum thrust. There are five primary thrusters situated around the craft pointed down toward Luna (negative x-axis). There are 12 ACS thrusters that are also placed symmetrically around the coordinate system origin. The maximum thrust of the primary and ACS thrusters is 450 and 20 N, respectively.

The resulting $CG_{\Delta max}$ for the 10-percent ACS thruster level is shown in Figure 7.

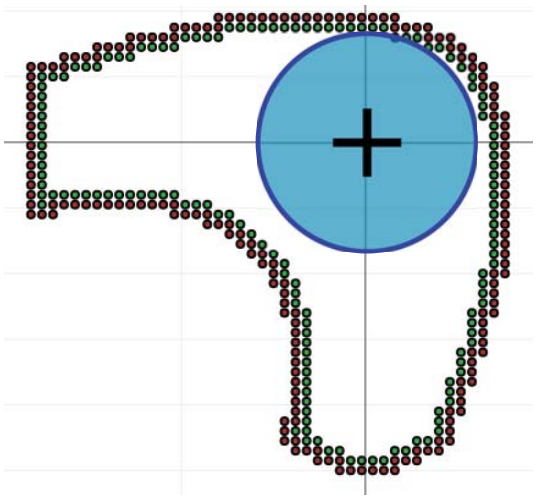


Figure 5: Sample results of lander with complicated stability region geometry. Red dots indicate instable solutions. Green dots indicate stable solutions. Predicted SC CG is at center of blue circle. Blue circle indicates how far CG can move before system becomes unstable. Maximum radius ($CG_{\Delta max}$) only includes feasible points dictates size of circle.

The 10-percent ACS thrust level reserves the remaining 90 percent of the ACS thrust for maneuvering. Maneuvering enables the craft to translate and rotate in a controlled fashion. The solution space shows that offsets of as much as 0.26 m (26 cm) are allowable if 10 percent of ACS power is allocated to CG offset compensation. Figure 6 shows the summary information of graphs from the entire solution space. The 10-percent value of 0.26 m shown in Figure 6 corresponds to the radius of the circle shown in Figure 7. This large margin for CG movement is typical of hover with off-pulsing cases.

The same example calculated for hover without off-pulsing will yield a $CG_{\Delta max}$ of 0.009 m (0.9 cm) for 10-percent ACS usage. This indicates that the majority of the CG offset capability is driven by the ability to use primary thrusters.

6 Conclusion

A tool was developed in Python^{TM}and shown to be capable of performing static controls analysis for lander-type spacecraft (SC). This analysis gives an indication on the allowable movement of the center of gravity (CG) of the SC given the current CG location, and the thrusters available for torque and force balance operations. This analysis can act as a first cut at control analysis and should be supported with time domain controls analysis. The tool covers several types of landing cases, including hover and vertical descent. Hover with off-pulsing is the most permissive case and will result in the largest allowable CG offset. Vertical descent is the least permissive case and will result in the smallest allowable CG offset. Typical inputs for the scenarios are described as well as the equations used in computing the output. This tool can be included in a larger parametric optimization program as a computationally inexpensive first cut at controllability analysis.

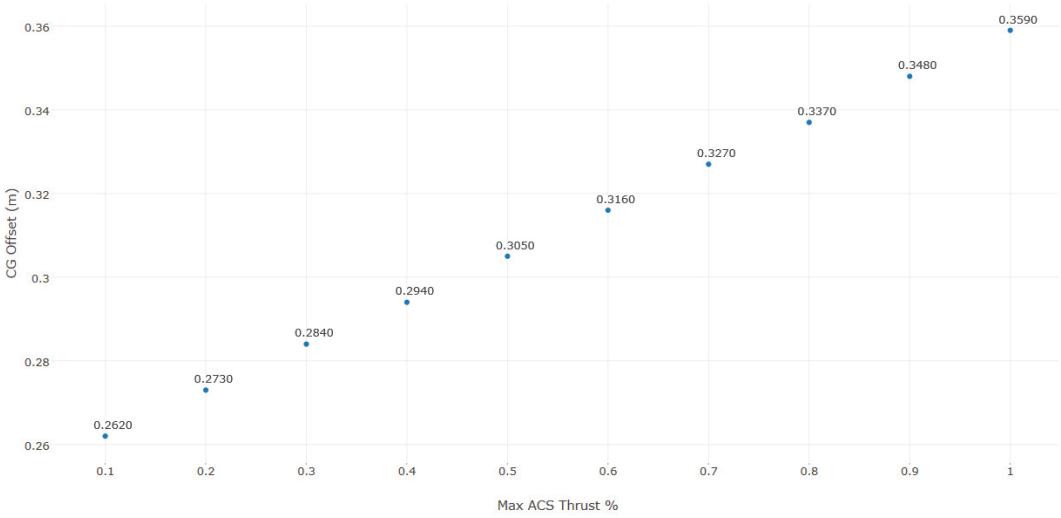


Figure 6: Final tool output example for hover with off-pulsing. Full solution space for maximum ACS percentage allocated versus maximum allowable center of gravity offset from initial guess. Each blue dot corresponds to $CG_{\Delta max}$. Breakout details of 10 percent case are shown in Figure 7.

References

1. NASA, "Proceedings of the Apollo Lunar Landing Mission Symposium," NASA-TM-X-58006, June 1966.

2. Dantzig, G. B., "Origins of the Simplex Method," Technical Report SOL 87-5, Stanford, CA, May 1987.

3. Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, MD, 1982.

Table 4: EXAMPLE THRUSTER CONFIGURATION (APPROXIMATE VALUES). VALUES ARE INPUT TO THE TOOL AS A .csv TABLE

$ThrustPrimPos\_SC$, (m)			$ThrustPrimOri$			Type	Max. thrust
[x]	[y]	[z]	[x]	[y]	[z]	[1 = primary]	(N)
1.3	0.1	0.7	0	1	1	0	20
1.3	-0.1	-0.7	0	-1	-1	0	20
0.1	0.7	-0.1	0	1	-1	0	20
0.1	-0.7	0.1	0	-1	1	0	20
1.3	0.1	-0.7	0	1	-1	0	20
1.3	0.1	0.7	0	-1	1	0	20
0.1	-0.7	-0.1	0	-1	-1	0	20
0.1	0.7	0.1	0	1	1	0	20
1.3	0	-0.7	1	0	0	0	20
1.3	0	0.7	1	0	0	0	20
0.05	0.7	0	-1	0	0	0	20
0.05	-0.7	0	-1	0	0	0	20
0	0	0	-1	0	0	1	450
0	0.25	0.25	-1	0	0	1	450
0	0.25	-0.25	-1	0	0	1	450
0	-0.25	0.25	-1	0	0	1	450
0	-0.25	-0.25	-1	0	0	1	450

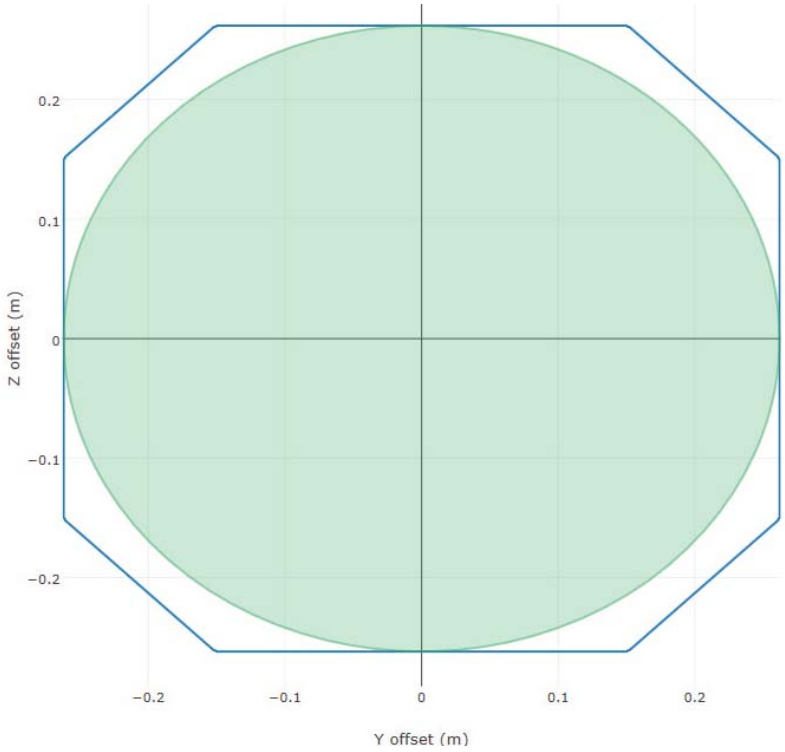


Figure 7: Tool output: hover with off-pulsing solution space for 10 percent ACS thrust. Blue lines represent points in valid area of solution space found by boundary-following algorithm. Green circle represents $CG_{\Delta max}$.