Guidance Optimization for Mars Pinpoint Landing with Optimal Trigger and Re-optimization

Tiago Amaral
tiagomedeirosamaral@ist.utl.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

May 2019

Abstract

Precise landings will be required for future Mars missions and will be crucial to missions landing on challenging places such as craters or valleys. Also, in future manned missions, it is important to land close to cargo that was previously sent. In this work, a guidance function is implemented and tested for precise landings on Mars, based on a method developed by NASA. In this method, convex optimization is used to create fuel-optimal trajectories that guide the spacecraft from a point A to land on a point B. In this work, it was developed a method for finding the optimal position to trigger the powered descent, a loop for re-optimizing the trajectory and a simple method to take the drag effect into account. A Mars landing scenario was simulated with 1500 different initial states and atmospheric conditions. The algorithm allowed landing with half-meter precision and to save on average 30 % of fuel in comparison with an enhanced Apollo-like guidance function. It was determined with simulations that the introduced methods are viable and robust. The developed function has shown to be able of, autonomously and in real-time, generate fuel-optimal trajectories to land on target.
Keywords: Pinpoint Landing, Powered Descent Guidance, Optimal Trigger Point, Convex Optimization, Re-optimization

1. Introduction

The next years will certainly be exciting for anyone interested in Mars exploration. With recent advances in rocket technology, such as the re-usability of launch vehicles and the cost reduction associated, not only Mars manned missions are closer than ever but also new ambitious missions are surging. One important requirement for these future missions will be the ability to perform a pinpoint landing, which is defined as autonomously guiding a spacecraft to land in a chosen target with an accuracy smaller than 100 meters. This is crucial for missions that require landing in challenging places, such as craters or valleys [5]. Also, in future manned missions, it is important to land close to cargo that was previously sent [5, 6].

In a typical mission to the Mars surface, the spacecraft is first injected into a transfer orbit from Earth to Mars. When arriving at Mars, the process of Entry, Descent and Landing (EDL) starts, in order to decelerate the spacecraft from hypersonic velocities to near zero and soft land. The entry phase of EDL starts typically at an altitude of 125 km and ends at the deployment of the parachute [13]. In this phase, the spacecraft enters the Martian atmosphere and aerobrakes to supersonic speeds using a heat-shield or an aeroshell [2]. To initiate the descent phase, a single or a combination of parachutes are deployed, typically 10 km or less above Mars' surface [13]. Due to the thin Martian atmosphere, the terminal velocity achieved with a parachute is significantly higher than in Earth's atmosphere, typically around 50 m/s [13]. Therefore, Mars EDL usually finishes with a powered descent phase where rocket thrusters are used to slow down the spacecraft to the desired velocity at touchdown.

Due to the high unpredictability of Mars' atmosphere, the spacecraft can accumulate large position and velocity errors during the atmospheric entry phase. Additionally, because of the winds in the parachute descent phase, the state of the spacecraft (position and velocity) gets even more dispersed and it can not be accurately estimated [2]. These position dispersions can be up to 8-10 km with respect to a prescribed target at the start of the powered descent phase [2, 12]. Thus, previous missions to the red planet didn't have good landing precision. Mars exploration rovers had a landing accuracy of 35 km and later the Mars Science Laboratory, with rover Curiosity, had an accuracy of 10 km [13]. However, high precision landing was not a requirement in these past missions.

Therefore, to achieve pinpoint landing in future missions, new onboard techniques must be used to correct the errors accumulated during entry and descent [3]. This calls for a powered descent guidance (PDG) algorithm capable of calculating a fuel-optimal trajectory that guides a spacecraft from an original state to the target. This trajectory must be calculated in real-time, because the location of the spacecraft at the start of the powered descent phase cannot be correctly predicted. Thus, it is essential to use an algorithm that is fast and with guaranteed convergence to the global optimum [3]. This lead to the interest of using convex optimization, more specifically second-order cone programming (SOCP), which is a subclass of convex optimization.

Convex optimization has proven to be a practical way to solve the pinpoint landing problem [5]. NASA developed an algorithm for pinpoint landing on Mars using convex optimization [3]. It was proven that the non-convex pinpoint landing problem can be reformulated as a convex optimization problem, more specifically as a second-order cone problem. Additionally, and more important, it was shown that this problem can be losslessly convexified [1]. That is, the problem can be converted into a convex optimization problem whose optimal solution is the same as the original nonconvex problem. In this convex programming approach, a fuel-optimal trajectory and thrust profile are created to autonomously guide the spacecraft from an initial state (position and velocity) to the target and perform a soft landing. It is worth mentioning that convex optimization is a key enabling factor of SpaceX's Falcon 9 landings. SpaceX generates real-time flight code using convex optimization [5]. Lars Blackmore, one of the co-inventors of the convex programming approach to pinpoint landing, is now the principal rocket landing engineer at SpaceX.

The objective of this work is to develop and implement a guidance algorithm for precise landing on Mars, based on NASA's approach to pinpoint landing [2–4]. In this framework, some features were introduced to improve robustness and overall feasibility of the original guidance function. First, a search method is introduced to determine the optimal position to trigger the powered descent. Second, a periodic loop for re-optimizing the trajectory is included. This improves robustness since it is possible to efficiently correct dispersions that may occur because of drag or unexpected winds. Third, a procedure to take the drag effect into consideration is introduced.

The guidance algorithm was validated with simulations of pinpoint landings on Mars. The landing is tested for 1500 initial states, with dispersions of 5 km and different atmospheric conditions, based on possible initial states from an ESA mission [9]. The quality and feasibility of the improved guidance function was tested, based on optimality, robustness and computation time.

2. Background

Convex optimization was selected because it guarantees convergence to the global optimum, which is essential for a real-time onboard computation of an optimal trajectory [3]. Additionally, it has been shown that the set of feasible initial states, i.e, the initial states for which it is possible to land on target, is more than twice larger when convex optimization is used than other traditional approaches [11].

Convex optimization is a subclass of mathematical optimization, or simply optimization, where the cost function must be convex, all equality constraints must be linear and the inequality constraints have to define a convex set. In layman terms, convex optimization can be summarized as finding the optimal value of a convex function in a domain without holes or indentations [10].

This work is based on the convex programming approach for Mars landing, developed by NASA [2–4]. It was proven that the pinpoint landing can be reformulated as a convex optimization problem, more specifically as a second-order cone problem. Additionally, it was shown that the problem is lossless, i.e, that it can be converted into a convex optimization problem whose optimal solution is also the optimal solution of the original nonconvex problem [1].

2.1. Pinpoint Landing Problem

The constrained powered descent problem consists in finding the optimal time-of-flight and thrust-vector profile that minimizes the fuel consumption required to guide a spacecraft between specified initial and final conditions, subject to an array of constraints.

In the original literature, several steps were performed to eliminate any type of nonconvexity in the original formulation of the problem. The main ones were a change of variables and introduction of slack variables to relax a non-convex constraint. After the relaxations and the time discretization of the problem, the final SOCP problem that solves the pinpoint landing for a given t_f can be represented by Problem 1.

Problem 1

$$\min_{t_f, u(\cdot), \sigma(\cdot)} \quad -z_N \tag{1}$$

$$\text{subject to} \quad \|u_k\| \leq \sigma_k \tag{2}$$

$$r_{k+1} = r_k + \frac{\Delta t}{2}(\dot{r}_k + \dot{r}_{k+1}) + \frac{\Delta t^2}{12}(u_{k+1} - u_k) \tag{3}$$

$$\dot{r}_{k+1} = \dot{r}_k + \frac{\Delta t}{2}(u_k + u_{k+1}) + g\Delta t \tag{4}$$

$$z_{k+1} = z_k - \frac{\alpha \Delta t}{2}(\sigma_k + \sigma_{k+1}) \tag{5}$$

$$\rho_1 e^{-z_{0_k}} \left[1 - [z_k - z_{0_k}] + \frac{[z_k - z_{0_k}]^2}{2}\right] \leq \sigma_k \tag{6}$$

$$\sigma_k \leq \rho_2 e^{-z_{0_k}} \left[1 - [z_k - z_{0_k}]\right] \tag{7}$$

$$\ln(m_{wet} - \alpha \rho_2 k \Delta t) \leq z_k \tag{8}$$

$$z_k \leq \ln(m_{wet} - \alpha \rho_1 k \Delta t) \tag{9}$$

$$\|A_i x + b_i\|_2 \leq {c_i}^T x + d_i, \quad i = 1, ..., p \tag{10}$$

$$z(0) = \ln(m_{wet}), \quad r(0) = r_0, \quad \dot{r}(0) = \dot{r}_0 \tag{11}$$

$$r(N) = \dot{r}(N) = 0 \tag{12}$$

2.2. Solving the Pinpoint Landing Problem

The solver ECOS was selected to solve the pinpoint landing problem because it is fast, open-source and meets all the requirements for onboard implementation. For small and medium-size problems, ECOS is faster than most existing second-order cone programming solvers [7].

In order to use ECOS solver, the problem had to be reformulated into a specific formulation required by ECOS such as

$$\begin{aligned} \text{minimize} \quad & \omega^T y, && \text{(13a)} \\ \text{subject to} \quad & Ay = B, && \text{(13b)} \\ & Gy \leq h. && \text{(13c)} \end{aligned}$$

The vector y contains the optimization variables and the vector ω^T represents the cost of the problem. Matrices A and B represent the equality constraints and the matrices pair G and h contains the inequality constraints. The problem was rewritten in a way that all optimization variables are contained in the column vector y and all constraints are represented by the matrices A, B, G and h.

A comparison of results with the original literature of the convex optimization method for pinpoint landings was made. It was seen that the results of this work are in agreement with the original literature, with differences smaller than 1 %, which confirmed that the method to solve the pinpoint landing using ECOS is working properly.

3. Approach to pinpoint landing

In this chapter, the approach to pinpoint landing and the design of the guidance function will be discussed. The guidance function will be based on the convex optimization method for pinpoint landing, but some improvements will be made to address some shortcomings and enhance the overall performance and feasibility of the method. The guidance function of this work will be referred to as the improved guidance algorithm, while the NASA algorithm will be referred to as the original guidance algorithm.

4. Improved Guidance Algorithm

The original guidance algorithm uses convex optimization to calculate a fuel-optimal trajectory that guides a spacecraft from a point A to land in a point B. However, the original guidance algorithm is not capable of deciding where to trigger the engines, i.e, the point A is where the powered trajectory begins and it has to be provided as input. One contribution of this work is to introduce a method to autonomously determine the optimal position to trigger the powered descent.

Additionally, in the original guidance algorithm, the spacecraft follows only one thrust profile, i.e, after the powered descent begins, there are no more trajectory calculations. In the improved guidance function, however, multiple trajectories are calculated during the powered descent, to re-optimize and correct errors efficiently. Another contribution is a method to take the drag effect into consideration, which is not taken in the original guidance algorithm.

With these improvements, the guidance function is able to autonomously guide a spacecraft from the end of the atmospheric braking phase through the landing. Every input required by the improved guidance function can be read during the flight, which makes this guidance function completely autonomous. The required inputs are the position, velocity, acceleration and mass of the spacecraft. The guidance function outputs are the thrust commands for the propulsion systems of the spacecraft and also trajectory and velocity profiles for the control systems. The improved guidance function is designed to be robust, reliable and able to always generate a feasible trajectory.

4.1. Optimal Trigger Location

In the original method for pinpoint landing [3], what the algorithm does is to find the fuel-optimal trajectory that gets the spacecraft from a given initial position to the target. However, there is no procedure to determine the optimal position to trigger the powered descent, i.e, the initial position from which to start the trajectory. This initial position from which the trajectory is calculated is crucial. If the trajectory starts too late, the spacecraft may not slow down enough and crash. If the trajectory starts too soon, it will not be a fuel-optimal trajectory because the spacecraft will spend more time fighting gravity.

A simple but effective approach is developed here. After heuristically analyzing the triggering of the powered descent on different points of the descending flight, it was observed that it is not efficient to start the powered trajectory at the beginning of the descent phase. In unpowered flight, the required fuel to reach target decreases over time and has a

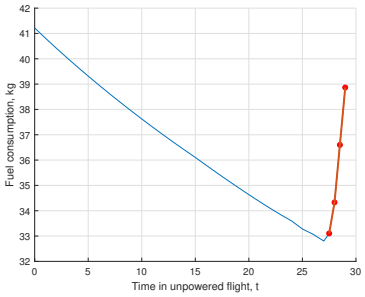

Figure 1: Fuel consumption that it would take to land starting the maneuver at different times of the unpowered descent flight. Time represents the duration of unpowered flight. Here, $t = 0$ marks the beginning of the study, in a position known to be far from the optimal trigger position. The red section represent infeasible solutions.

minimum near the point in flight where it is not possible anymore to slow down and land the spacecraft (see the profile in Figure 1). This behaviour was observed in all of the 1500 simulations.

With this consistent pattern, the approach to find the optimal triggering point can be to make in-flight periodic comparisons of the fuel consumption that it would take if the trigger happened in the current position or in a predicted future state. If the consumption of the future state is lower than the current state, the unpowered flight continues and so does the comparison. Whenever the fuel consumption of the future state is higher than the current state or if it is infeasible to start the maneuver in the future state, then it is optimal to trigger the powered descent in the current position.

In this approach, some factors have to be taken into consideration. First, the interval of time in which a state is propagated ahead has to be larger than the time required to compute the trajectory starting at that state. Second, the error between the propagated and actual states (position and velocity) must be small enough. Large errors in propagation can lead to bad trigger decisions or saturation of the spacecraft's control systems, which can lead to failure.

Since there is no knowledge of the time-of-flight of the powered trajectory, the first trajectory calculations need to perform a line-search to find the optimal time-of-flight and its associated trajectory. After that, since there is already a good estimate of the optimal t_f, it is possible to predict the optimal t_f of the next state based on the t_f of the previous state. For example, if the next point is five seconds ahead, its optimal t_f will be around the previous t_f minus five seconds. This was seen to be true and therefore only a few trajectories calculations are enough to find the optimal t_f and the optimal trajectory of the next state.

To select the period of the optimal trigger search, it is important to analyze the number of trajectories that must be computed on each search loop and also the time required to compute each trajectory.

4.2. Real-time Trajectory Computation Time

In a Mars mission, the landing guidance algorithm is supposed to run on a radiation-hardened flight processor. Thus, it is important to characterize the execution time of the algorithm in a flight processor. Since flight processors are way more complex than desktop processors (different instruction sets and memory access speeds), the run-time on a desktop computer cannot simply be scaled to a flight processor by the ratio of clock speeds [8].

It was not possible in this work to use a radiation-hardened processor and analyze the real execution time of the algorithm. However, the authors of the convex programming approach to pinpoint landing implemented their code in a NASA flight processor and their results can be used as a reference point for the real-time execution times [8]. The customized solver used in [8] has similar performance to ECOS and there is no need for a solver customization. Since the execution time of the algorithm of this work-frame is affected only by the speed of ECOS (the code prior to the call of ECOS runs instantaneously), the performance of both algorithms will be similar. In addition, the trajectory example used in [8] is similar to the ones of this work. Thus, it is considered acceptable to use the results from [8] as a reference.

An extensive analysis was done in [8] on the execution time of a trajectory calculation on a radiation-hardened processor and it was concluded that an optimal trajectory can be calculated on average in 0.7 s. The tests were performed in a BAE RAD750 PowerPC, which is part of a flight software testbed at NASA Jet Propulsion Laboratory. The execution time varies if the solution is feasible or not. Figure 2 shows the mean run-times of the optimization problem as function of the time of flight. The red dot in this figure separates infesible times-of-flight (on the left) from feasible ones (on the right). For feasible times of flight, the execution time does not oscillate much and the worst-case execution time is 0.8 s. For infeasible times-of-flight, the execution time fluctuates between 0.4 s and 1.3 s. Thus, the worst-case execution time that will be used is 1.3 s for infeasible trajectories and 0.8 s for feasible trajectories.

This is just a reference point and the execution times of this work can be slightly different. Testing the real execution times of the improved guidance function in a flight processor must be addressed in future work. Nonetheless, these execution times are

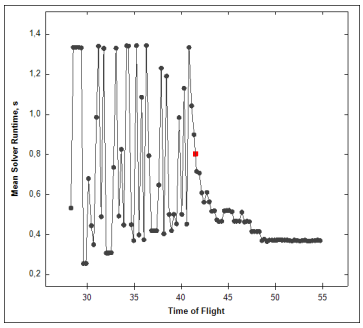

Figure 2: Mean execution times of the optimization problem for different times-of-flight [8].

the worst-case scenario and can be used as a reference in the demonstration of the optimal trigger search method.

4.3. Trigger Search Time-step

To choose a time-step for the optimal trigger search, it is important to know how large is the window of times-of-flight where the optimal t_f of the next state can be. It was seen that the optimal t_f of the next state lies around the optimal t_f of the previous loop minus the time-step of search. This t_f will be called the predicted time-of-flight. Occasionally the optimal t_f of the next step is smaller or larger than the predicted t_f. The difference between the larger and smaller possible times-of-flight will be called the window of times-of-flight where the optimal t_f of the next step can be. The size of this window will determine how many trajectories have to be computed for each trigger search loop and, consequently, the time-step of the search. If the size of the window is, for example, nine seconds, not all nine corresponding trajectories must be calculated. An analysis was made to measure the consumption of trajectories with a t_f larger than the optimal. It was seen that if the t_f of a trajectory is one or even two seconds larger than the optimal t_f, the additional fuel consumption is lower than 2%. Therefore, in a window of nine seconds, only four times-of-flight could be tested with acceptable results.

The size of such windows of times-of-flight was analyzed for different trigger search time-steps. Table 1 lists the window size for different time-steps used in the optimal trigger search. Each trigger search time-step was tested in 1500 descent trajectories, starting from different initial states. It was seen that, for every time-step tested, only three times-of-flight must be tested.

In the optimal trigger search loop, when computing the trajectories of the different times-of-flight, the worst-case scenario would be to have just one feasible t_f and the remaining times-of-flight infeasible, that can take longer to compute. So, to compute three trajectories, corresponding to the three t_{fs} of Table 1, the worst-case execution time would be $1.3 + 1.3 + 0.8 = 3.4$s. Therefore, it was decided to use the trigger search time-step of 4 s, since it is the minimum time-step of Table 1 larger than 3.4 s.

5. Re-optimization in real time

Another development to the pinpoint landing algorithm is a periodic loop for re-optimizing the trajectory during powered flight. Re-optimizing the trajectory improves the robustness of the guidance and allows an efficient correction of errors. This means that, whenever a dispersion in fight occurs, instead of having a trajectory control constantly correcting that error, a new fuel-optimal trajectory can be produced from the dispersed position.

The time sequence for the re-optimization case is simple. Whenever the spacecraft starts following the last thrust profile generated, there is enough time to compute the future state and its associated trajectory. This can be quickly computed with few trajectory calculations since the window of the possible times-of-flight is small.

The ideal condition would be to re-optimize the trajectory as frequent as possible to correct possible errors sooner in the powered trajectory, leading to smaller "jumps" in the thrust magnitude and direction. However, the frequency of the re-optimization has to take into account the computations execution of the next loop. As in the optimal trigger search, the size of the times-of-flight window of the next re-optimization loop will determine the re-optimization time-step. Table 2 lists the window of possible times-of-flight for different re-optimization time-steps and the worst-case execution time associated. Again, the worst-case scenario is when all times-of-flight generate infeasible solutions, with exception of one t_f, which will correspond to the optimal solution. Therefore, the time-step selected for the re-optimization was 5 s since it is the smallest time-step that is larger than the worst-case execution time.

In the context of the re-optimization, a final vertical descent phase was included. Instead of pointing directly to the target, the algorithm produces trajectories to reach a specific point above the target with a specific downward velocity and then a final thrust profile is produced to vertically land on the target. This was included to assure a vertical and smooth descent in the last seconds of the flight, which improves the safety of the landing. Other approaches could be made to achieve the same effect, such as reducing the maximum tilt angle and maximum thrust in the last seconds of flight. However, having a separate final vertical descent is a simple and robust solution, that integrates well in the re-optimization scenario.

Table 1: Range of possible times-of-flight in the optimal trigger search and worst-case execution time associated.

Search time-step [s]	Minimum difference from predicted t_f [s]	Maximum difference from predicted t_f [s]	# of Trajectory Calculations	Worst-Case Execution Time [s]
2	-1	2	3	3.4
3	-2	2	3	3.4
4	-2	2	3	3.4
5	-1	2	3	3.4
6	-2	3	3	3.4

Table 2: Range of possible times-of-flight and worst-case execution time in the re-optimization loop.

Re-optimization time-step [s]	Minimum difference from predicted t_f [s]	Maximum difference from predicted t_f [s]	# of Trajectory Calculations	Worst-Case Execution Time [s]
2	0	6	3	3.4
3	0	7	4	4.7
4	0	8	4	4.7
5	0	9	4	4.7
6	0	8	4	4.7
7	0	9	4	4.7

6. Drag Effect

Due to the quadratic formulation, the drag force could not be included in the original pinpoint landing algorithm. If drag is not taken into account, especially when acceleration due to drag is a significant part of the acceleration profile, the flight path will be completely different from what was calculated and the trajectory has to be constantly corrected. To overcome this situation, we tried to linearly approximate the drag force so that it could be included in the problem formulation. However, this cannot be done because its inclusion would invalidate the change of variables made and consequently the overall formulation of the problem.

There is however, an alternative. Since the algorithm does not take drag into account, the thrust profile produced can be interpreted as the optimal force that should act on the spacecraft. Since the drag effect is helping to slow down the spacecraft, it can be subtracted from the calculated thrust profile, if we measure the drag force using the spacecraft's accelerometers.

However, subtracting the drag force has side effects that cannot be ignored. Since the beginning of the powered descent starts at relatively high speeds, the drag acceleration is about 40 % of the maximum thrust. This value is even higher than the minimum thrust provided by the engine. Thus, subtracting the drag from the optimal force can drastically change the direction and magnitude of thrust. This could mean that the spacecraft had to turn upside down at the beginning of the maneuver and have thrust magnitude lower than the minimum possible.

To address this problem, a simple function was included to check and correct if the resulting thrust profile violates any of the constraints. If the resulting thrust direction violates the maximum pointing angle, the direction of thrust is set at the original angle calculated by the algorithm and the thrust magnitude remains the same. In the case that the resulting thrust magnitude is lower than the engine minimum, the thrust magnitude is changed to the minimum value possible, while the thrust direction stays the same. Both violations can happen at the same time. In that case, the direction of thrust is corrected first, and then the magnitude. Tests showed that this method performs well because it incorporates smoothly in the re-optimization technique. Whenever the thrust vector has to be corrected, the trajectory is slightly different from what was calculated but the differences are small and are corrected in the next re-optimization loop. The thrust command corrections always increase the vertical component of thrust by decreasing the thrust-pointing angle or increasing the thrust magnitude. Therefore, the spacecraft will always have enough vertical deceleration and there is no risk of creating infeasible solutions due to these adjustments.

6.1. Pinpoint Landing Example

We developed an example to illustrate a pinpoint landing on Mars. The simulations were developed in Simulink and the landing parameters are listed

Table 3: Landing parameters.

Parameter	Value
g	$\begin{bmatrix}0 & 0 & 3.725258\end{bmatrix}^T$ m/s^2
m_{wet}	346 kg
I_{sp}	311 s
n	1
ϕ	0 deg
$\overline{T}$	3600 N
T_{max}	100 %
T_{min}	30 %
θ_{max}	60 deg
γ_{gs}	5 deg
$trg_{\Delta t}$	4 s
$reopt_{\Delta t}$	5 s

in Table 3. The target is located at the origin of the reference frame and the initial conditions are

$$r_0 = \begin{bmatrix}-5540 & 570 & 6420\end{bmatrix}^T \text{ m}, \tag{14a}$$

$$v_0 = \begin{bmatrix}207 & -25 & -103\end{bmatrix}^T \text{ m/s}, \tag{14b}$$

$$m_0 = 346 \text{ kg}. \tag{14c}$$

All trajectories are calculated to end at an altitude of 20 m above target, with downward velocity of 5 m/s. Afterwards, a final vertical and smooth deceleration is performed to land at target with zero velocity. In this flight, the decision to trigger the engines was made at the state:

$$r_0 = \begin{bmatrix}-832 & 1 & 2884\end{bmatrix}^T \text{ m}, \tag{15a}$$

$$v_0 = \begin{bmatrix}130 & -17 & -147\end{bmatrix}^T \text{ m/s}, \tag{15b}$$

Figure 3 illustrates the time evolution of position, velocity, acceleration, net thrust force, throttle level and the angle between the thrust vector direction and the vertical direction. During the powered trajectory, three re-optimizations were made and the total time-of-flight was 36 s. The fuel consumption was 34.6 kg. Figure 4 compares the thrust-pointing angle of this trajectory with or without a thrust command correction. It can be seen that the thrust commands would violate the maximum-pointing angle constraint and therefore they are corrected in the first ten seconds of the powered flight. Consequently, the flight path is slightly different from what was expected but everything is corrected with the re-optimization.

Figure 5 illustrates how the re-optimization method affects the thrust profile. The solid lines represent the re-optimization thrust profile and the dashed ones the original thrust profile where there is no re-optimization, no final vertical burn and no thrust command correction. The last five seconds of the re-optimization thrust profile correspond to the final vertical burn.

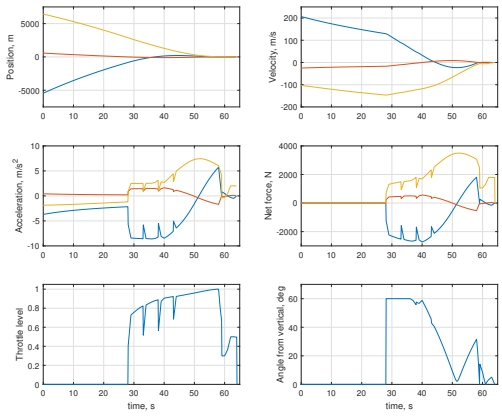

Figure 3: Mars pinpoint landing example with $r_0 = [-5540, 570, 6420]^T$ m, $v_0 = [207, -25, -103]^T$ m/s, $m = 346$ kg and $t_f = 36$ s. The fuel consumption is 34.6 kg.

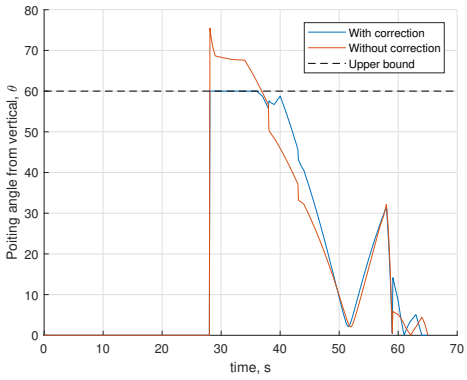

Figure 4: Thrust pointing constraint with and without correction.

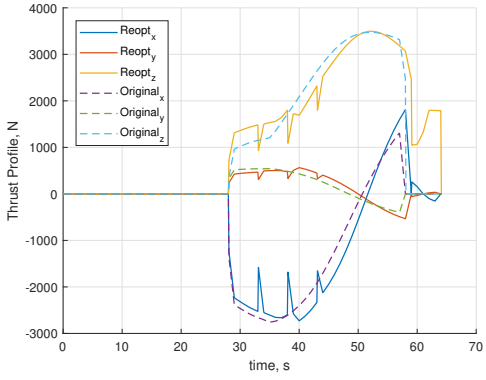

Figure 5: Thrust profile with and without re-optimization.

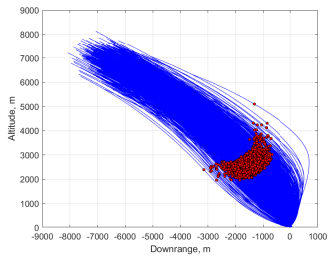

Figure 6: Plot of the 1537 trajectories.

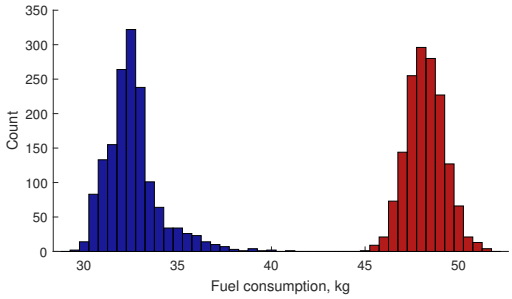

Figure 7: Comparison of fuel consumption between the improved convex guidance algorithm, on the left, and an Apollo-like guidance algorithm on the right. Both guidance functions were capable of pinpoint landings.

7. Results

To obtain the typical shape of the flight path, a plot was made with the 1537 simulated trajectories. Figure 6 illustrates the vertical profile of these trajectories, where the marked dots represent the trigger location. It was observed that the optimal altitude to trigger tends to be in the range of 2 km to 3 km and that in some trajectories, the optimal behaviour is to overshoot the target.

In every case studied, the improved guidance function was able to generate a feasible trajectory to land on the target. Although during each flight the guidance function calculates trajectories to start from dozens of locations, during the trigger search and the re-optimization phases, not a single case of in-feasibility was found, demonstrating two aspects of the improved guidance function. First, that calculating three trajectories in the trigger search and four trajectories in the re-optimization phase is enough to find a feasible and optimal solution. Second, it also shows that the subtraction of the drag force and the thrust commands corrections associated do not produce infeasible trajectories and do not impact the feasibility of subsequent trajectory calculations.

The average fuel consumption of the landings is 32.6 kg, with the worst case being 45.77 kg. This worst case consumption happened only once in the 1537 trajectories and the second worst case is 40.8 kg.

To study the fuel-optimality of the improved guidance function, its consumption was compared to an enhanced Apollo-like PD algorithm, capable of pinpoint landing but not fuel-optimal. This Apollo-like guidance function was used in end-to-end simulations of pinpoint landings on Mars [9]. The conditions of both simulations were set to be the same. The difference is that more than 1537 initial states were studied with the Apollo-like guidance. Figure 7 illustrates the fuel consumption of both cases. The convex optimization guidance function, being fuel-optimal, is capable of a lower consumption. On average, it is possible to save 30 % of fuel when using the improved convex optimization guidance function.

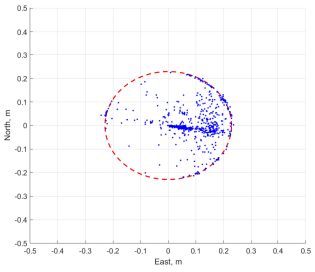

Figure 8: Landing ellipse of the convex optimization guidance function.

Regarding landing accuracy, it was possible to land with better than half-meter precision in every case studied. Figure 8 illustrates the landing ellipse of the improved guidance function, where the worst lateral error is 25 cm.

It was seen that the horizontal errors in the final position are minimal and do not affect the mission safety. Landing with half-meter horizontal precision is considered to be excellent. However, a vertical error was observed in the final position, which cannot be ignored. Figure 9 illustrates this final vertical error, which ranges from -0.53 m to 0.64 m. Having final positions above the ground would mean that when the engines shut down, the lander would fall, which is not desirable. In the case where the final position is below ground level, the spacecraft can impact the surface still with downward velocity. Nonetheless, these errors are considered to be small and can be easily corrected with trajectory control systems. Note that all the trajectories profile calculated by the convex solver land perfectly on target with zero velocity. Thus, the position and velocity profiles can be used as reference for the trajectory control. The final position errors are probably explained by the approximations in the original formulation of the optimization problem [3].

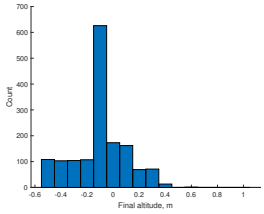

Figure 9: Final vertical position error.

Table 4: Minimum, average and maximum executions times of guidance function tasks.

Task	Execution times [ms]		
	Min.	Average	Max.
Line-Search	60.3	78.3	155.2
Trigger Search	32.2	52.5	76.6
Re-optimization	24.6	89.9	151.7

The final velocity errors are minimal and can be ignored. The final velocity in the horizontal directions ranges from $-0.006\,\mathrm{m/s}$ to $0.007\,\mathrm{m/s}$. The final vertical velocity error ranges from $-0.04\,\mathrm{m/s}$ to $0.08\,\mathrm{m/s}$, which is also considered not large enough to raise concerns.

7.1. Execution Times

The final code is supposed to run on a flight computer but it is worth evaluating its execution times on a regular computer. These simulations were run on a laptop, with Windows 10 and 16 GB of RAM. The processor is a six-core eight-generation i7 (model 8750H) with a base clock speed of 2.2 GHz. During simulations, the processor was working at an average frequency of 3.9 GHz and MATLAB and Simulink were the only tasks running.

Table 4 lists the minimum, average and worst-case execution times of the different tasks of the guidance function. These tasks are the first line-search, the optimal trigger search and the re-optimization computations. What is being studied is the total execution time to find the optimal trajectory. In other words, the total execution time of the line search, the three trajectory computations in the trigger search and the four trajectory calculations in the re-optimization phase.

The results imply that the flight processor can be up to 33 times slower than processor used, in order to compute the re-optimization trajectories under 5 seconds. For the optimal trigger search, the processor can be 50 times slower. In single trajectory calculations, it was on average possible to find a solution under 19 ms. The code prior to the actual call of the solver, which is the construction of data and matrices to provide as input to the solver, is running in less than 1 microsecond. Therefore, the run-time of a trajectory calculation is being affected only by the call of the solver.

7.2. Optimality

Since the optimal search is performed with a period of four seconds, there will be cases where the spacecraft will trigger its engines before the exact optimal location. Additionally, when calculating the trajectory of the next loop on the optimal trigger search and in the re-optimization loop, not every t_f is tested. In the trigger search, only three out of five possible times-of-flight are tested and in the re-optimization loop, only four out of nine times-of-flight are tested. Therefore, in some cases, the optimal t_f will not be selected. These deviations from the optimal scenario can accumulate, therefore, it is important to test the optimality of the landings. To test this, we performed additional 1537 simulations where the period of the optimal trigger was set at 1 s. Also, in the trigger search and re-optimization loops, every t_f of the window of times-of-flight was tested. This is still an approximation since the real optimal trigger time could be between the trigger times tested, but having a time-step of one second was considered to be a good approximation.

The fuel consumption of the optimal scenario was compared with the actual landing consumption to measure the overall optimality of the improved guidance function. It was seen that 96 % of the flight trajectories are at least 98 % fuel-optimal and that the worst case of optimality was 89 %. This shows that even if the optimal trigger search is performed with a period of four seconds, and that not every t_f is tested in trajectory calculations, it is possible to find a near-optimal trajectory. It is worth mentioning that in the two worst fuel consumption cases, the optimality of the solution was 97 % and 99 %. Thus, they were not caused by the improved guidance function design.

8. Conclusions

The main goal of this thesis was to develop a fuel-optimal guidance function for pinpoint landings on Mars, based on a convex optimization method developed by NASA. Some developments were made to improve robustness and readiness of the original method.

The original formulation of the problem was successfully implemented in C source code. Considerable changes were made to transform the problem into matrix format in order to use the ECOS solver. It was confirmed that the numerical results were in agreement with the original literature of the convex programming approach to pinpoint landing, which validated the transcription. The C code was optimized to both memory use and execution time. The

ECOS solver was designed for embedded applications and so it is possible to implement the source code of this work on any spacecraft. Additionally, the code was developed with open-source available tools, making it inexpensive and flexible.

The improved guidance function was tested for more than 1500 cases of Mars landing and it was always possible to land with better than half-meter precision. The performance of this guidance function was compared to an enhanced Apollo-like guidance algorithm, capable of pinpoint landings but not fuel-optimal. Using the convex optimization guidance function, it was possible to save 30 % in fuel consumption.

The methods for finding the optimal trigger and re-optimizing the trajectory were tested and have shown to be reliable and robust. Not a single case of infeasibility was found. The optimality achieved with these two methods was seen to be at least 98 % for 96 % of the cases studied. Thus, it was concluded that the guidance function is capable of generating nearly optimal trajectories and to decide when is the optimal time to trigger the powered descent. Additionally, in a preliminary design phase, the execution times of the guidance function appear to be viable for real-time calculations.

Since there was a successful validation of the guidance function design, the next step is to implement the guidance function with control systems in a more realistic simulation. The major aspects to test are if the spacecraft is capable of following the thrust profile calculated and correct the final position error. Other important issue to study is the extra fuel required to correct eventual deviations and the adequate risk level of the guidance function.

References

[1] B. Açıkmeşe, J. Carson, and L. Blackmore. Lossless convexification of nonconvex control bound and pointing constraints in the soft landing optimal control problem. *Control Systems Technology, IEEE Transactions on*, PP:1–1, 11 2013.

[2] B. Açıkmeşe, J. Casoliva, J. M. Carson, and L. Blackmore. G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing. In *Concepts and Approaches for Mars Exploration*, volume 1679 of *LPI Contributions*, page 4193, June 2012.

[3] B. Açıkmeşe and S. Ploen. Convex programming approach to powered descent guidance for mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5):1353–1366, 9 2007.

[4] B. Açıkmeşe, D. Scharf, L. Blackmore, and A. Wolf. Enhancements on the convex programming based powered descent guidance algorithm for mars landing. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 08 2008.

[5] L. Blackmore. Autonomous precision landing of space rockets. *The Bridge*, 46:15–20, 01 2016.

[6] R. D. Braun and R. M. Manning. Mars Exploration Entry, Descent, and Landing Challenges. *Journal of Spacecraft and Rockets*, 44(2):310–323, 2007.

[7] A. Domahidi, E. Chu, and S. P. Boyd. Ecos: An socp solver for embedded systems. *2013 European Control Conference (ECC)*, pages 3071–3076, 2013.

[8] D. Dueri, B. Açıkmeşe, D. Scharf, and M. W. Harris. Customized real-time interior-point methods for onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics*, 40, 08 2016.

[9] J. Ferreira, T. Hormigo, J. Seabra, D. Esteves, F. Câmara, and J. Oliveira. End-to-end gnc design, test and validation of mars pinpoint landing with retropropulsion. In *69th International Astronautical Congress*, Bremen, Germany, 8 2018.

[10] M. W. Harris. *Lossless convexification of optimal control problems*. PhD thesis, The University of Texas at Austin, 2014.

[11] S. Ploen, B. Açıkmeşe, and A. Wolf. A comparison of powered descent guidance laws for mars pinpoint landing. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 08 2006.

[12] D. Way. On the use of a range trigger for the mars science laboratory entry, descent, and landing. In *2011 Aerospace Conference*, pages 1–8, March 2011.

[13] A. Wolf, C. Graves, R. Powell, and W. Johnson. Systems for pinpoint landing at mars. *Advances in the Astronautical Sciences*, 119, 02 2005.