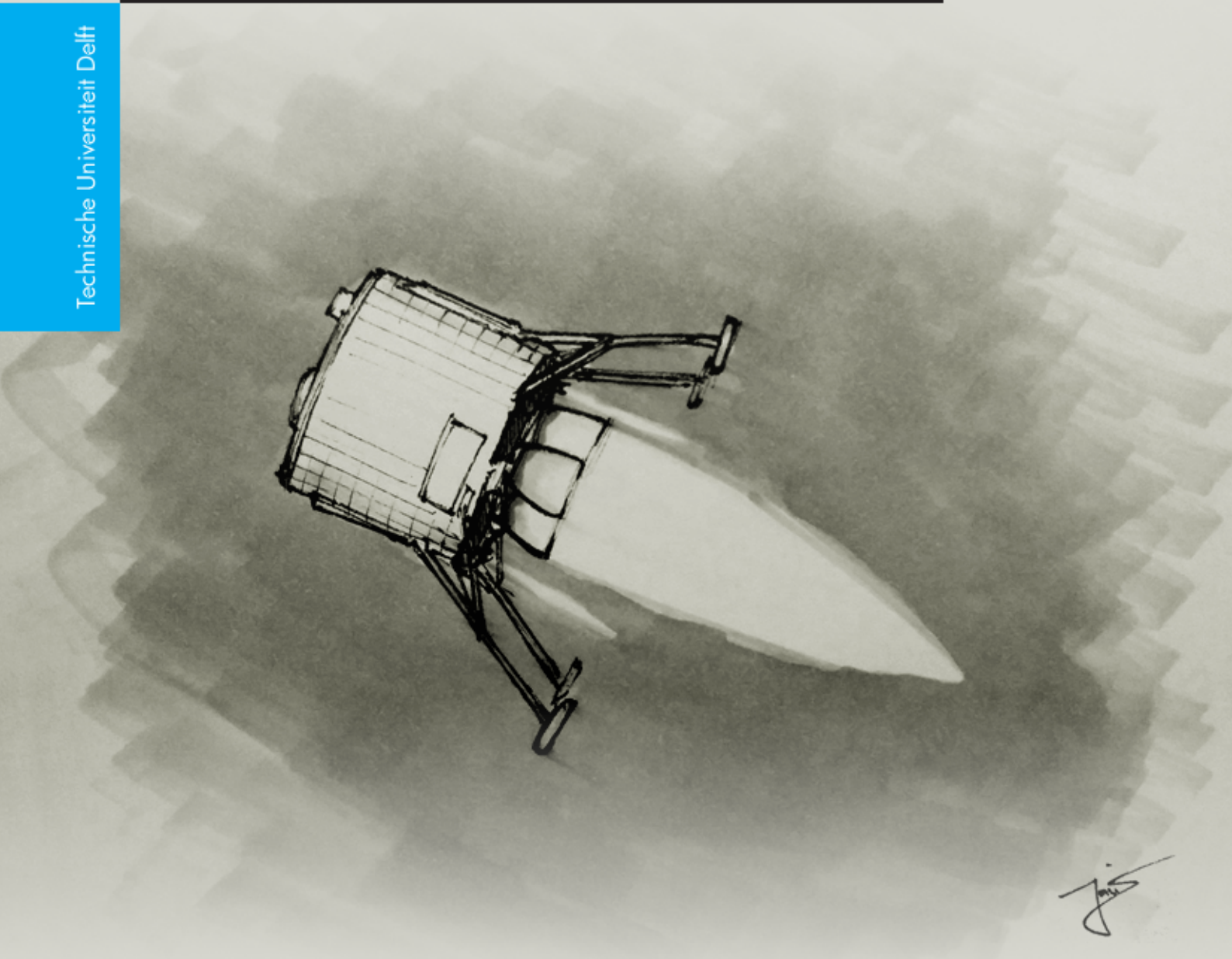
Convex Guidance, Navigation, and Control for Pin-Point Lunar Landing

Krister Mark de Ridder

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

Convex Guidance, Navigation, and Control for Pin-Point Lunar Landing

Krister Mark de Ridder
June 28, 2016
4244710

Cover art by
Joris van Leeuwen

TU Delft Delft University of Technology

Abstract

Convex guidance is a developing guidance algorithm that seeks to unify the trajectory optimisation for pin-point landings. The unification would mean the elimination of the need of different optimisation techniques for different stages of the landing. Using convex optimisation, convex guidance is guaranteed to find the global-optimum of the design space.

This study analysis the convex guidance algorithm in depth and establishes the magnitude of the errors introduced by simplifying assumptions. Subsequently, the performance and robustness of convex guidance is investigated in a full guidance, navigation and control loop.

It is found that convex guidance can practically be used for a pin-point Lunar landing. However, the combination of a large discretisation step with a zero-order hold function for the optimal control proved to be an inefficient solution for optimising the spacecraft's attitude. A linear control function proved to greatly reduce the attitude control effort, and requires far less computational effort than reducing the discretisation step of convex guidance. Therefore, it is recommended to at least use a first-order function for the control vector.

Furthermore, because no attitude dynamics are modelled by convex guidance, large attitude discontinuities can occur during the landing, and no initial- or final-boundary attitude conditions can be imposed. Carefully tuned controllers can deal with these discontinuities without upsetting the system, but deviations from the optimal trajectory cannot be avoided. Therefore, a future study is recommended into expanding convex guidance with angular rate limiters to improve the robustness of the algorithm.

Contents

Acronyms

AG Approach Gate

AP Approach Phase

DCM Direction Cosine Matrix

DSN Deep Space Network

EKF Extended Kalman Filter

GNC Guidance, Navigation and Control

HDA Hazard Detection and Avoidance

LLO Low Lunar Orbit

MBP Main Braking Phase

MSFN Manned Space Flight Network

PF Particle Filter

PID Proportional-Integral-Derivative

PD Proportional-Derivative

PDI Powered Descent Initiation

PWM Pulse-Width Modulation

QF Quaternion Feedback

RCS Reaction Control System

RK4 fourth-order Runge-Kutta

SDRE State-Dependent Riccati Estimator

SOCP Second-Order Cone Problem

SPKF Sigma-Point Kalman Filter

TG Terminal Gate

UKF Unscented Kalman Filter

Symbols

α	Glide-slope angle	rad
Γ	Thrust slack variable	$\mathrm{m/s^2}$
γ	Sensor angle	rad
$\delta_{\mathcal{L}}$	Landing site latitude	rad
δ_s	Spacecraft latitude	rad
ϵ	Elevation	rad
ζ	Damping ratio	$-$
θ	Pointing margin	rad
θ	Roll angle	rad
θ	Rotation over eigen-axis	rad
μ	Gravitational parameter	$\mathrm{m^3/s^2}$
ρ_p	Propellant density	$\mathrm{kg/m^3}$
ρ_s	Structural density	$\mathrm{kg/m^3}$
σ	Acceleration slack variable	$\mathrm{m/s^2}$
$\boldsymbol{\sigma}$	Modified Rodrigues Parameters	$-$
$\tau_{\mathcal{L}}$	Landing site longitude	rad
τ_s	Spacecraft longitude	rad
$\boldsymbol{\tau}$	Specific thrust vector	$\mathrm{m/s^2}$
ϕ	Pitch angle	rad
ψ	Yaw angle	rad
$\omega_{\leftmoon}$	Angular rate of moon	$\mathrm{rad/s}$
ω_n	Natural frequency	$\mathrm{1/s}$
$\boldsymbol{\omega}$	Angular rate of spacecraft	$\mathrm{rad/s}$

a	Mass flow to thrust ratio	$\mathrm{s/m}$
$\mathbf{e}$	Eigen-axis	$-$
$\mathbf{F}$	Force	N
g	Gravitational acceleration	$\mathrm{m/s^2}$
h_{cm}	Centre of mass height	m
h_p	Propellant height (in tank)	m
h_s	Structure height	m
$\mathbf{I}$	Inertia matrix	$\mathrm{kg\ m^2}$
I_{sp}	Specific impulse	s
$\mathbf{K}$	Gain matrix	$-$
m	Total mass	kg
$\dot{m}$	Mass flow	$\mathrm{kg/s}$
m_p	Propellant mass	kg
m_s	Structure mass	kg
$\hat{\mathbf{n}}$	Normal vector	$-$
$\mathbf{P}$	Estimation covariance matrix	$-$
$\mathbf{Q}$	Propagation covariance matrix	$-$
$\bar{\mathbf{q}}$	Quaternion of rotation	$-$
$\hat{\mathbf{q}}$	Estimated attitude quaternion	$-$
R	Radius of moon	m
$\mathbf{R}$	Direction cosine matrix	$-$
$\mathbf{R}$	Observation covariance matrix	$-$
r	Radius	m
r_s	Structure radius	m
r_t	Propellant tank radius	m
$\mathbf{r}$	Position	m
$\mathbf{T}$	Thrust	N
T_l	Lower thrust bound	N
T_h	Upper thrust bound	N
t_{set}	Settling time	s
$\mathbf{u}$	Control vector	$\mathrm{m/s^2}$
$\mathbf{v}$	Velocity	$\mathrm{m/s}$
v_e	Exhaust velocity	$\mathrm{m/s}$
$\mathbf{x}$	State vector	$-$
$\hat{\mathbf{x}}$	Estimated state vector	$-$
$\mathbf{y}$	Observation vector	$-$

1 Introduction

Almost all Lunar landings performed to date have been executed by the USA and the USSR during the Cold War. After the 1970s the interest in Lunar landings seemed to have been largely lost. In more recent years, space agencies from around the world are planning on landing on the Moon again. In 2013, China was the first to lead the new wave of Moon landings with their Chang'e 3 spacecraft. Japan, India and Russia are all planning to follow China to land on the Moon within this decade. ESA and NASA developed plans to return to the Moon, as well. Unfortunately, both their missions have been canceled due to financial setbacks. Besides these government-funded projects, Google and XPrize have a current challenge to commercially reach the Moon by the end of 2017. For this specific challenge, one team recently secured a launch agreement for the second half of 2017.[1]

The novelty of these new Lunar landings, compared to the early Lunar landings during the Cold War, is the pin-point landing of the spacecraft. Whereas early landings could be kilometres off their targeted landing site (Renzetti, 1969) (Bennett, 1970), modern landing missions require much better precision to target specific sites of great scientific value. For ESA's Lunar Lander and NASA's Altair spacecraft, for example, the landing-precision requirement was specified to be within a few hundred metres of the initially targeted landing site (Ely et al., 2012) (Kerr et al., 2013). This includes possible retargeting after hazard identification. The identification of the landing site and potential hazards require the spacecraft to be fitted with highly accurate sensors and sophisticated algorithms to scan the area. As a result, the landing trajectory becomes highly constrained to make sure the landing site is within the field of view of the sensors (Gerth, 2014).

To facilitate the increasing demand of pinpoint landings with highly constrained trajectories, new guidance algorithms are developed with high landing accuracies. One such algorithm, developed by Açikmeşe and Ploen (2005), is based on convex optimization to guarantee convergence to the global optimum. The advantage of this property is that it can be used on board the spacecraft to directly evaluate a new trajectory when a hazard has been detected. Gerth (2014) expanded this algorithm with, for example, an improved gravity model, allowing con-

[1] http://lunar.xprize.org/

vex guidance to be used for the main braking phase as well, instead of only for the approach phase. This eliminates the need of different guidance algorithms for each stage of the landing.

Convex guidance is a promising new guidance algorithm. However, most studies into guidance algorithms assume instantaneous attitude control and perfect state knowledge. Therefore, this study aims to investigates the practical application of convex guidance in a full Guidance, Navigation and Control loop. With this goal in mind, the main question of this thesis is formulated as follows:

Is convex guidance robust enough for use in a guidance, navigation and control loop, to the purpose of a pin-point Lunar landing?

To answer the main question, the research is divided into the following sub questions:
- What simplifying assumptions are made for convex guidance?
- How large are the errors introduced by the simplifying assumptions?
- How do these assumptions influence attitude control?
- How do these assumptions influence trajectory tracking?
- How does convex guidance perform in an integrated Guidance, Navigation and Control loop?

Chapter 2 gives an overview of past and future Lunar missions. It also provides a detailed description of ESA's Lunar Lander, which is used as the reference vehicle for this study. The chapter closes with a discussion on the assumptions made when modelling the spacecraft. This is followed by Chapter 3, discussing fundamental mathematics for this study, like coordinate systems, reference frames and the equations of motion. Chapter 4 is dedicated to convex guidance. The chapter starts with a detailed description of the algorithm with its simplifying assumptions. Subsequently, extensions to the algorithm are discussed and the problem is transcribed to the conical form. Chapter 5 verifies the transcription and the implementation of the guidance algorithm against literature, while Chapter 6 quantifies the errors that are introduces by the simplifying assumptions. Chapter 7 follows this up by introducing control algorithms to control the attitude of the spacecraft and correct for deviations from the optimal trajectory. The chapter is concluded by performance and sensitivity analyses on the convex guidance and control loop. This is followed by Chapter 8, introducing navigation errors to the system. This chapter investigates the performance and robustness of a guidance, navigation and control loop with convex guidance. Ultimately, Chapter 9 gives the conclusions and recommendations of this study.

2 Heritage

This chapter aims to establish a reference mission on which the study is based. To do so, Section 2.1 investigates missions from the past, Section 2.2 discusses planned future Moon missions and Section 2.3 goes into the findings of theoretical Moon-landing studies. Subsequently, Section 2.4 provides a reference scenario and vehicle for this study.

2.1 Past Missions

Almost all landings on the Moon up to this day have been performed by the USSR or the USA in the 1960s and 1970s, as part of the space race during the Cold War. In early 1966 the USSR achieved the first landing with their Luna 9 probe. The USA followed with the Surveyor 1, mid 1966.[1]

For its landing, the Surveyor 1 utilized Sun sensors, a star tracker and gyroscopes to determine its attitude. Initial navigation was performed by tracking the spacecraft using NASA's Deep Space Network (DSN) on Earth, as discussed by Renzetti (1969). During the last stage of its descent, only an altimeter radar and a Doppler radar were used in combination with Vernier engines. Because a fixed altitude versus velocity schedule was used for the landing, no navigation or guidance with respect to its target was available. This resulted in landing 14 km off-target. However, the primary goal of this mission was not to perform a high accuracy landing, but to show that making a soft landing on the Moon was possible at all.

After one more Luna probe in late 1966 and four more Surveyor probes through 1967 and 1968, Apollo 11 was the first manned Moon landing by the USA in mid 1969.

Like the Surveyor missions, the Apollo was tracked by the DSN and Manned Space Flight Network (MSFN) of NASA. This was the main means of navigation for Apollo. For backup (and in fear of the USSR jamming spacecraft-ground intercommunications), an onboard navigation system was developed as well. In practice, this system was mainly used to verify the

[1] http://solarsystem.nasa.gov/

spacecraft tracking by the DSN and MSFN from the Earth's surface.[2] Onboard navigation for Apollo 11 was performed by measuring angles between stars and landmarks on both the Earth and the Moon. From this technique the attitude of the spacecraft was determined as well as the relative position to the Earth or the Moon. During the approach for landing, the Apollo astronauts noted timestamps when overflying Lunar landmarks to verify navigation. The Lunar Lander could perform the landing autonomously, using Quadratic Guidance and Velocity Nulling Guidance, as discussed by Parker (1974). However, the Apollo mission relied on the astronauts to identify hazards and to take manual control to avoid hazards near the landing site. The Apollo Moon landings were very influential in terms of Guidance, Navigation and Control (GNC) technology for further lander missions and studies. Whereas the Surveyor mission directly descended to the Moon's surface from a hyperbolic orbit, the Apollo mission first entered a circular Low Lunar Orbit (LLO) before descending to the surface. Going into an elliptical or circular orbit before landing is safer than directly descending from a hyperbolic orbit, because in case of an engine failure, the spacecraft would fly by the Moon instead of crash into it.

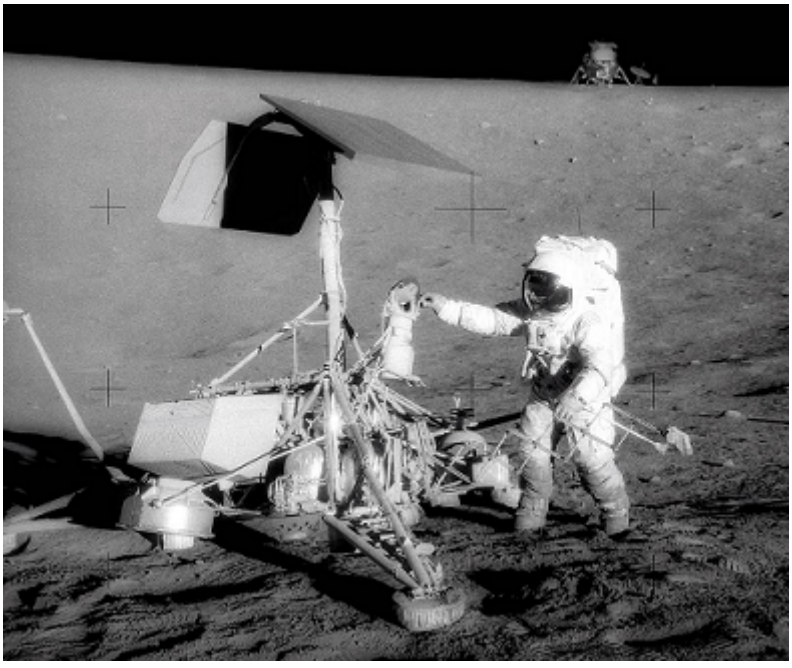


Figure 2.1: Apollo 12 commander Pete Conrad next to Surveyor 3 on the surface of the Moon, with the Apollo 12 Lunar Module 'Intrepid' in the background. Image source: NASA.

Late 1969, Apollo 12 followed Apollo 11 as the second manned landing on the Moon. The USSR responded with a fully robotic sample return mission and a fully robotic Lunar rover, both in 1970. Four more manned missions were performed by the USA in 1971 and 1972.

[2] http://spaceref.com/missions-and-programs/nasa/apollo/apollo-lunar-landing-mission-symposium/apollo-navigation-guidance-and-control.html

The USSR sent one more rover and three more sample return missions to the Moon from 1972 through 1976.[3]

For the remainder of the 20th century, no more soft lunar landings were performed. Only research through remote observations was continued using orbiters. For the current century, only one soft Lunar landing has been performed so far. China landed Chang'e 3 in late 2013, carrying a robotic rover. Although Sun et al. (2013) discuss the Chang'e 3 mission in general, only little information is available on this Chinese lander. In the article, constant-thrust explicit guidance and quadratic polynomial guidance are mentioned amongst "other guidance modes", but their use or their relation with respect to the landing phases are never mentioned. Sun et al. (2013) further state that laser ranging was used for terrain scanning and hazard detection. The attitude and speed changes commanded by the guidance algorithm were executed by a Proportional-Integral-Derivative (PID) controller and Pulse-Width Modulation (PWM).

2.2 Future Missions

Despite the lack of recent soft landings on the Moon, a number of missions have been planned before the end of this decade by China, India, Japan, Russia and the USA in the form of landers, rovers and sample return missions. All these programmes are government funded.

Next to the government-funded projects, there is also the Google Lunar XPrize, a $30M competition to land a privately funded spacecraft on the Moon before the end of 2017. For this competition, Israeli team SpaceIL, recently announced to have secured a launch agreement for a Falcon 9, to be launched in the second half of 2017.[4]

The European Space Agency has been developing a Lunar lander as well, originally to be landed on the Moon's south pole in 2018. Unfortunately, this project was shelved in 2012 due to a lack of funding. Because ESA is now looking for collaboration with the Russians on a Moon lander, this project is unlikely to fly any time soon. Even though the future is unsure for the Lunar lander, its development has already returned some valuable research.

ESA aims to improve the navigation performance of the lander by using vision-based navigation. Fisackerly et al. (2011) discuss the techniques and challenges in using landmark recognition for absolute vision-based navigation. For this, camera images are compared to an onboard landmark database to determine the spacecraft's position. As a source for this database, data can be used from JAXA's SELENE (nicknamed Kaguya) or NASA's Lunar Reconnaissance Orbiter spacecraft. The optical navigation is combined with a direct range measurement to determine

[3] http://solarsystem.nasa.gov/

[4] http://lunar.xprize.org/

the lander's altitude. This greatly improves the navigation accuracy required for the landing.

For the later stage of the landing, a Hazard Detection and Avoidance (HDA) system was under development. Images would be combined with Lidar to generate a detailed map of the landing site. This way, hazards, like boulders or steep slopes, could be detected and avoided, if needed.

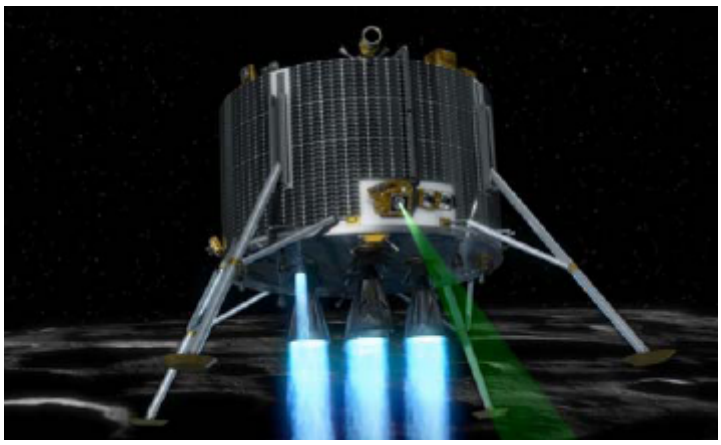


Figure 2.2: Artist's impression of ESA's Lunar Lander. Image source: ESA.

Kerr et al. (2013) discuss the guidance and control algorithms used by ESA's Lunar Lander. Time-varying PID controllers are employed, which use gain schedules versus time. Kerr et al. (2013) continued their study with closed-loop simulations, incorporating the full GNC and HDA systems. Monte Carlo runs were used, varying parameters like thrust misalignment, state errors (position, speed, attitude and angular rate) at the Powered Descent Initiation (PDI), mass distribution and remaining fuel uncertainty. Furthermore, diverts were also simulated at altitudes of 1,800 metres and 160 metres altitude. These simulations showed that ESA's Lunar Lander can reliably re-target the spacecraft and land with a two metre precision.

Whereas ESA's lander is unmanned, NASA has been designing the next manned missions to the Moon. The designed Altair Lunar Lander has a capacity of four crew members, whereas the Apollo Lunar Module only had a crew of two. Unfortunately, this project has also been canceled for now, making it unsure if Altair will ever land on the moon.

NASA designed the spacecraft to have a global landing capacity for the Moon, greatly improving its flexibility over other landers. Ely et al. (2012) discuss the navigation architecture and performance of Altair. An optical system was designed to identify landmarks on the Moon to determine the spacecraft's position. This system was given to have a 2.5 m uncertainty below an altitude of 5 km. Furthermore, the optical sensor would be mounted on a gimbal, so that the sensor could be focused on the landing site without having to change the spacecraft's attitude. Further navigational sensors used by Altair are: 1) Radar for altimetry and velocimetry, yielding

36 m and 0.16 m/s accuracies. 2) An inertial measurement unit with 30 μg uncertainty and 3) star trackers for attitude determination. As an external navigational source, radiometric tracking was proposed using NASA's DSN for orbit determination prior to the PDI. Radiometric tracking would provide 0.7 mm/s and 2 m accuracies for speed and position determination. For the fusion of the navigational data, Ely et al. (2012) use a Kalman filter.

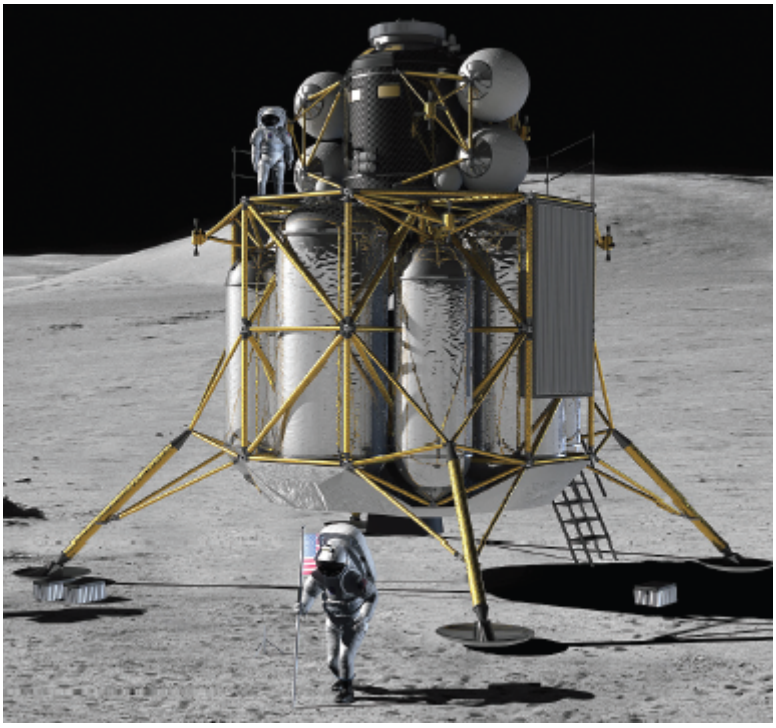


Figure 2.3: Artist's impression of NASA's Altair Lunar Lander. Image source: NASA.

Lee et al. (2010) give a more detailed description of the proposed GNC system for Altair, especially on the guidance algorithm. For guidance, a two-point boundary value problem would be solved to obtain a fourth-order polynomial function of time to describe the spacecraft's trajectory. Using Monte-Carlo simulations and the aforementioned sensors, Ely et al. (2012) continued with a performance analysis of the proposed GNC system. They showed that Altair could land on the Moon with position uncertainties of only 2.9 m.

2.3 Studies

Holtkamp (2014) performed a GNC study for a lander on Enceladus. Although it concerns a study for a landing on a different body, the results can be applied for Moon landings as well because both moons are atmosphereless bodies.

The goal of the study was to confirm if landing would be possible using current-day GNC and HDA technology. For this reason, relatively elementary algorithms were used.

At 3 km altitude, gravity-turn guidance was used. This aligns the thrust vector with the spacecraft's velocity vector. Arriving at 2 km altitude, the guidance algorithm was switched to Quadratic Guidance, which was used by the Apollo programme as well (see Section 2.1). In the very last stage of the landing, the algorithm switches to Velocity Nullifying Guidance. This algorithm purely focuses on matching the velocity vector with a predefined state. This allows to touch the lander down with no horizontal movements and a set vertical speed.

For the control algorithm, Holtkamp (2014) used a quaternion proportional-derivative controller. This is a Proportional-Derivative (PD) controller, modified to work with quaternions of rotation. For navigation, Inertial Navigation, Star Sensors, Horizon Sensors, LIDAR and orbiter-lander inter-distance measurements were used.

Simulations of this configuration yielded landing errors smaller than 0.5 m. Although this study does not add many new ideas to the field of GNC, it is a nice overview on how different techniques can be integrated for an end-to-end GNC-system.

Gerth (2014) took a new guidance algorithm developed by Açikmeşe and Ploen (2007) and expanded it by implementing new features. Whereas Açikmeşe and Ploen (2007) studied a landing on Mars, Gerth (2014) applied the new guidance algorithm to a landing on the Moon. The study concerned an algorithm that relied on the optimization of a convex problem. By formulating the guidance problem as a convex problem, this algorithm is guaranteed to find the global optimal solution. In principle it can be set to optimize any parameter, but it is most common to minimize the required propellant mass. Simulations by Gerth (2014) showed landing-position errors smaller than 1.2 cm. However, these are only the errors produced by the guidance algorithm, because perfect knowledge of the state vector was assumed in combination with instantaneous attitude changes. Therefore, one of Gerth's recommendations was to study the algorithm in a more representative environment, as an end-to-end loop is likely to yield larger errors.

2.4 Reference Mission

This study seeks to expand on the knowledge on the performance of Convex Guidance, as currently extended by Gerth (2014). Therefore, the same reference mission is chosen as used by the study of Gerth: ESA's Lunar Lander. Besides this mission already being used by Gerth as a reference, there is also plenty of information available about the mission in literature. ESA's Lunar Lander mission was outlined in Section 2.2. This section elaborates on the technical

details of the spacecraft itself and its trajectory.

Mass Distribution

Because no estimates of the mass distribution of ESA's Lunar Lander are publicly available, other information is used to obtain a rough estimate. To start, both Kerr et al. (2013) and Gerth (2014) show a lander mass of a little over 900 kg after touch down. Therefore, the dry weight of the spacecraft is assumed to be 900 kg.

Figure 2.4 shows that the body of ESA's Lunar Lander represents a cylinder of 2,560 mm in diameter and 1,659 mm in height. Furthermore, Riehle et al. (2012) show the Lunar Lander to carry four 0.370 m^3 cylindrical fuel tanks.

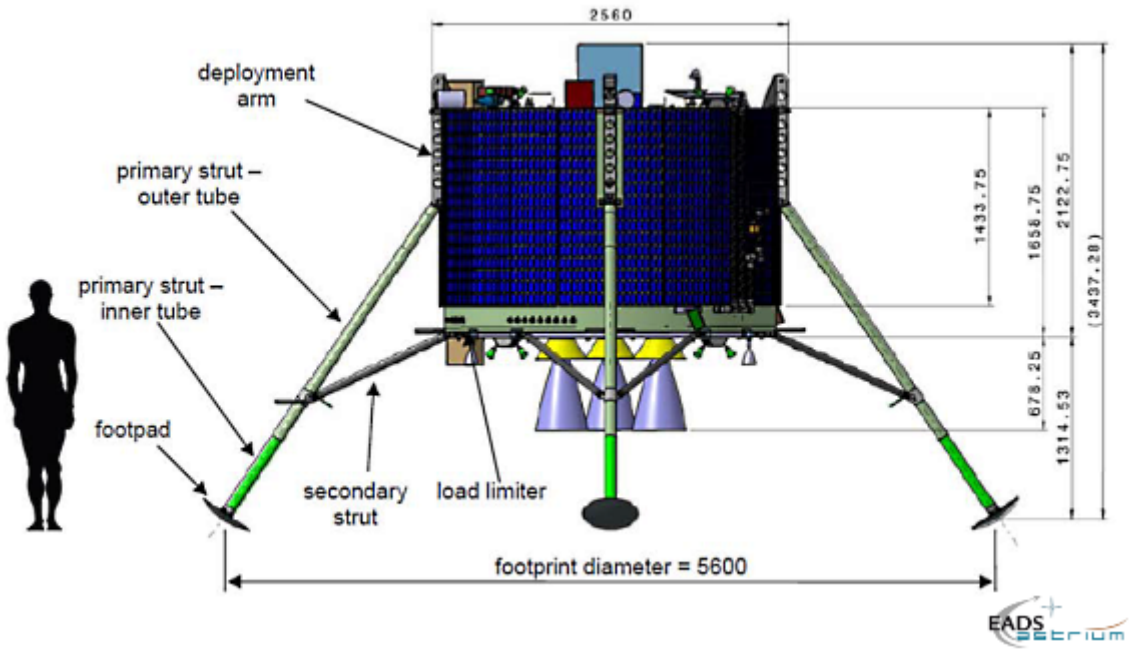


Figure 2.4: ESA Lunar Lander Dimensions. Source: Astrium.

Engines similar to those of ESA's Lunar Lander run on monomethylhydrazine (MMH) fuel and N_2O_4 oxidizer at a mixture ratio of 1.65.[5] These propellants have a density of 0.875 kg/L and 1.442 kg/L, respectively. Because these densities have the same density ratio as the mixture ratio of the engines, the volume flow of the two propellants are 1:1. As a consequence, the fuel levels in all four tanks will be similar throughout the flight. It is further assumed that the four tanks are positioned close to the central axis of the spacecraft and run from the bottom plane to the top plane. Because the fuel level will be similar for all tanks and the tanks are close to the central axis, the fuel is modelled to be in one big cylinder of 1.480 m^3 running

[5] http://www.space-propulsion.com/

from top to bottom of the spacecraft (533 mm radius). Because of the fuel in the centre, the structure is now a thick-walled hollow cylinder.

The structural density can be obtained as follows, where m_s is the structural mass, h_s is the spacecraft's height, r_s is the structure radius and r_t is the tank radius:

$$\rho_s = \frac{m_s}{h_s \pi (r_s^2 - r_t^2)} \tag{2.1}$$

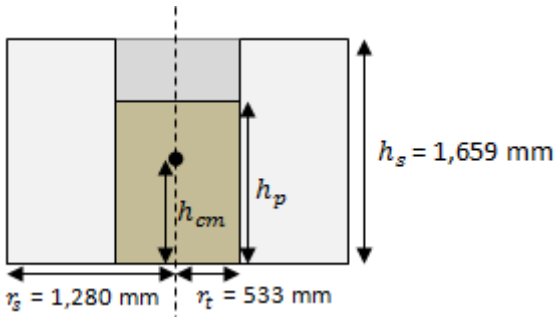


Figure 2.5: Spacecraft mass distribution model.

To get the moments of inertia for a cylinder with homogeneous density, Meriam and Kraige (2008) give the following relations:

$$I_{xx} = I_{yy} = \frac{1}{4} m r^2 + \frac{1}{12} m h^2 \tag{2.2a}$$

$$I_{zz} = \frac{1}{2} m r^2 \tag{2.2b}$$

However, since the structure is hollow to accommodate the propellant, a central piece is removed from the cylinder to create a 1.480 m^3 tank:

$$I_{s,xx} = I_{s,yy} = \frac{1}{4} \rho_s \pi h_s \left(r_s^4 - r_t^4 + \frac{1}{3} h_s^2 (r_s^2 - r_t^2) \right) \tag{2.3a}$$

$$I_{s,zz} = \frac{1}{2} \rho_s \pi h_s (r_s^4 - r_t^4) \tag{2.3b}$$

Note that the above moments of inertia are for the structure only. Because the lander will be under constant thrust during its descent, the propellant is expected to rest at the ground plane at all times. And because MMH and N_2O_4 are mixed, a combined density of ρ_p = 1,159 kg/m^3 is

used. Given the total propellant mass m_p, the height of the propellant from the ground plane is obtained as follows:

$$h_p = \frac{m_p}{\rho_p \pi r_t^2} \tag{2.4}$$

Using the structural height h_s and the propellant height h_p, the centre of mass with respect to the bottom plane is obtained, where m is the total mass:

$$h_{cm} = \frac{m_s h_s + m_p h_p}{2m} \tag{2.5}$$

The moments of inertia of the propellant column ($I_{p,xx}$, $I_{p,yy}$ and $I_{p,zz}$) are directly given by Equation 2.2. As final step for the mass distribution, the structural moments of inertia and the propellant moments of inertia are combined to obtain the total moment of inertia with respect to the centre of mass of the spacecraft:

$$I_{xx} = I_{s,xx} + m_s(h_s/2 - h_{cm})^2 + I_{p,xx} + m_p(h_p/2 - h_{cm})^2 \tag{2.6a}$$

$$I_{yy} = I_{s,yy} + m_s(h_s/2 - h_{cm})^2 + I_{p,yy} + m_p(h_p/2 - h_{cm})^2 \tag{2.6b}$$

$$I_{zz} = I_{s,zz} + I_{p,zz} \tag{2.6c}$$

Table 2.1 shows the results returned by the above mass-distribution equations for different fuel levels. An important note here is that at PDI, the ESA Lunar Lander only carries approximately 50% of its fuel, because it is injected into a trans-lunar orbit and needs to circularize its own orbit upon arrival around the Moon (Fisackerly et al., 2011).

Table 2.1: Mass distributions for different fuel levels.

Fuel Level	**100%**	**50%**	**25%**	**0%**
m [kg]	2,615	1,757	1,329	900
h_{cm} [mm]	830	627	629	830
I_{xx} [kg m^2]	1154	825	788	639
I_{yy} [kg m^2]	1154	825	788	639
I_{zz} [kg m^2]	1109	987	926	865

Thrusters

Kerr et al. (2013) showed that the Lunar Lander is equipped with five 500 N main engines, six 220 N assist engines and sixteen 22 N Reaction Control System (RCS) thrusters. For the former two types of engine, Riehle et al. (2012) specify the specific impulses at >323 s and >280 s, respectively. Because the main engines are non-throttleable, the lower thrust limit is 2,500 N. The 220 N assist engines can be throttled, using pulse modulation, to a minimum of 0 N. In ESA's mission design, the 500 N main engines would not be throttled, but powered off sequentially. However, Schulte (2004) notes that the 400 N predecessor engine can be pulsed to a lower thrust bound of 83.3 N. To make the design more flexible, this lower thrust limit is assumed for the 500 N main engines. This sets the lower thrust limit of the overall spacecraft to 416.5 N.

The main engines and the assist engines are all mounted on the bottom plane of the spacecraft. The engines are modelled as one big engine aligned with the centre of mass. The modelled engine has the accumulated thrust of the individual main and assist engines. Therefore, the modelled engine is throttleable from 416.5 N to 3,820 N with a specific impulse of 308 s.

The RCS thrusters are grouped by two. Four groups are evenly distributed over the upper and lower edges of the cylindrical body. The individual RCS thrusters are angled 45° with respect to the top or bottom plane, as shown by Figure 2.6, to provide control over all three axes of rotation.

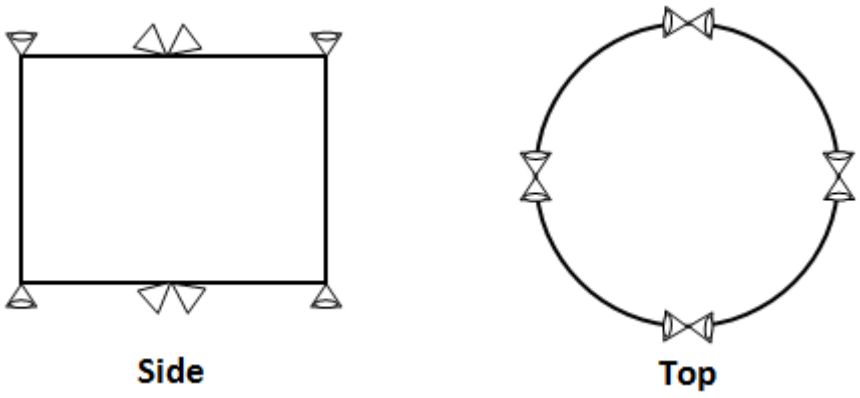


Figure 2.6: RCS thruster configuration.

Attitude Control Torques

The RCS thrusters are the only attitude-control actuators mentioned for ESA's Lunar Lander in literature. Their configuration has been discussed in the previous paragraph. This paragraph analyses the achievable torque using the described configuration, shown in Figure 2.6.

For generating a torque around the vertical axis of the spacecraft, 4 RCS thrusters fire on both the top and bottom planes, as shown by Figure 2.7. Because the thrusters are angled at 45° with respect to the top and bottom planes, only $\frac{1}{2}\sqrt{2}$ of the 22 N of thrust is used to generate torque (the other part is in the vertical axis and is cancelled by the thrusters of the opposite plane). Using the earlier assumed spacecraft diameter of 2,560 mm, each thruster has a moment arm of 1,280 mm. The combination of the 8 firing thrusters, yields a maximum torque of 159.3 Nm. Using the mass properties of 50% fuel level (at PDI) from Table 2.1, yields an achievable angular acceleration around the vertical axis of 0.161 rad/s^2. Over time, this approaches 0.184 rad/s^2 when going to 0% fuel.

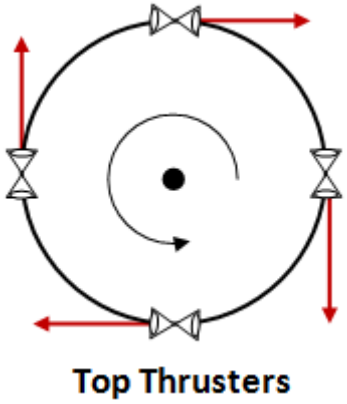


Figure 2.7: RCS thruster firing configuration for rotation around the vertical axis.

Generating torques around the x- or y-axis involves a more complicated thruster firing configuration. For a counter-clockwise rotation, as in Figure 2.8, both thrusters of the top-left group and bottom-right group fire (from the perspective of Figure 2.8). Of the centre groups, only one of the two thrusters fire, as depicted in Figure 2.8 as well. Also for this case, only $\frac{1}{2}\sqrt{2}$ of the thrust is utilized due to the 45° angle of the thrusters. The four top-left and bottom-right thrusters have an arm of 1,280 mm (half the diameter of the spacecraft), yielding 79.6 Nm. Assuming the centre of mass is in the centre of the tank, the four thrusters on the centre line of Figure 2.8, have a moment arm of 829.5 mm (half of height). Combined, these thrusters produce a torque of 51.6 Nm. Note that this torque does not change when the centre of mass shifts down, because the reduction in arm for the lower thrusters, equals the increase in arm for the upper thrusters, thus maintaining a constant total torque. This analysis shows that a total torque of 131.2 Nm can be generated around x- or y-axis. Using the mass properties of 50% fuel level (at PDI) from Table 2.1 again, yields an achievable angular acceleration of 0.159 rad/s^2. This approaches 0.205 rad/s^2 when going to 0% fuel.

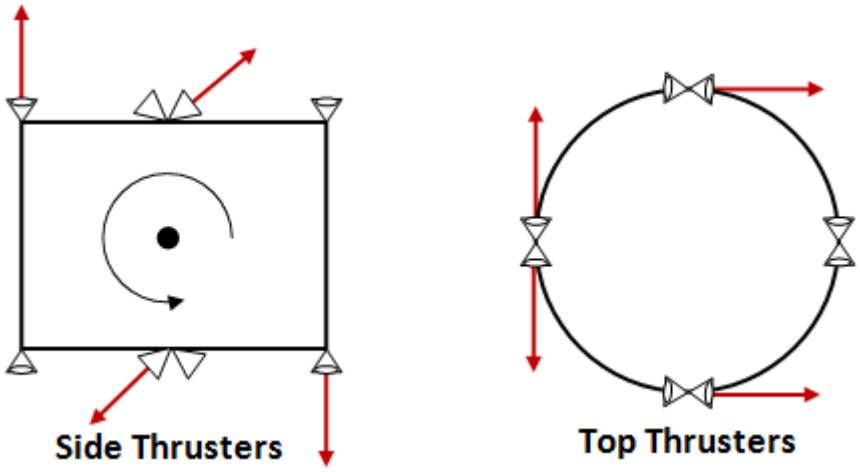


Figure 2.8: RCS thruster firing configuration for rotation around a horizontal axis.

Table 2.2: Achievable attitude control torques.

	Max. Torque [Nm]	50% Fuel $\dot{\omega}_{max}$ [rad/s^2]	0% Fuel $\dot{\omega}_{max}$ [rad/s^2]
x-axis	131.2	0.159	0.205
y-axis	131.2	0.159	0.205
z-axis	159.3	0.161	0.184

Sensor Configuration

The configuration of navigational sensors can have a large impact on the operations of the spacecraft. There are two main configurations: either the sensors are rigidly fixed to the spacecraft's body (ESA), or the sensors can be rotated with respect to the spacecraft's body using a gimbal (NASA). Fixing the sensors simplifies the structural design, but makes the landing trajectory more constrained. Using a gimbal gives more freedom to the descent trajectory, but complicates the sensor's mechanical design.

Because this study concerns the landing performance under highly constrained situations, a sensor suite is chosen that is fixed with respect to the spacecraft's body. This results in a highly constraint descent trajectory that Convex Guidance should be able to solve. This is considered a good test for the system performance.

The orientation of the sensors with respect to the spacecraft's body is a sensitive parameter that needs to be optimized for each specific mission. As a reference, Gerth (2014) found an angle of 2° (with respect to the body-down vector) to be optimal for a mission based on ESA's

Lunar Lander.

Flight Profile

ESA's Lunar Lander is injected into a trans-Lunar orbit. Arrived at the Moon, the Lunar Lander circularizes its orbit to obtain a 100 x 100 km LLO. From this orbit the descent orbit is initiated, lowering the spacecraft to an altitude of 10 km, through a 10 x 100 km Hogmann transfer.

Perilune (at 10 km altitude) of the Hohmann transfer marks the Powered Descent Initiation (PDI), where most of the spacecraft's velocity is cancelled during the Main Braking Phase (MBP). Once the spacecraft is only a couple of kilometres away from the landing site, the Approach Phase (AP) starts, where the spacecraft scans the landing site from possible hazards. In case of a hazard detection, a new landing site in the area can be targeted instead.

This study start at PDI, the beginning of the MBP. The initial conditions at PDI influence the propellant consumption during the flight. Therefore, it is important to not only optimise the trajectory, but also to optimise the initial conditions. Because Gerth (2014) also studied a Lunar landing with convex guidance, with ESA's Lunar Lander as reference vehicle, his findings for the optimal initial conditions are used.

Gerth (2014) found the optimal conditions for the Powered Descent Initiation (PDI) to be 630 km downrange from the landing site and at an altitude of 10 km. The velocity corresponds with a 10x100 km orbit, where the PDI is located at perilune. The optimal time of flight for the MBP is 660 seconds for reaching the final conditions, which are also the initial conditions for the AP.

The transition from the MBP to the AP happens at the Approach Gate (AG). Gerth (2014) gives the optimal AG as:

$$\mathbf{r}_{AG} = \begin{bmatrix} 0 \\ -1265 \\ 1625 \end{bmatrix}, \quad \mathbf{v}_{AG} = \begin{bmatrix} 0 \\ 67 \\ -60 \end{bmatrix} \tag{2.7}$$

Where $\mathbf{r}_{AG}$ is in metres and $\mathbf{v}_{AG}$ is in metres per second. Furthermore, note that $\mathbf{r}_{AG}$ and $\mathbf{v}_{AG}$ are expressed in the landing site frame, while the initial state at the PDI are expressed in a spherical frame.

The end of the AP is referred to as the Terminal Gate (TG), from which the spacecraft continues its decent vertically, as an open loop, until touch down. The conditions at TG are defined as

follows, in the landing site frame:

$$\mathbf{r}_{TG} = \begin{bmatrix} 0 \\ 0 \\ 15 \end{bmatrix}, \quad \mathbf{v}_{TG} = \begin{bmatrix} 0 \\ 0 \\ -1.5 \end{bmatrix} \tag{2.8}$$

Gerth (2014) finds the optimal time of flight for the AP to be 52 seconds, making the total flight from the PDI to the TG last 712 seconds.

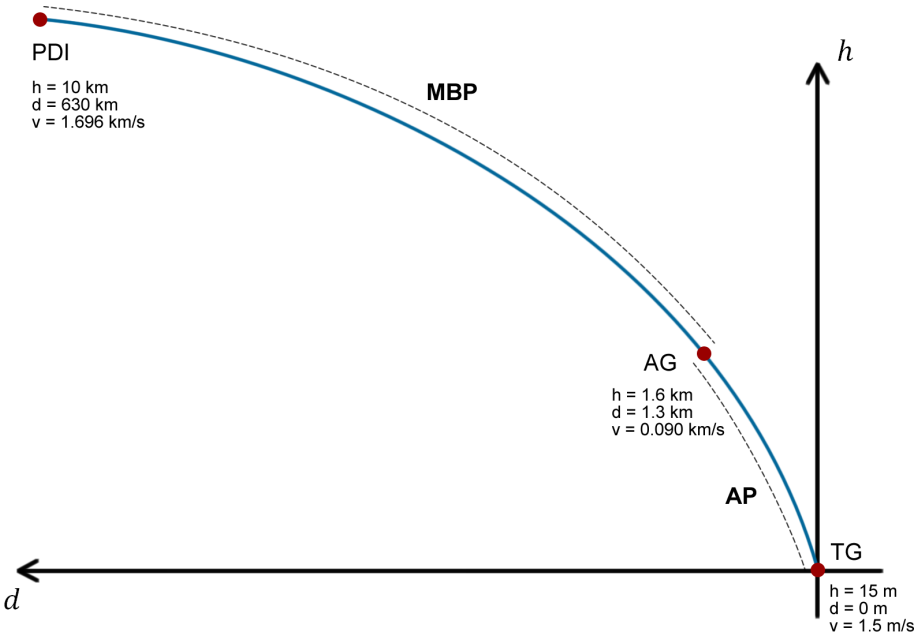


Figure 2.9: Landing phases.

2.5 Mission Requirements

For their Lunar Lander, ESA officially sets a landing precision requirement of 200 metres inertially and 20 metres relative to the terrain (Kerr et al., 2013). NASA defines requirements in the same order for their Lunar missions (Ely et al., 2012). However, ESA and NASA might be conservative in the figures they publish. For example, Gerth (2014) defines much more strict requirements of 1.5 metre horizontal and 1.0 metre vertical accuracy, which he obtained from ESA inside-contacts. Therefore, Gerth's requirements are also used for this study to explore the limits of convex guidance.

Kerr et al. (2013) further define the maximum velocity error to be within 1 m/s, the attitude error to be within 2° and angular-rate error to be within 2.5°/s. Table 2.3 gives an overview of all allowed errors at touchdown.

Table 2.3: Maximum allowed errors at touchdown in the landing-site frame.

	Position	**Velocity**	**Attitude**	**Angular rate**
x	1.5 m	1.0 m/s	2°	2.5°/s
y	1.5 m	1.0 m/s	2°	2.5°/s
z	1.0 m	1.0 m/s	2°	2.5°/s

3 Flight Mechanics

3.1 Coordinate Systems

Considered coordinate systems in this study are the Cartesian and spherical coordinate systems, both shown in Figure 3.1. For the spherical system, τ defines the longitude, δ defines the latitude and r defines the radius. It is important to note that for $\delta = \pm 90°$, the value of τ becomes undefined. The same singularity occurs when using spherical coordinates for the expression of velocity. A spherical coordinate system is well suited for when the spacecraft has a global range, while a Cartesian system is better suited for local activities, close to the landing site.

Remember, this study only concerns the landing phase from the Powered Descent Initiation (PDI) onwards, which is about 650 km downrange from the landing site, as discussed in Section 2.4. Compared to the Moon's circumference, this is less than 6%. Although curvature still has a significant effect, this is considered relatively local and hence, the Cartesian expression is used for spacecraft's position and velocity in this study.

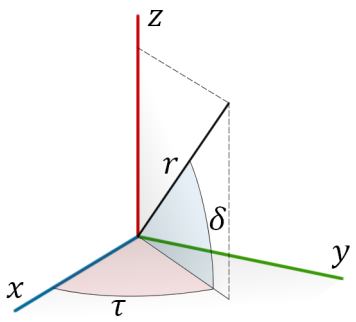


Figure 3.1: Spherical system τ-δ-r expressed in the Cartesian x-y-z system.

3.2 Reference Frames

All frames considered in this study are right-handed reference frames, meaning that the following relation holds:

$$\mathbf{z} = \mathbf{x} \times \mathbf{y} \tag{3.1}$$

$\mathcal{I}$ – Lunar Inertial Frame The Lunar inertial frame $\mathcal{I}$ has its origin at the Moons centre of mass. The $\mathbf{xy}$-plane coincides with the Moon's equator and the $\mathbf{z}_{\mathcal{I}}$-axis lines up with the Moon's rotational axis. This frame does not corotate with the Moon, but the direction of the $\mathbf{x}_{\mathcal{I}}$-axis is predefined at a certain epoch instead. For this study, the direction of the $\mathbf{x}_{\mathcal{I}}$-axis is defined as the direction of the Moon's prime meridian (the meridian that faces Earth) at the time of the PDI.

$\mathcal{C}$ – Lunar Corotating Frame The Lunar corotating frame $\mathcal{C}$ is defined in a similar fashion as the $\mathcal{I}$-frame. The difference is that the $\mathcal{C}$-frame rotates around the $z_{\mathcal{C}}$-axis with respect to the $\mathcal{I}$-frame with the angular rate of the Moon $\omega_{\mathbb{C}}$, so that the $x_{\mathcal{C}}$-axis stays located at the Moon's prime meridian. As defined by the discussion on the $\mathcal{I}$-frame, the $\mathcal{C}$-frame coincides with the $\mathcal{I}$-frame at time t_0.

$\mathcal{L}$ – Landing Site Frame The landing site Frame $\mathcal{L}$ has its origin at the targeted landing site, at longitude $\tau_{\mathcal{L}}$ and latitude $\delta_{\mathcal{L}}$. The $\mathbf{xy}$-plane coincides with the local horizontal plane of the landing site, the $\mathbf{x}_{\mathcal{L}}$-axis points south, the $\mathbf{y}_{\mathcal{L}}$-axis points east and the $\mathbf{z}_{\mathcal{L}}$-axis points zenith (up), as shown in Figure 3.3.

$\mathcal{V}$ – Local Vertical Frame The local vertical frame $\mathcal{V}$ is very similar to the Landing Site Frame $\mathcal{L}$. The xy-plane coincides with the local horizontal plane and the $\mathbf{z}_{\mathcal{V}}$-axis points zenith. The difference is that the origin is located at the centre of mass of the spacecraft, instead of the landing site. Therefore, this frame can be pictured very similarly as Figure 3.3, but then for the current longitude, latitude and altitude of the spacecraft. Note that conventionally, z is nadir instead of zenith. However, because the implementation of convex guidance by Açikmeşe and Ploen (2007) defines the positive direct to be up instead of down, the $\mathcal{V}$-frame (and the $\mathcal{L}$-frame) are defined to correspond with this definition.

$\mathcal{G}$ **– Geometric Frame** The geometric frame $\mathcal{G}$ is fixed to the body of the spacecraft. The xy-plane coincides with the bottom plane of the cylinder representing the spacecraft. The $\mathbf{z}_{\mathcal{G}}$-axis points up. When the spacecraft lands, the axes of the $\mathcal{G}$-frame coincide with those of the $\mathcal{L}$-frame. This means that it is the goal to land the spacecraft to get the $\mathcal{G}$-frame and $\mathcal{L}$-frame origins to coincide (except from an offset in the z-axis) with an as little as possible rotation between the two frames. This assumes that at touchdown, the rotation around the vertical axis of the spacecraft matters.

$\mathcal{B}$ **– Body Fixed Frame** The body fixed frame $\mathcal{B}$ has the same orientation as the $\mathcal{G}$-frame, but its origin is located at the current centre of mass of the spacecraft instead of the bottom plane of the spacecraft. Effectively, the origin of the $\mathcal{G}$-frame is shifted in the $\mathbf{z}_{\mathcal{G}}$ direction over distance h_{cm}, as discussed in Section 2.4, to construct the $\mathcal{B}$-frame.

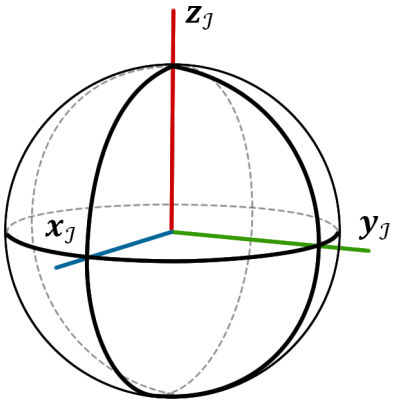


Figure 3.2: Lunar Inertial Frame with the $\mathbf{xy}$-plane in the equatorial plane and the $\mathbf{z}_{\mathcal{I}}$-axis aligning with the Moon's rotation axis.

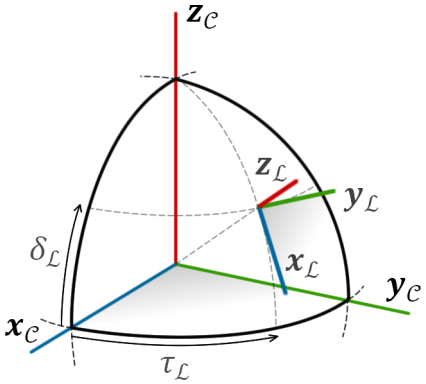


Figure 3.3: Landing Site Frame on the surface of the Moon for longitude $\tau_{\mathcal{L}}$ and latitude $\delta_{\mathcal{L}}$.

3.3 Attitude Representations

For the representation of the spacecraft's attitude, different expressions are considered. Two commonly used parameter sets are Euler angles and quaternions. Where Euler angles are based on roll ϕ, pitch θ and yaw ψ rotations, quaternions are based on an eigenaxis rotation ($\mathbf{e}$, θ), giving parameters q_1, q_2, q_3 and q_4.

For transforming vectors between different frames, the direction cosine matrices of Equations 3.2 and 3.3 are used for Euler angles and quaternions, respectively (Shuster, 1993). For Equation 3.2, c represents the cosine function and s represents the sine function.

$$\mathbf{R}_{312}(\phi,\theta,\psi) = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\phi - \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi & -\mathrm{s}\psi\mathrm{c}\theta \\ -\mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi & \mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi - \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi & \mathrm{c}\psi\mathrm{c}\theta \end{bmatrix} \tag{3.2}$$

$$\mathbf{R}(\bar{\mathbf{q}}) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_2q_1-q_3q_4) & 1-2(q_1^2+q_3^1) & 2(q_2q_3+q_1q_4) \\ 2(q_3q_1+q_2q_4) & 2(q_3q_2-q_1q_4) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{3.3}$$

From Equations 3.2 and 3.3 it can be seen that using Euler angles require many sine and cosine evaluations, while using quaternions only requires multiplication and addition of the quaternion components. Furthermore, Euler angles suffer from singularities in the dynamic equations. These singularities can be circumvented by the use of shadow sets. Quaternions have no such singularities, but quaternions do have an norm constraint. Mainly from the computational point of view, quaternions are selected over Euler angles for this study.

Quaternions

A quaternion of rotation is based on eigenaxis rotations. Shuster (1993) gives the definition of a quaternion of rotation:

$$\bar{\mathbf{q}}(\mathbf{e},\theta) = \begin{bmatrix} \mathbf{q}(\mathbf{e},\theta) \\ q_4(\theta) \end{bmatrix} = \begin{bmatrix} \mathbf{e}\sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{3.4}$$

$$\bar{\mathbf{q}}^T\bar{\mathbf{q}} = 1 \tag{3.5}$$

Figure 3.4 gives a visual representation of how frame 1 is rotated to the second frame. Here $\mathbf{e} = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^T$ is the eigenaxis around which the frame is rotated and θ is the angle over which the frame is rotated. The eigenaxis $\mathbf{e}$ is a unit column vector.

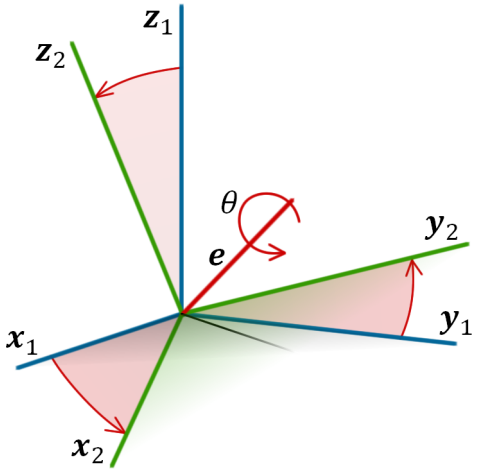


Figure 3.4: Transformation of fame 1 to frame 2 by rotation around $\mathbf{e}$ over an angle of θ.

To transform a column vector between two frames, it is pre-multiplied with the Direction Cosine Matrix (DCM), $\mathbf{R}$. This transformation matrix is constructed from the quaternion as follows (Shuster, 1993):

$$\mathbf{R}(\bar{\mathbf{q}}) = (q_4^2 - \mathbf{q}^T\mathbf{q})\mathbf{I}_3 + 2\mathbf{q}\mathbf{q}^T - 2q_4[\mathbf{q}\times] \tag{3.6a}$$

$$[\mathbf{q}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{3.6b}$$

Or:

$$\mathbf{R}(\bar{\mathbf{q}}) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_2q_1-q_3q_4) & 1-2(q_1^2+q_3^1) & 2(q_2q_3+q_1q_4) \\ 2(q_3q_1+q_2q_4) & 2(q_3q_2-q_1q_4) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{3.7}$$

Subsequent rotations can be achieved by multiplying DCMs as shown in Equation 3.8. Here,

$\bar{\mathbf{q}}_C$ is the sequential rotation of $\bar{\mathbf{q}}_A$ followed by $\bar{\mathbf{q}}_B$.

$$\mathbf{R}(\bar{\mathbf{q}}_C) = \mathbf{R}(\bar{\mathbf{q}}_B)\mathbf{R}(\bar{\mathbf{q}}_A) \tag{3.8}$$

This sequential rotation can also be expressed as quaternion multiplication $\bar{\mathbf{q}}_C = \bar{\mathbf{q}}_B \otimes \bar{\mathbf{q}}_A$, such that:

$$\mathbf{R}(\bar{\mathbf{q}}_B \otimes \bar{\mathbf{q}}_A) = \mathbf{R}(\bar{\mathbf{q}}_B)\mathbf{R}(\bar{\mathbf{q}}_A) \tag{3.9}$$

The practical operation for this quaternion multiplication is given by Shuster (1993) as follows:

$$\bar{\mathbf{q}}_C = \bar{\mathbf{q}}_B \otimes \bar{\mathbf{q}}_A \tag{3.10a}$$

$$\bar{\mathbf{q}}_C = \begin{bmatrix} \mathbf{q}_C \\ q_{4_C} \end{bmatrix} = \begin{bmatrix} q_{4_A}\mathbf{q}_B + q_{4_B}\mathbf{q}_A + \mathbf{q}_A \times \mathbf{q}_B \\ q_{4_A}q_{4_B} - \mathbf{q}_A^T\mathbf{q}_B \end{bmatrix} \tag{3.10b}$$

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}_C = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}_B \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}_A \tag{3.10c}$$

3.4 Frame Transformations

For the definition of the quaternions of rotation between the reference frames defined in Section 3.2, sequential rotations are used around the x-, y- and z-axes. For this purpose, $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are defined as unit vectors in the x, y and z directions, respectively. This section continues to define several quaternions of rotation between the different frames.

$\mathcal{I}$–frame to $\mathcal{C}$–frame The $\mathcal{C}$-frame rotates over its z-axis with respect to the $\mathcal{I}$-frame. The angular rate is defined as the angular rate of the Moon $\omega_{\mathbb{C}}$. Therefore, the angle of rotation at time Δt with respect to the initial condition (where the $\mathcal{C}$-frame coincides with the $\mathcal{I}$-frame), equals to $\omega_{\mathbb{C}}\ \Delta t$. Because the $\mathcal{C}$-frame rotates over its z-axis, the eigen rotation for this transformation is eigenaxis $\mathbf{k}$ with an angle of $\omega_{\mathbb{C}}\ \Delta t$. Therefore, the quaternion of rotation from the

$\mathcal{I}$-frame to the $\mathcal{C}$-frame is defined as follows, where $\bar{\mathbf{q}}$ is the function to construct a quaternion as defined by Equation 3.4:

$$\bar{\mathbf{q}}_{\mathcal{CI}} = \bar{\mathbf{q}}\left(\mathbf{k}, -\omega_{\oplus}\,\Delta t\right) \tag{3.11}$$

$\mathcal{C}$–frame to $\mathcal{L}$–frame For a landing site at $\delta_{\mathcal{L}}$ latitude and $\tau_{\mathcal{L}}$ longitude, the $\mathcal{C}$ frame is rotated over its z-axis by angle $\tau_{\mathcal{L}}$ first, followed by a rotation over its y-axis by angle $\pi/2 - \delta_{\mathcal{L}}$. Therefore, transforming a vector from the $\mathcal{C}$-frame to the $\mathcal{L}$-frame takes the reversed order:

$$\bar{\mathbf{q}}_{\mathcal{LC}} = \bar{\mathbf{q}}\left(\mathbf{k}, -\tau_{\mathcal{L}}\right) \otimes \bar{\mathbf{q}}\left(\mathbf{j}, \delta_{\mathcal{L}} - \pi/2\right) \tag{3.12}$$

$\mathcal{C}$–frame to $\mathcal{V}$–frame The same relation holds for transforming the $\mathcal{C}$-frame to the $\mathcal{V}$-frame for a spacecraft flying at δ_s latitude and τ_s longitude:

$$\bar{\mathbf{q}}_{\mathcal{VC}} = \bar{\mathbf{q}}\left(\mathbf{k}, -\tau_{s}\right) \otimes \bar{\mathbf{q}}\left(\mathbf{j}, \delta_{s} - \pi/2\right) \tag{3.13}$$

$\mathcal{C}$–frame to $\mathcal{B}$–frame When the attitude of the spacecraft is given in terms of roll, pitch and yaw with respect to the local horizon, the attitude quaternion with respect to the $\mathcal{V}$-frame is obtained as follows. For this transformation, the yaw (ψ) $\rightarrow$ pitch (θ) $\rightarrow$ roll (ϕ) sequence is assumed.

$$\bar{\mathbf{q}}_{\mathcal{BV}} = \bar{\mathbf{q}}(\mathbf{i}, \phi) \otimes \bar{\mathbf{q}}(\mathbf{j}, \theta) \otimes \bar{\mathbf{q}}(\mathbf{k}, \psi) \tag{3.14}$$

Having the attitude of the spacecraft expressed with respect to the local vertical frame, the attitude can be expressed with respect to the inertial frame using a sequence of rotations:

$$\bar{\mathbf{q}}_{\mathcal{BI}} = \bar{\mathbf{q}}_{\mathcal{BV}} \otimes \bar{\mathbf{q}}_{\mathcal{VC}} \otimes \bar{\mathbf{q}}_{\mathcal{CI}} \tag{3.15}$$

Inverting an eigen rotation can philosophically either be performed by using the negative rotation or the negative eigenaxis. Mathematically however, both approaches lead to the expres-

sion as presented in Equation 3.16.

$$\bar{\mathbf{q}}_{\mathcal{BI}} = \begin{bmatrix} \mathbf{e}\sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix}, \quad \bar{\mathbf{q}}_{\mathcal{IB}} = \begin{bmatrix} -\mathbf{e}\sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{3.16}$$

When making use of the DCMs, the inverse rotation is given by the inverse of the matrix. However, because all rotations were performed around the orthogonal x-, y- and z-axes, the DCM is an orthogonal matrix. For such matrices, Lay (2012) gives the inverse as its transpose:

$$\mathbf{R}(\bar{\mathbf{q}}_{\mathcal{IB}}) = \mathbf{R}^{-1}(\bar{\mathbf{q}}_{\mathcal{BI}}) = \mathbf{R}^{T}(\bar{\mathbf{q}}_{\mathcal{BI}}) \tag{3.17}$$

3.5 Translational Equations of Motion

For the translational equations of motion, the selected reference frame has a large influence on the expression. The equations of motion are more intuitive in the $\mathcal{I}$-frame than they are in the $\mathcal{C}$-frame (due to Coriolis effects). Hence, it would be easier to understand the results and to debug the simulator if it were written in $\mathcal{I}$-frame. Furthermore, the additional terms for the equations of motion in the $\mathcal{C}$-frame can introduce nonlinearities. This can slow down the simulation, because a smaller time step might be needed during integration to assure a given accuracy. Therefore, the equations of motion are expressed in the $\mathcal{I}$-frame.

The well-known second law of motion by Newton reads:

$$\mathbf{F} = \frac{d(m\mathbf{v})}{dt} \tag{3.18}$$

Rewriting this equations yields the following, where $\mathbf{v}_e$ is the exhaust velocity of the thruster:

$$\mathbf{F} = \mathbf{v}_e \frac{dm}{dt} + m\frac{d\mathbf{v}}{dt} \tag{3.19}$$

The change of mass is multiplied with the exhaust velocity, because in the closed system the propellant is not lost, but in fact given the exhaust velocity. This component of the force is the thrust due to the spending of fuel. Turner (2004) gives the following definition, where $\dot{m}$ is the mass flow (of the propellant):

$$T = \dot{m}v_e \tag{3.20}$$

Substituting Equation 3.20 into Equation 3.19, yields an expression for the acceleration of the spacecraft in the $\mathcal{I}$-frame:

$$\mathbf{a} = \frac{\mathbf{F} - \mathbf{T}}{m} \tag{3.21}$$

For this equation, $\mathbf{F}$ are the external forces on the spacecraft and $\mathbf{T}$ is the thrust provided by the engines. The magnitude of the exhaust velocity v_e is also defined as the product of the standard gravitational acceleration g_0 and the specific impulse of the engine I_{sp} (Turner, 2004):

$$v_e = g_0 I_{sp} \tag{3.22}$$

Spherical Harmonics

For the of the modelling gravitational field, Wakker (2015) provides a model based on spherical harmonics in Equation 3.23, where $P_{n,m}$ are Legendre polynomials and their associated Legendre functions of the first kind. Furthermore, μ is the Moon's gravitational parameter, R is the Moon's reference radius, r is the distance of the spacecraft with respect to the centre of mass of the Moon, and δ and τ are the spacecraft's latitude and longitude, respectively.

$$U = -\frac{\mu}{r}\left[1 + \sum_{n=2}^{\infty}\sum_{m=0}^{n}\left(\frac{R}{r}\right)^n P_{n,m}(\sin\delta)\{C_{n,m}\cos(m\tau) + S_{n,m}\sin(m\tau)\}\right] \tag{3.23a}$$

$$P_{n,m}(x) = (1-x^2)^{m/2}\frac{d^m P_n(x)}{dx^m} \tag{3.23b}$$

$$P_n(x) = \frac{1}{(-2)^n n!}\frac{d^n}{dx^n}(1-x^2)^n \tag{3.23c}$$

When the C and S coefficients are normalized, the fully normalized Legendre Polynomials $\bar{P}_{n,m}$ need to be used in Equation 3.23a, as provided by Konopliv et al. (2013).

$$\bar{P}_{n,m} = \sqrt{\frac{2(n-m)!(2n+1)}{(n+m)!}}P_{n,m} \tag{3.24}$$

As Equation 3.23 shows, an infinite series needs to be used to obtain the true gravitational field. Practically however, it is not possible to use and infinite number of terms. Therefore,

degree n is limited to a number that yields a sufficient approximation of the gravity field. The degree up to which to gravity field needs to be simulated for this specific study is discussed in Section 6. When using degree and order zero ($n = m = 0$), a spherical gravity field is obtained:

$$U = \frac{-\mu}{r} \tag{3.25}$$

From the gravitational potential in Equation 3.23, the gravitational acceleration is found as follows:

$$\mathbf{f} = -\boldsymbol{\nabla} U = -\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{bmatrix} U \tag{3.26}$$

3rd Body Perturbations

Any celestial bodies - other than the Moon - nearby or massive enough, also exert a significant gravitational force on the spacecraft. For example, because of the proximity of the Earth, it perturbs the spacecraft by 10^{-5} m/s^2 (assuming $\mu_\oplus = 3.986 \cdot 10^{14}\ \mathrm{m}^3/\mathrm{s}^2$ (Wertz, 2001), $R = 1.736{\cdot}10^6\ \mathrm{m}$ and $r = 384.40{\cdot}10^6\ \mathrm{m}$ (Lissauer and de Pater, 2013)). Even though this is about 0.001% of the Moon's gravitational acceleration, a 10 minute integration of the perturbation yields a deviation in the order of metres.

When the Earth is assumed to be a point mass, and the position vector running from the spacecraft to the Earth is defined as $\mathbf{r}_\oplus$, then the gravitational acceleration experienced by the spacecraft due to the Earth is given by, where $\mu_\oplus$ is the gravitational parameter of the Earth:

$$\mathbf{f} = \frac{\mu_\oplus \mathbf{r}_\oplus}{\|\mathbf{r}_\oplus\|^3} \tag{3.27}$$

However, at the same time, the Moon also experiences an acceleration due to the gravity field from the Earth, where $\mathbf{R}$ is the position vector from the Moon to the Earth:

$$\mathbf{f} = \frac{\mu_\oplus \mathbf{R}}{\|\mathbf{R}\|^3} \tag{3.28}$$

The perturbing acceleration experienced by the spacecraft orbiting the Moon, is the difference between the accelerations that the spacecraft and the Moon experience, due to the gravita-

tional pull of the Earth (Wakker, 2015):

$$\mathbf{f} = \mu_{\oplus} \left(\frac{\mathbf{r}_{\oplus}}{\|\mathbf{r}_{\oplus}\|^3} - \frac{\mathbf{R}}{\|\mathbf{R}\|^3} \right) \tag{3.29}$$

To fully express the perturbing acceleration from the perspective of a Moon-central frame, $\mathbf{r}_{\oplus}$ can be expressed as $\mathbf{R} - \mathbf{r}$, where $\mathbf{r}$ is the position of the spacecraft with respect to the Moon:

$$\mathbf{f} = \mu_{\oplus} \left(\frac{\mathbf{R} - \mathbf{r}}{\|\mathbf{R} - \mathbf{r}\|^3} - \frac{\mathbf{R}}{\|\mathbf{R}\|^3} \right) \tag{3.30}$$

3.6 Rotational Equations of Motion

Angular-rate measurements are assumed to be provided by gyroscopes around the x-, y- and z-axes of the body frame, represented by $\boldsymbol{\omega}$. Equation 3.31 shows the rotational dynamics of a spacecraft in its body frame as provided by Wie (1998). For this expression, $\mathbf{I}$ is the inertial matrix, $\boldsymbol{\omega}$ is the angular rate and $\mathbf{M}$ is the external moment or torque acting on the spacecraft.

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = \mathbf{M} \tag{3.31}$$

Attitude measurements by star trackers are with respect to inertial space. Therefore, the attitude quaternion $\bar{\mathbf{q}}$ is defined as the spacecraft's attitude in inertial space. To transform the angular rates in the body frame to a time derivative of the attitude quaternion in inertial space, Wie (1998) provides the kinematic equations of Equation 3.32.

$$\dot{\bar{\mathbf{q}}} = \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{q}_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(q_4\boldsymbol{\omega} + \mathbf{q} \times \boldsymbol{\omega}) \\ -\frac{1}{2}\boldsymbol{\omega}^T\mathbf{q} \end{bmatrix} \tag{3.32a}$$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} \tag{3.32b}$$

4 Guidance

Traditionally, multiple guidance algorithms are used for different phases of the descent, which makes the overall guidance system complex. Current space missions increasingly need higher landing accuracies, hazard detection and hazard avoidance. A new and promising guidance algorithm, based on convex optimization, was developed by Açikmeşe and Ploen (2005). This technique guarantees to converge to the global optimum, even when the solution space is highly constrained. Therefore, convex guidance is very well suited for pinpoint landings in hazardous environments. The fact that it guarantees to return the global optimum, allows the spacecraft to re-evaluate the optimal trajectory on the fly, when a new hazard has been identified.

As Figure 4.1 shows, this chapter continues in Section 4.1 with a general discussion on convex guidance as developed by Açikmeşe and Ploen (2007). This is followed by the proposed additions to convex guidance by Gerth (2014), in Section 4.2. Subsequently, the problem is written in the conical form in Section 4.3, to improve the solving speed of the algorithm.

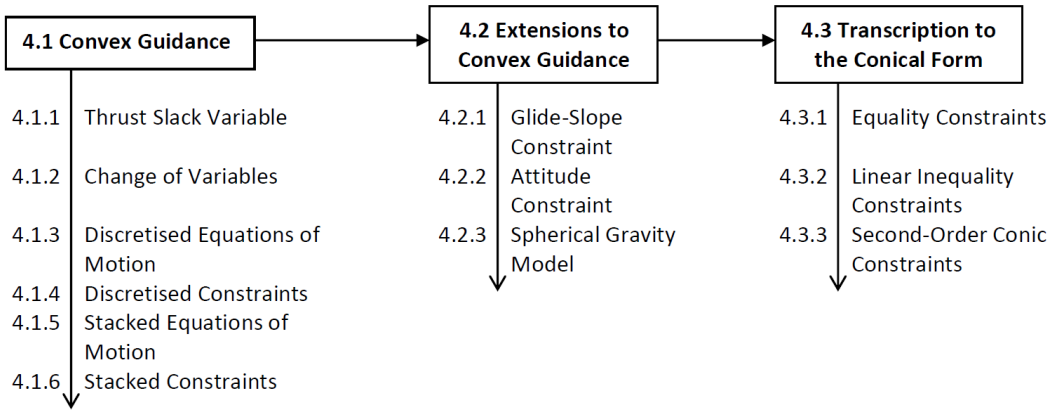


Figure 4.1: Chapter overview.

4.1 Convex Guidance

The problem statement from Table 4.1 shows the basis from which Açikmeşe and Ploen (2007) derive convex guidance. The problem seeks to minimize the propellant used. However, since a constant linear relation is assumed between the mass flow and thrust, the magnitude of the thrust-vector $\mathbf{T}$ is integrated over the time of flight t_f for the cost function, as shown by Equation 4.1.

$$\int_0^{t_f} \sqrt{\mathbf{T}(t)^T \mathbf{T}(t)} dt \tag{4.1}$$

While minimizing the cost function, the system needs to conform to the equations of motion and the relation of the mass flow with respect to the thrust, as given by Equations 4.2 and 4.3, respectively.

$$\ddot{\mathbf{r}} = \mathbf{g} + \mathbf{T}(t)/m(t) \tag{4.2}$$

$$\dot{m}(t) = -a \left\| \mathbf{T}(t) \right\| \tag{4.3}$$

For these equations, $\ddot{\mathbf{r}}$ is the spacecraft's acceleration, $\mathbf{g}$ is the gravitational acceleration, m is the spacecraft's mass, $\dot{m}$ is the mass flow and a is the thrust to mass flow ratio. The ratio of a is a function of the specific impulse of the engine I_{sp} and the standard gravitational acceleration[1] g_0:

$$a = \frac{1}{g_0 I_{sp}} \tag{4.4}$$

The magnitude of the thrust-vector is limited by a lower and upper thrust bound, T_l and T_h. Note that the lower thrust bound T_l is a non-zero value. This is a consequence of the assumption that the main engine cannot be restarted, but only throttled down to a specific flame-out point.

$$0 < T_l \leq \left\| \mathbf{T}(t) \right\| \leq T_h \tag{4.5}$$

[1] Because the engines are tested at Earth, Earth's standard gravitatonal acceleration is used: $g_0 = 9.80665 \text{ m/s}^2$

Next to the equality and inequality constraints mentioned above, initial boundary conditions are imposed on the spacecraft's mass, position and velocity (Equation 4.6), and final boundary conditions are imposed on the spacecraft's position and velocity (Equation 4.7):

$$m(0) = m_0, \quad \mathbf{r}(0) = \mathbf{r}_0, \quad \dot{\mathbf{r}}(0) = \dot{\mathbf{r}}_0 \tag{4.6}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f, \quad \dot{\mathbf{r}}(t_f) = \dot{\mathbf{r}}_f \tag{4.7}$$

The equations discussed above, describe the optimization problem that needs to be solved by the spacecraft's guidance algorithm. Table 4.1 gives an overview of the obtained guidance problem.

Table 4.1: Original continuous guidance problem. Source: Gerth (2014).

Minimize	$\int_0^{t_f} \sqrt{\mathbf{T}(t)^T \mathbf{T}(t)} dt$	*Cost function*
Subject to	$\ddot{\mathbf{r}}(t) = \mathbf{g}(t) + \mathbf{T}(t)/m(t)$	*Equations of motion*
	$\dot{m}(t) = -a \left\|\mathbf{T}(t)\right\|$	*Mass flow relation to thrust*
	$0 < T_l \leq \left\|\mathbf{T}(t)\right\| \leq T_h$	*Thrust constraints*
	$m(0) = m_0, \quad \mathbf{r}(0) = \mathbf{r}_0, \quad \dot{\mathbf{r}}(0) = \dot{\mathbf{r}}_0$	*Initial conditions*
	$\mathbf{r}(t_f) = \mathbf{r}_f, \quad \dot{\mathbf{r}}(t_f) = \dot{\mathbf{r}}_f$	*Final conditions*

4.1.1 Thrust Slack Variable

The lower thrust limit T_l causes the problem to be non-convex. Therefore the problem is convexified by introducing a scalar slack variable Γ (Açikmeşe and Ploen, 2007). The equation for the thrust limits (repeated in Equation 4.8) is replaced by the combination of Equations 4.9a and 4.9b.

$$0 < T_l \leq \|\mathbf{T}\| \leq T_h \tag{4.8}$$

$$0 < T_l \leq \Gamma \leq T_h \tag{4.9a}$$

$$\|\mathbf{T}\| \leq \Gamma \tag{4.9b}$$

Although the control space is relaxed, Açikmeşe and Ploen (2007) prove that the solution to this problem is also the solution to the initial problem. From the same analysis, Açikmeşe and Ploen (2007) show that a max-min-max solution is to be expected from the guidance problem.

4.1.2 Change of Variables

New variables are defined to cast the control forces into control accelerations, to linearise the equations of motion. Using this method, the problem can be time-discretised for the convex solver.

$$\sigma(t) = \frac{\Gamma(t)}{m(t)} \tag{4.10}$$

$$\boldsymbol{\tau}(t) = \frac{\mathbf{T}(t)}{m(t)} \tag{4.11}$$

The definition of σ also causes the cost function to change to the following:

$$\min \int_0^{t_f} \sigma(t) dt \tag{4.12}$$

Note that the cost function now represents the required ΔV for the landing. Subsequently, z is defined as the natural logarithm of the spacecraft's mass:

$$z(t) = \ln m(t) \tag{4.13}$$

Which yields (from Equation 4.3):

$$\dot{z}(t) = -a\sigma(t) \tag{4.14}$$

And allows the limits of the slacked thrust variable from Equation 4.9a to be rewritten as:

$$T_l e^{-z(t)} \leq \sigma(t) \leq T_h e^{-z(t)} \tag{4.15}$$

The problem with the above equation is that it is not linear. Therefore, Taylor Series Expansions are used to linearise the thrust limits:

$$\mu_1\{1 - [z - z_1]\} \leq \sigma \leq \mu_2\{1 - [z - z_2]\} \tag{4.16}$$

Where:

$$\mu_1(t) = T_l e^{-z_1(t)} \tag{4.17a}$$

$$\mu_2(t) = T_h e^{-z_2(t)} \tag{4.17b}$$

$$z_1(t) = \ln(m_0 - \alpha T_l t) \tag{4.18a}$$

$$z_2(t) = \ln(m_0 - \alpha T_h t) \tag{4.18b}$$

Note that a first-order approximation is used for both bounds of the thrust limit, while Açikmeşe and Ploen (2007) use a second-order approximation for the lower bound. Experimentation showed that using a first order approximation makes practically no difference compared to using a second order approximation, as shown in Section 5. However, using the first-order approximation greatly eases the transcription of the problem to the conical form, discussed in Section 4.3.

The magnitude of acceleration $\boldsymbol{\tau}$ is limited by the slack variable σ. Because the magnitude of a three dimensional vector is compared to a scalar, this constraint forms a four dimensional Second-Order Cone Problem (SOCP).

$$\|\boldsymbol{\tau}(t)\| \leq \sigma(t) \tag{4.19}$$

To help guide the solver to the optimal solution, Açikmeşe and Ploen (2007) introduce limits on the mass of the spacecraft:

$$z_1(t) \leq z(t) \leq z_2(t) \tag{4.20}$$

This auxiliary constraint is redundant to the thrust constraint. Theoretically, the mass constraint can improve the convergence rate of the solver. However, in practise it was found that it slows down the solving process, without improving the results. Therefore, this constraint is not used in this study.

Table 4.2 gives an overview of the guidance problem as obtained thus far.

Table 4.2: Changed guidance problem.

Minimize	$\int_0^{t_f} \sigma(t)dt$	*Cost function*
Subject to	$\ddot{\mathbf{r}}(t) = \mathbf{g}(t) + \boldsymbol{\tau}(t)$	*Equations of motion*
	$\dot{z}(t) = -a\sigma(t)$	*Mass flow relation to thrust*
	$\|\boldsymbol{\tau}(t)\| \leq \sigma(t)$	*Thrust constraint*
	$\mu_1\{1 - [z - z_1]\} \leq \sigma(t) \leq \mu_2\{1 - [z - z_2]\}$	*Slack variable constraints*
	$z_1(t) \leq z(t) \leq z_2(t)$	*Mass constraints*
	$m(0) = m_0, \quad \mathbf{r}(0) = \mathbf{r}_0, \quad \dot{\mathbf{r}}(0) = \dot{\mathbf{r}}_0$	*Initial conditions*
	$\mathbf{r}(t_f) = \mathbf{r}_f, \quad \dot{\mathbf{r}}(t_f) = \dot{\mathbf{r}}_f$	*Final conditions*

4.1.3 Discretised Equations of Motion

For the discretisation, the equations of motion are written into a linear time-invariant state equation. For this purpose, Açikmeşe and Ploen (2007) define the state vector $\mathbf{x}$ and the control input $\mathbf{u}$:

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}^T & \dot{\mathbf{r}}^T & z \end{bmatrix}^T \in \mathbb{R}^7 \tag{4.21}$$

$$\mathbf{u} = \begin{bmatrix} \boldsymbol{\tau}^T & \sigma \end{bmatrix}^T \in \mathbb{R}^4 \tag{4.22}$$

Using the above definitions of the state and input vectors, the equations of motion are given

by:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t) + \mathbf{B}_c \begin{bmatrix} \mathbf{g}^T(t) & 0 \end{bmatrix}^T \tag{4.23}$$

Where:

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 \end{bmatrix} \in \mathbb{R}^{7\times7} \tag{4.24}$$

$$\mathbf{B}_c = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & -a \end{bmatrix} \in \mathbb{R}^{7\times4} \tag{4.25}$$

Discretisation of system matrix $\mathbf{A}_c$ and control matrix $\mathbf{B}_c$ is given by Açikmeşe and Ploen (2007) as follows:

$$\mathbf{A}_d = e^{\mathbf{A}_c\Delta t} \tag{4.26}$$

$$\mathbf{B}_d = \int_0^{\Delta t} e^{\mathbf{A}_c(\Delta t - s)}\mathbf{B}_c \, ds \tag{4.27}$$

Approximating these expressions by a Taylor Series Expansion yields:

$$\mathbf{A}_d = \mathbf{I}_{7\times7} + \mathbf{A}_c\Delta t + \frac{1}{2!}\mathbf{A}_c^2\Delta t^2 + \frac{1}{3!}\mathbf{A}_c^3\Delta t^3 + \cdots \tag{4.28}$$

$$\mathbf{B}_d = \mathbf{B}_c\Delta t + \frac{1}{2!}\mathbf{A}_c\mathbf{B}_c\Delta t^2 + \frac{1}{3!}\mathbf{A}_c^2\mathbf{B}_c\Delta t^3 + \cdots \tag{4.29}$$

Note however, that with the definition of $\mathbf{A}_c$ in Equation 4.24, the square of the system matrix yields a zero matrix: $\mathbf{A}_c^2 = \mathbf{0}_{7\times7}$. As a result, the following expressions for the discretised system and control matrices are exact.

$$\mathbf{A}_d = \mathbf{I}_{7\times7} + \mathbf{A}_c\Delta t \tag{4.30}$$

$$\mathbf{B}_d = \mathbf{B}_c \Delta t + \frac{1}{2}\mathbf{A}_c \mathbf{B}_c \Delta t^2 \tag{4.31}$$

Given the initial state vector $\mathbf{x}(0)$, the state vector at time step k is obtained as follows:

$$\mathbf{x}(k) = \mathbf{A}_d^k \mathbf{x}_0 + \sum_{j=1}^{k} \mathbf{A}_d^{k-j} \mathbf{B}_d \left\{ \mathbf{u}(j) + \begin{bmatrix} \mathbf{g}^T(j) & 0 \end{bmatrix}^T \right\} \tag{4.32}$$

Equation 4.32 consists of a homogeneous term and a particular term, where the particular term contains a control input component and a gravity component. In the formulation of the guidance problem, the control vector $\mathbf{u}$ is the only unknown that needs to be solved.

Caveat Even though the control propagation matrix $\mathbf{B}_d$ is an analytic expression and returns the exact answer, it propagates a discretised control vector. As a consequence, the control input is propagated as if it is a zero-order hold function or a stepping function as shown in Figure 4.2. Such discontinuities are physically impossible and are bound to introduce errors. The effect of these errors is investigated in Section 6.

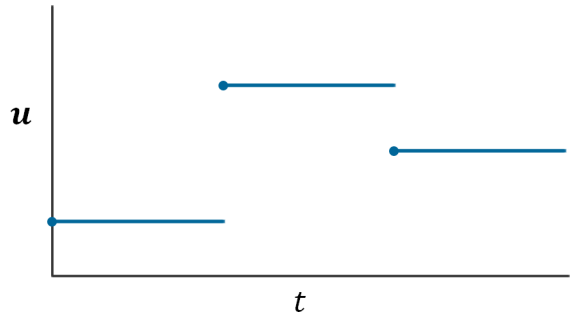


Figure 4.2: Zero-order hold function.

4.1.4 Discretised Constraints

Last section defined state vector $\mathbf{x}$ and control vector $\mathbf{u}$, and discretised the equations of motion. This section uses the definitions of the state and control vectors to rewrite the guidance optimization problem.

Açikmeşe and Ploen (2007) define the new cost function as the discrete integration of the thrust slack variable over N temporal nodes:

$$\min \sum_{k=1}^{N} \mathbf{e}_\sigma^T \mathbf{u}(k) \Delta t \tag{4.33}$$

Where $\mathbf{e}_\sigma$ is used to extract the thrust slack variable σ from control vector $\mathbf{u}$:

$$\mathbf{e}_\sigma = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \in \mathbb{R}^4 \tag{4.34}$$

Limiting the thrust to the slack variable is now expressed by Açikmeşe and Ploen (2007) as the following SOCP:

$$\|\mathbf{E}_{\boldsymbol{\tau}}\mathbf{u}\| \leq \mathbf{e}_\sigma^T\mathbf{u} \tag{4.35}$$

In the above equation, $\mathbf{E}_{\boldsymbol{\tau}}$ extracts the thrust vector $\boldsymbol{\tau}$ from control vector $\mathbf{u}$.

$$\mathbf{E}_{\boldsymbol{\tau}} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \end{bmatrix} \in \mathbb{R}^{3\times4} \tag{4.36}$$

A similar method is applied to the constraints on the slack variable:

$$\mu_1\{1 - [\mathbf{e}_z^T\mathbf{x} - z_1]\} \leq \mathbf{e}_\sigma^T\mathbf{u} \leq \mu_2\{1 - [\mathbf{e}_z^T\mathbf{x} - z_2]\} \tag{4.37}$$

Here, $\mathbf{e}_z$ is used to extract the mass z from the state vector $\mathbf{x}$:

$$\mathbf{e}_z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \in \mathbb{R}^7 \tag{4.38}$$

Using the forgoing examples, rewriting the mass constraints does not require the definition of any new extraction vectors or matrices:

$$z_1 \leq \mathbf{e}_z^T\mathbf{x} \leq z_2 \tag{4.39}$$

Ultimately, also the final boundary conditions are rewritten:

$$\mathbf{E}_{\mathbf{x}_f}\mathbf{x}(N) = \begin{bmatrix} \mathbf{r}_f^T & \dot{\mathbf{r}}_f^T \end{bmatrix}^T \tag{4.40a}$$

$$\mathbf{E}_{\mathbf{x}_f} = \begin{bmatrix} \mathbf{I}_{6\times6} & \mathbf{0}_{6\times1} \end{bmatrix} \tag{4.40b}$$

Notice how the above equation only uses the spacecraft's position and speed from the state vector, not its mass, as the mass has a free terminal condition.

Table 4.3 gives an overview of the guidance problem as derived thus far. It is assumed that N discretisation steps are used to reach the time of flight t_f from $t = 0$. This also fixes the discretisation step size Δt to an according value. Note how the equality constraints for the equations of motion and the mass flow have been omitted from the problem formulation. These equalities do not need to be stated specifically any longer, because they are incorporated in the definition of state vector $\mathbf{x}$, see Section 4.1.3. Also the specific definition of the initial conditions have disappeared because the initial conditions return in the homogeneous part of state vector $\mathbf{x}$, see Equation 4.32.

Table 4.3: Discretised guidance problem.

Minimize	$\sum_{k=1}^{N} \mathbf{e}_\sigma^T \mathbf{u}(k) \Delta t$	*Cost function*
Subject to	$k = 1, \cdots, N$	*For all epochs*
	$\|\mathbf{E}_{\boldsymbol{\tau}} \mathbf{u}(k)\| \leq \mathbf{e}_\sigma^T \mathbf{u}(k)$	*Thrust constraint*
	$\begin{cases} \mu_1(k)\{1 - [\mathbf{e}_z^T \mathbf{x}(k) - z_1(k)]\} \leq \mathbf{e}_\sigma^T \mathbf{u}(k) \\ \mathbf{e}_\sigma^T \mathbf{u}(k) \leq \mu_2(k)\{1 - [\mathbf{e}_z^T \mathbf{x}(k) - z_2(k)]\} \end{cases}$	*Slack variable constraints*
	$z_1(k) \leq \mathbf{e}_z^T \mathbf{x}(k) \leq z_2(k)$	*Mass constraints*
	$\mathbf{E}_{\mathbf{x}_f} \mathbf{x}(N) = \begin{bmatrix} \mathbf{r}_f^T & \dot{\mathbf{r}}_f^T \end{bmatrix}^T$	*Final boundary*

4.1.5 Stacked Equations of Motion

At this point, the control vector $\mathbf{u} = \begin{bmatrix} \boldsymbol{\tau}^T & \sigma \end{bmatrix}^T$ is the only unknown in the problem. The control vector is directly visible in the thrust constraints, but remember also that the state vector $\mathbf{x}$ is a linear function of the control vector $\mathbf{u}$, as shown by Equation 4.32.

By stacking all time steps of the discretised problem, an expression is developed that allows to solve the control vectors for all temporal nodes in one batch:

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}^T(0) & \cdots & \mathbf{u}^T(N-1) \end{bmatrix}^T \in \mathbb{R}^{4N} \tag{4.41}$$

The same method is used in stacking the state vectors, initial conditions and the gravitational

acceleration:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(1) & \cdots & \mathbf{x}^T(N) \end{bmatrix}^T \in \mathbb{R}^{7N} \tag{4.42}$$

$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}^T(0) & \cdots & \mathbf{x}^T(0) \end{bmatrix}^T \in \mathbb{R}^{7N} \tag{4.43}$$

$$\mathbf{G} = \begin{bmatrix} \begin{bmatrix} \mathbf{g}^T(0) & 0 \end{bmatrix}^T & \cdots & \begin{bmatrix} \mathbf{g}^T(N-1) & 0 \end{bmatrix}^T \end{bmatrix}^T \in \mathbb{R}^{4N} \tag{4.44}$$

Note that the control vectors are stacked for time steps 0 to $N-1$, while the state vectors are stacked for time steps 1 to N. For the control vectors, there is no point in including $\mathbf{u}(N)$, because no further control input is required when the final condition is reached. For the state vectors however, $\mathbf{x}(N)$ needs to be included because it is the final condition, while $\mathbf{x}(0)$ is excluded because it is stacked in a separate vector, $\mathbf{X}_0$. Despite this difference, $\mathbf{U}$, $\mathbf{X}$, $\mathbf{X}_0$ and $\mathbf{G}$ all stack N of their respective vectors. For example, because $\mathbf{x}(1)$ to $\mathbf{x}(N)$ are all functions of the initial state, $\mathbf{X}_0$ stacks $\mathbf{x}(0)$ for a number of N times.

The stacked gravity vectors in Equation 4.44 show that the gravity changes over time. In fact, gravity vector $\mathbf{g}(k)$ depends on position vector $\mathbf{r}(k)$ in the state vector $\mathbf{x}(k)$, which in term depends on the history of the control vector $\mathbf{u}(k)$. However, because $\mathbf{u}(k)$ is tried to be solved for, it is not possible to define exact values of $\mathbf{g}(k)$. Therefore, the gravity field is assumed to be homogeneous, fixing $\mathbf{g}$ to a constant value.

Using the definitions of the stacked vectors from Equations 4.41 through 4.44, the state propagation of Equation 4.32 is now written by Açikmeşe and Ploen (2007) as follows:

$$\mathbf{X} = \mathbf{S}_{\mathbf{X}}\mathbf{X}_0 + \mathbf{S}_{\mathbf{U}}\mathbf{G} + \mathbf{S}_{\mathbf{U}}\mathbf{U} \tag{4.45}$$

Where:

$$\mathbf{S}_{\mathbf{X}} = \begin{bmatrix} \mathbf{A}_d^1 & \mathbf{0}_{7\times7} & \mathbf{0}_{7\times7} \\ \mathbf{0}_{7\times7} & \ddots & \mathbf{0}_{7\times7} \\ \mathbf{0}_{7\times7} & \mathbf{0}_{7\times7} & \mathbf{A}_d^N \end{bmatrix} \in \mathbb{R}^{7N\times7N} \tag{4.46}$$

$$\mathbf{S_U} = \begin{bmatrix} \mathbf{B}_d & \mathbf{0}_{7\times4} & \cdots & \mathbf{0}_{7\times4} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \cdots & \mathbf{0}_{7\times4} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^{N-1}\mathbf{B}_d & \mathbf{A}_d^{N-2}\mathbf{B}_d & \cdots & \mathbf{B}_d \end{bmatrix} \in \mathbb{R}^{7N\times4N} \tag{4.47}$$

To shorten the notation, $\mathbf{X}_h$ is defined as the combination of the homogeneous solution and the gravity particular solution:

$$\mathbf{X}_h = \mathbf{S_X}\mathbf{X}_0 + \mathbf{S_U}\mathbf{G} \tag{4.48}$$

This simplifies Equation 4.45 to the following:

$$\mathbf{X} = \mathbf{X}_h + \mathbf{S_U}\mathbf{U} \tag{4.49}$$

4.1.6 Stacked Constraints

Açikmeşe and Ploen (2007) use the definitions of the stacked vectors from last section to rewrite the guidance problem given by Table 4.3. The cost function still is the discretized integral of the thrust slack variable σ. But instead of writing it as a sum, it can now be stated as follows:

$$\mathbf{e}_{N\sigma\Delta t}^T\mathbf{U} \tag{4.50}$$

Where $\mathbf{e}_{N\sigma\Delta t}$ extracts σ from all individual control vectors and multiplies them with the discretized step size:

$$\mathbf{e}_{N\sigma\Delta t} = \begin{bmatrix} \mathbf{e}_\sigma^T\Delta t & \cdots & \mathbf{e}_\sigma^T\Delta t \end{bmatrix}^T \in \mathbb{R}^{4N} \tag{4.51}$$

Because the thrust constraint is an SOCP and utilizes the norm of a vector, these constraints cannot easily be stacked. Therefore, Açikmeşe and Ploen (2007) formulate the thrust constraints as follows for each temporal node:

$$\|\mathbf{E}_{\boldsymbol{\tau}}\mathbf{E}_{k,\mathbf{U}}\mathbf{U}\| \leq \mathbf{e}_\sigma^T\mathbf{E}_{k,\mathbf{U}}\mathbf{U}, \quad k = 1, \cdots, N \tag{4.52}$$

Where $\mathbf{E}_{k,\mathbf{U}}$ extracts $\mathbf{u}(k)$ from $\mathbf{U}$:

$$\mathbf{E}_{k,\mathbf{U}} = \begin{bmatrix} \underbrace{\mathbf{0}_{4\times4} \quad \cdots}_{\times(k-1)} & \mathbf{I}_{4\times4} & \underbrace{\cdots \quad \mathbf{0}_{4\times4}}_{\times(N-k)} \end{bmatrix} \in \mathbb{R}^{4\times4N} \tag{4.53}$$

The constraints of the slack variable are linear and can therefore be stacked as follows:

$$\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - (\mathbf{E}_z\mathbf{X} - \mathbf{z}_{1N})\right] \leq \mathbf{E}_\sigma\mathbf{U} \leq \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - (\mathbf{E}_z\mathbf{X} - \mathbf{z}_{2N})\right] \tag{4.54}$$

Where:

$$\mathbf{E}_\sigma = \begin{bmatrix} \mathbf{e}_\sigma^T & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \ddots & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} & \mathbf{e}_\sigma^T \end{bmatrix} \in \mathbb{R}^{N\times4N} \tag{4.55}$$

$$\mathbf{E}_z = \begin{bmatrix} \mathbf{e}_z^T & \mathbf{0}_{1\times7} & \mathbf{0}_{1\times7} \\ \mathbf{0}_{1\times7} & \ddots & \mathbf{0}_{1\times7} \\ \mathbf{0}_{1\times7} & \mathbf{0}_{1\times7} & \mathbf{e}_z^T \end{bmatrix} \in \mathbb{R}^{N\times7N} \tag{4.56}$$

$$\boldsymbol{\mu}_{1N} = \begin{bmatrix} \mu_1(1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mu_1(N) \end{bmatrix} \in \mathbb{R}^{N\times N} \tag{4.57}$$

$$\boldsymbol{\mu}_{2N} = \begin{bmatrix} \mu_2(1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mu_2(N) \end{bmatrix} \in \mathbb{R}^{N\times N} \tag{4.58}$$

$$\mathbf{z}_{1N} = \begin{bmatrix} \ln\left(m_0 - aT_l\Delta t \cdot 1\right) \\ \vdots \\ \ln\left(m_0 - aT_l\Delta t \cdot N\right) \end{bmatrix} \in \mathbb{R}^N \tag{4.59}$$

$$\mathbf{z}_{2N} = \begin{bmatrix} \ln\left(m_0 - aT_h\Delta t \cdot 1\right) \\ \vdots \\ \ln\left(m_0 - aT_h\Delta t \cdot N\right) \end{bmatrix} \in \mathbb{R}^N \tag{4.60}$$

$$\mathbf{1}_N = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^N \tag{4.61}$$

Using the above definitions, the mass constraint can be stacked as follows:

$$\mathbf{z}_{2N} \leq \mathbf{E}_z \mathbf{X} \leq \mathbf{z}_{1N} \tag{4.62}$$

This only leaves the final boundary conditions to be expressed as a function of $\mathbf{X}$:

$$\mathbf{E}_{\mathbf{x}(N)} \mathbf{X} = \begin{bmatrix} \mathbf{r}_f^T & \dot{\mathbf{r}}_f^T \end{bmatrix}^T \tag{4.63}$$

Where:

$$\mathbf{E}_{\mathbf{x}(N)} = \begin{bmatrix} \underbrace{\mathbf{0}_{6\times 7} \quad \cdots}_{\times(N-1)} & \mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 1} \end{bmatrix} \in \mathbb{R}^{6\times 7N} \tag{4.64}$$

Table 4.4 shows an overview of the final expression for the convexified base guidance problem. An overview of all variables and their relations is given in Appendix B.

Table 4.4: Convexified discretized stacked guidance problem. Source: Gerth (2014).

Minimize	$\mathbf{e}_{N\sigma\Delta t}^T \mathbf{U}$	*Cost function*
Subject to	$\lVert \mathbf{E}_{\boldsymbol{\tau}} \mathbf{E}_{k,\mathbf{U}} \mathbf{U} \rVert \leq \mathbf{e}_\sigma^T \mathbf{E}_{k,\mathbf{U}} \mathbf{U}, \quad k = 1, \cdots, N$	*Thrust constraint*
	$\begin{cases} \boldsymbol{\mu}_{1N} \left[\mathbf{1}_N - (\mathbf{E}_z \mathbf{X} - \mathbf{z}_{1N}) \right] \leq \mathbf{E}_\sigma \mathbf{U} \\ \mathbf{E}_\sigma \mathbf{U} \leq \boldsymbol{\mu}_{2N} \left[\mathbf{1}_N - (\mathbf{E}_z \mathbf{X} - \mathbf{z}_{2N}) \right] \end{cases}$	*Slack variable constraints*
	$\mathbf{z}_{2N} \leq \mathbf{E}_z \mathbf{X} \leq \mathbf{z}_{1N}$	*Mass constraints*
	$\mathbf{E}_{\mathbf{x}(N)} \mathbf{X} = \begin{bmatrix} \mathbf{r}_f^T & \dot{\mathbf{r}}_f^T \end{bmatrix}^T$	*Final boundary*

4.2 Extensions to Convex Guidance

Table 4.4 in Section 4.1.6 shows the standard guidance problem, expressed as a convex problem. As a consequence, any optimal solution found to the problem, is guaranteed to be the global optimum. From this base expression, additional constraints can be imposed to improve the guidance algorithm. Section 4.2.1 adds a glide slope, as provided by Açikmeşe and Ploen (2007). Section 4.2.2 adds an attitude constraint, as originally proposed by Açikmeşe and Ploen (2007), but extended by Gerth (2014). Section 4.2.3 improves the internal gravity model of the guidance algorithm,as developed by Gerth (2014).

4.2.1 Glide–Slope Constraint

Although the problem defined in Table 4.4 can be solved by a guidance algorithm, it might not return the desired result. Even when the initial condition for the altitude $r_z(0)$ is given a positive value and the final condition is $r_z = 0$, no constraint prevents $r_z(k)$ to become negative throughout the flight. Physically this would represent a sub-surface flight, meaning the spacecraft would crash into the Lunar surface. A solution to this problem could be to add a constraint to force r_z to remain positive. However, the Lunar terrain is not perfectly flat and the spacecraft could still fly into mountains, hills, boulders or other objects on the surface near the landing site. Therefore, a glide-slope is introduced to stay clear of the Lunar surface and possible obstacles near the landing site. The required angle of the glide-slope depends on the geometry of the landing site and its surroundings.

Açikmeşe and Ploen (2007) define the glide-slope by angle α and the elevation of the spacecraft by angle ϵ, both with respect to the landing site as shown in Figure 4.3. Note that, because the spacecraft can come in from any direction to land, the glide-slope forms a cone focussed at the landing site. Next, $\tilde{\epsilon}$ is defined as the co-elevation of the spacecraft and $\tilde{\alpha}$ is defined as the co-glide-slope. From the geometry in Figure 4.3, one obtains the following relation:

$$\sqrt{r_x^2 + r_y^2} = \tan(\tilde{\epsilon}) r_z \tag{4.65}$$

Both sides of the above equality represent a downrange distance from the landing site. Similarly, the downrange distance of the glide-slope for a given altitude can be given by $\tan(\tilde{\alpha}) r_z$.

For the spacecraft to be within the glide-slope-cone, the downrange distance of the spacecraft needs to be smaller than or equal to the maximum allowed downrange distance defined by the

glide-slope cone:

$$\sqrt{r_x^2 + r_y^2} \leq \tan(\tilde{\alpha}) r_z \tag{4.66}$$

Stacking this inequality for all temporal nodes, produces the following relation:

$$\|\mathbf{E_{xy}} \mathbf{E}_{k,\mathbf{X}} \mathbf{X}\| \leq \mathbf{e}_d^T \mathbf{E}_{k,\mathbf{X}} \mathbf{X}, \quad k = 1, \cdots, N \tag{4.67}$$

Where $\mathbf{e}_d$ determines the limit of the downrange distance given the spacecraft's altitude, $\mathbf{E_{xy}}$ extracts the $\mathbf{xy}$-downrange-vector from state vector $\mathbf{x}$ and $\mathbf{E}_{k,\mathbf{X}}$ extracts $\mathbf{x}(k)$ from $\mathbf{X}$:

$$\mathbf{e}_d = \begin{bmatrix} \mathbf{0}_{1\times 2} & \tan(\pi/2 - \alpha) & \mathbf{0}_{1\times 4} \end{bmatrix}^T \in \mathbb{R}^7 \tag{4.68}$$

$$\mathbf{E_{xy}} = \begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 5} \end{bmatrix} \in \mathbb{R}^{2\times 7} \tag{4.69}$$

$$\mathbf{E}_{k,\mathbf{X}} = \begin{bmatrix} \underbrace{\mathbf{0}_{7\times 7} \quad \cdots}_{\times(k-1)} & \mathbf{I}_{7\times 7} & \underbrace{\cdots \quad \mathbf{0}_{7\times 7}}_{\times(N-k)} \end{bmatrix} \in \mathbb{R}^{7\times 7N} \tag{4.70}$$

Because the magnitude of a 2 dimensional vector is compared to a scalar, the glide-slope constraint is a 3 dimensional SOCP.

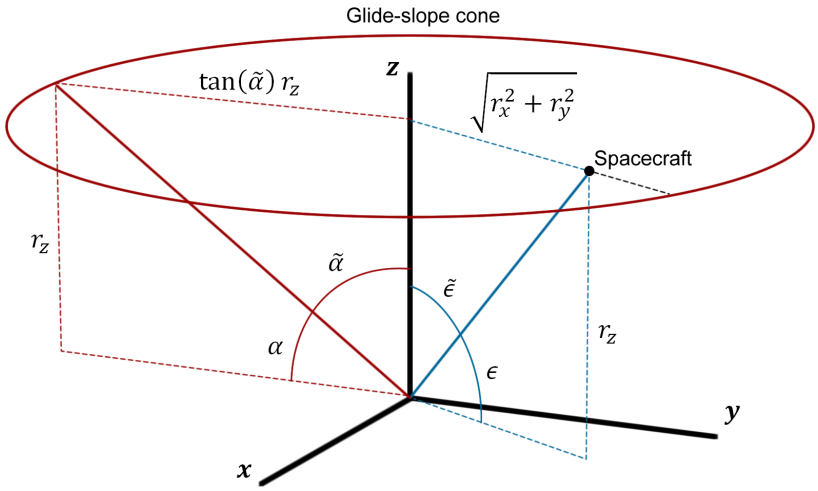


Figure 4.3: The spacecraft's position within the approach cone, as defined by the glide-slope angle.

4.2.2 Attitude Constraint

The guidance algorithm developed thus far, optimizes a trajectory from the initial state to the final conditions without allowing subsurface flight. It does however, allow for any kind of attitude of the spacecraft. Out of safety considerations, or by the limitations of navigational sensors, it might be required to limit the allowed pitch angle of the spacecraft. Açikmeşe and Ploen (2007) add an additional constraint, which constricts the thrust vector within a specified cone. Because the spacecraft changes the direction of the thrust vector by changing the attitude of the spacecraft, the thrust pointing constraint is also an attitude constraint.

The combination of Equation 4.71 and Figure 4.4 shows that the thrust vector $\mathbf{T}$ must match the unit vector $\hat{\mathbf{n}}$ to within an angle of θ. In this particular example, Açikmeşe and Ploen (2007) defined $\hat{\mathbf{n}}$ as the normal vector to the Lunar surface. However, $\hat{\mathbf{n}}$ could practically be any unit vector.

$$\hat{\mathbf{n}}^T\mathbf{T} \geq \|\mathbf{T}\| \cos\theta \tag{4.71}$$

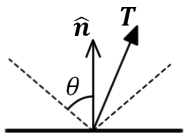


Figure 4.4: Thrust vector $\mathbf{T}$ must match $\hat{\mathbf{n}}$ in direction within the marging of θ.

Replacing the thrust vector with the slack variable and the replacement vector, and stacking the temporal nodes yields the constraints of Equation 4.72:

$$\hat{\mathbf{n}}^T\mathbf{E}_{\boldsymbol{\tau}}\mathbf{E}_{k,\mathbf{U}}\mathbf{U} \geq \mathbf{e}_\sigma^T\mathbf{E}_{k,\mathbf{U}}\mathbf{U}\cos\theta, \quad k = 1, \cdots, N \tag{4.72}$$

In Equation 4.72, $\hat{\mathbf{n}}$ points zenith. However, this vector can be replaced by any other unit vector. Furthermore, $\hat{\mathbf{n}}$ and θ have been assumed time invariant thus far, but can also be time variant.

Gerth (2014) uses this expression to develop a constraint for pointing the landing sensors towards the landing target. The difficulty in this, is that the sensor's field of view does not necessarily line up with the spacecraft's thrust vector. Gerth develops Equation 4.73 to acquire the pointing vector to align the sensor with the landing site:

$$\hat{\mathbf{n}} = R_z(-\tau)R_y(-\gamma)R_z(\tau)\hat{\mathbf{r}} \tag{4.73}$$

In this equation, τ denotes the direction of the spacecraft with respect to the landing site and γ denotes the offset of the sensor viewing direction with respect to the thrust vector (spacecraft's vertical axis). This new definition of the pointing vector is directly substituted in Equation 4.72. Now θ represents the limiting angle at which the sensors may be off-target from the landing site.

The challenge with this time-varying constraint is that it depends on the trajectory, while the trajectory is to be solved for. In other words, the trajectory depends on the attitude constraint, while the attitude constraint depends on the trajectory to line up the landing sensors. To overcome this dependency loop, the optimization problem is initially solved without attitude constraint. The acquired trajectory is then fed into the next optimization problem, which includes the attitude constraint. The trajectory from the initial problem is used as reference for this attitude constraint. At first, the margin θ for the attitude constraint is set very loose so that a solution is possible. Several iterations follow, where every iteration uses the trajectory of the previous iteration as a reference for their attitude constraint to tighten θ to its desired value. Directly using the final value for θ can require the same trajectory to be flown at very different angles, which may be impossible, yielding no solution. The guidance problem with the added glide-slope constraint and the attitude constraint is shown in Table 4.5. An overview of all variables and their relations is provided in Appendix B.

Table 4.5: Convexified discretized stacked guidance problem, with added glide-slope and attitude constraints. Source: Gerth (2014).

Minimize	$\mathbf{e}_{N\sigma\Delta t}^T \mathbf{U}$	*Cost function*
Subject to	$\|\mathbf{E}_{\boldsymbol{\tau}} \mathbf{E}_{k,\mathbf{U}} \mathbf{U}\| \leq \mathbf{e}_\sigma^T \mathbf{E}_{k,\mathbf{U}} \mathbf{U}, \quad k = 1, \cdots, N$	*Thrust constraint*
	$\begin{cases} \boldsymbol{\mu}_{1N} \left[\mathbf{1}_N - (\mathbf{E}_z \mathbf{X} - \mathbf{z}_{1N})\right] \leq \mathbf{E}_\sigma \mathbf{U} \\ \mathbf{E}_\sigma \mathbf{U} \leq \boldsymbol{\mu}_{2N} \left[\mathbf{1}_N - (\mathbf{E}_z \mathbf{X} - \mathbf{z}_{2N})\right] \end{cases}$	*Slack variable constraints*
	$\mathbf{z}_{2N} \leq \mathbf{E}_z \mathbf{X} \leq \mathbf{z}_{1N}$	*Mass constraints*
	$\|\mathbf{E}_{\mathbf{xy}} \mathbf{E}_{k,\mathbf{X}} \mathbf{X}\| \leq \mathbf{e}_d^T \mathbf{E}_{k,\mathbf{X}} \mathbf{X}, \quad k = 1, \cdots, N$	*Glide-slope constraint*
	$\hat{\mathbf{n}}^T \mathbf{E}_{\boldsymbol{\tau}} \mathbf{E}_{k,\mathbf{U}} \mathbf{U} \geq \mathbf{e}_\sigma^T \mathbf{E}_{k,\mathbf{U}} \mathbf{U} \cos\theta, \quad k = 1, \cdots, N$	*Attitude constraint*
	$\mathbf{E}_{\mathbf{x}(N)} \mathbf{X} = \begin{bmatrix} \mathbf{r}_f^T & \dot{\mathbf{r}}_f^T \end{bmatrix}^T$	*Final boundary*

4.2.3 Spherical Gravity Model

Thus far, all derivations assumed a flat Moon with constant gravity. However, because of the travelling speeds of the spacecraft, the curvature of the Moon has a significant effect on the trajectory of the spacecraft. By simulating gravity turns for the Main Braking Phase (MBP), Gerth (2014) shows that a flat Moon model can produce errors in the order of kilometres with respect to a spherical gravity model when arriving at Approach Gate (AG).

Linear Time-Variant System

To model a spherical gravity field instead of a homogeneous field, Liu and Lu (2014) change the equations of motion. Where the acceleration of the spacecraft was previously given by:

$$\ddot{\mathbf{r}} = \boldsymbol{\tau} + \mathbf{g} \tag{4.74}$$

Liu and Lu (2014) use the following expression:

$$\ddot{\mathbf{r}} = \boldsymbol{\tau} - \frac{\mu}{\|\mathbf{r}\|^3}\mathbf{r} \tag{4.75}$$

As a reminder, the dynamic equations used thus far are given in Section 4.1.3 as:

$$\underbrace{\begin{bmatrix}\dot{\mathbf{r}}\\ \ddot{\mathbf{r}}\\ \dot{z}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0\end{bmatrix}}_{\mathbf{A}_c}\underbrace{\begin{bmatrix}\mathbf{r}\\ \dot{\mathbf{r}}\\ z\end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{1\times3} & -a\end{bmatrix}}_{\mathbf{B}_c}\underbrace{\begin{bmatrix}\boldsymbol{\tau}\\ \sigma\end{bmatrix}}_{\mathbf{u}} + \underbrace{\begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{1\times3} & -a\end{bmatrix}}_{\mathbf{B}_c}\begin{bmatrix}\mathbf{g}\\ 0\end{bmatrix} \tag{4.76}$$

Using the expression proposed by Liu and Lu (2014) (Equation 4.75) changes the dynamic equations:

$$\underbrace{\begin{bmatrix}\dot{\mathbf{r}}\\ \ddot{\mathbf{r}}\\ \dot{z}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\\ \frac{-\mu}{\|\mathbf{r}\|^3}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0\end{bmatrix}}_{\mathbf{A}_c(\mathbf{r})}\underbrace{\begin{bmatrix}\mathbf{r}\\ \dot{\mathbf{r}}\\ z\end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1}\\ \mathbf{0}_{1\times3} & -a\end{bmatrix}}_{\mathbf{B}_c}\underbrace{\begin{bmatrix}\boldsymbol{\tau}\\ \sigma\end{bmatrix}}_{\mathbf{u}} \tag{4.77}$$

Comparing Equation 4.77 with 4.76 shows that the gravity vector $\mathbf{g}$ has been removed and non-zero terms have been added to $\mathbf{A}_c$. It shows that $\mathbf{A}_c$ has become a function of the position vector of the spacecraft, which is part of the state. As a consequence, the equations of motion have become a non-linear system, where the original equations of motion (Equation 4.76) form a linear time-invariant system. By using a nominal trajectory for $\mathbf{r}$ in $\mathbf{A}_c(\mathbf{r})$, Equation 4.77 becomes a linear time-varying system.

For discretization of the system and control matrices $\mathbf{A}_c$ and $\mathbf{B}_c$, the same approximation holds as discussed in Section 4.1.3:

$$\mathbf{A}_d(\mathbf{r}) = \mathbf{I}_{7\times 7} + \mathbf{A}_c(\mathbf{r})\Delta t + \frac{1}{2!}\mathbf{A}_c^2(\mathbf{r})\Delta t^2 + \frac{1}{3!}\mathbf{A}_c^3(\mathbf{r})\Delta t^3 + \cdots \tag{4.78}$$

$$\mathbf{B}_d(\mathbf{r}) = \mathbf{B}_c\Delta t + \frac{1}{2!}\mathbf{A}_c(\mathbf{r})\mathbf{B}_c\Delta t^2 + \frac{1}{3!}\mathbf{A}_c^2(\mathbf{r})\mathbf{B}_c\Delta t^3 + \cdots \tag{4.79}$$

Because the definition of $\mathbf{A}_c$ has changed, squaring no longer results in a zero-matrix, $\mathbf{A}_c^2 \neq \mathbf{0}_{7\times 7}$. Therefore, higher order terms are non-zero in this case and need to be included. Practically, it was found that including the terms up to the fourth order is a satisfactory approximation. Using the Δt^0 and Δt terms yields errors in the order of kilometres for the MBP, while including the Δt^2 term yields errors in the order of decimetres. Including the Δt^3 term reduces the errors to centimetres and including the Δt^4 term reduces the errors to the sub-millimetre level, as shown in Table 4.6.

Table 4.6: Position errors at the end of the MBP, caused by truncation of Equations 4.78 and 4.79.

Order	Uncertainty
Δt^1	kilometres
Δt^2	decimetres
Δt^3	centimetres
Δt^4	< millimetres

Because other sources are expected to cause errors in the order of metres, sub-millimetre level accuracy over the MBP is good enough. For $\Delta t = 5$ s for example, Gerth (2014) obtained errors in the order of metres. This error can be reduced by using a lower value for Δt. However, this squares the problem size and significantly increases the required computational memory and effort (see Section 6), making it less suitable for use onboard of a spacecraft.

For the linear time-invariant system, the discretized system and input matrices were stacked in

Section 4.1.5 as follows:

$$\mathbf{S_X} = \begin{bmatrix} \mathbf{A}_d^1 & \mathbf{0}_{7\times7} & \mathbf{0}_{7\times7} \\ \mathbf{0}_{7\times7} & \ddots & \mathbf{0}_{7\times7} \\ \mathbf{0}_{7\times7} & \mathbf{0}_{7\times7} & \mathbf{A}_d^N \end{bmatrix} \in \mathbb{R}^{7N\times7N} \tag{4.80}$$

$$\mathbf{S_U} = \begin{bmatrix} \mathbf{B}_d & \mathbf{0}_{7\times4} & \cdots & \mathbf{0}_{7\times4} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \cdots & \mathbf{0}_{7\times4} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^{N-1}\mathbf{B}_d & \mathbf{A}_d^{N-2}\mathbf{B}_d & \cdots & \mathbf{B}_d \end{bmatrix} \in \mathbb{R}^{7N\times4N} \tag{4.81}$$

Because $\mathbf{A}_d$ and $\mathbf{B}_d$ are now time-variant, the expressions change to the following equations, where $\mathbf{A}_t = \mathbf{A}_d(\mathbf{r}(t))$ and $\mathbf{B}_t = \mathbf{B}_d(\mathbf{r}(t))$:

$$\mathbf{S_X} = \begin{bmatrix} \mathbf{A}_0 & & & \\ & \mathbf{A}_1\mathbf{A}_0 & & \\ & & \ddots & \\ & & & (\mathbf{A}_{N-1}\mathbf{A}_{N-2}\cdots\mathbf{A}_0) \end{bmatrix} \in \mathbb{R}^{7N\times7N} \tag{4.82}$$

$$\mathbf{S_U} = \begin{bmatrix} \mathbf{B}_0 & & & \\ \mathbf{A}_1\mathbf{B}_0 & \mathbf{B}_1 & & \\ \mathbf{A}_2\mathbf{A}_1\mathbf{B}_0 & \mathbf{A}_2\mathbf{B}_1 & & \\ \vdots & \vdots & \ddots & \\ (\mathbf{A}_{N-1}\cdots\mathbf{A}_1\mathbf{B}_0) & (\mathbf{A}_{N-1}\cdots\mathbf{A}_2\mathbf{B}_1) & \cdots & \mathbf{B}_{N-1} \end{bmatrix} \in \mathbb{R}^{7N\times4N} \tag{4.83}$$

For the nominal trajectory of $\mathbf{r}$, the true trajectory cannot be used, because that is what is optimized for. Therefore, the initial condition is used for $\mathbf{r}$, as if the spacecraft is stationary at its initial position. After the first optimal trajectory is obtained, this trajectory is used as the nominal trajectory for $\mathbf{r}$ in the following iteration. Iterations are performed where $\mathbf{r}$ is based on the optimal trajectory from the previous iteration, until the change in cost from the cost-function is sufficiently small. In the order of 3 to 5 iterations can be expected.

The cost-function represents the required ΔV for the landing, in metres per second. Simulations of the MBP have shown that the order of change of the cost function, approximately corresponds with the order of the accuracy of the spacecraft's position at AG, as shown in Table 4.7. To introduce position errors smaller than millimetres at the end of the MBP, the iterative loop can be stopped once the change in cost is less than a mm/s.

Table 4.7: Position error at the end of the MBP, as affected by the iteration stop-criterium.

Change in cost	Uncertainty
m/s	metres
dm/s	decimetres
cm/s	centimetres
mm/s	millimetres

Caveat To model a spherical gravity field by the expression of 4.75, the position vector $\mathbf{r}$ needs to be expressed in the inertial frame $\mathcal{I}$. However, the glide-slope constraint has previously been expressed in the landing site frame $\mathcal{L}$ (Section 4.2.1). Therefore, the glide-slope constraint cannot be used when using this spherical gravity field model and vice versa. The attitude constraint can be expressed in either reference frames, as can the original thrust and mass constraints.

Time-Variant Gravity Vector

Gerth (2014) uses the glide-slope constraint in his research during the Approach Phase (AP). Therefore, he cannot make use of the above mentioned spherical gravity model. However, Gerth (2014) finds that using a homogeneous gravity field model yields errors in the order of metres at Terminal Gate (TG). To reduce these errors, Gerth (2014) proposes a spherical gravity model for the AP that can be expressed in any frame, including the landing site frame $\mathcal{L}$ so that the glide-slope constraint can be used.

Instead of changing the system matrix, Gerth (2014) makes the gravity vector $\mathbf{g}$ time-variant as shown by Equation 4.84. Because the system and control matrices remain unchanged, they can be stacked as discussed in Section 4.1.5.

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \\ z \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & -a \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau} \\ \sigma \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & -a \end{bmatrix} \begin{bmatrix} \mathbf{g}(\mathbf{r}) \\ 0 \end{bmatrix} \tag{4.84}$$

Here, $\mathbf{g}$ is a function of the position of the spacecraft $\mathbf{r}$. But because the trajectory of the spacecraft is unknown, a nominal trajectory for $\mathbf{r}$ must be used. Like for the linear time-variant system, the initial position of the spacecraft is used to determine $\mathbf{g}(\mathbf{r})$ on the first iteration. Any following iteration uses the optimal trajectory of the last iteration to determine $\mathbf{g}(\mathbf{r})$.

Simulations were performed to determine when the iterations can best be stopped. The approach phase as described in Section 2.4 was used as a reference. For the AP it was found that the iterations can be terminated when the change of the cost-function is less than 1 m/s, introducing errors smaller than a millimetre in the landing position.

This method of modelling a spherical gravity field is not used for the MBP, because it performs significantly less than the linear time-variant system. The relative lack in performance is a consequence of the equations of motion. Section 4.1.3 discusses how the equations of motion are discretized and how this assumes a zero-order hold function on the optimal control. Because for this spherical gravity model the gravity vector goes through the same control matrix as the control vector, the gravitational acceleration consequently also becomes a zero-order hold function as discussed Section 4.1.3. Therefore, this method of modelling a spherical gravity field is less accurate than the linear time-variant system. For example, simulations were run for a landing from a 10x100 km orbit, with a time of flight of 670 s, initiating the powered descent 630 km downrange and a Δt of 5 s for the temporal nodes of the guidance algorithm. These settings result in a landing error in the order of metres for the linear time-variant system. For the time-variant gravity vector however, this same simulation results in a landing error in the order of hundreds of metres. Therefore, it is adviced to only use this method when the problem cannot be expressed in the inertial frame, when a glide-slope is required for example. Both errors can be reduced by taking smaller values for Δt, but this exponentially increases the problem size and the required computational effort, as shown in Section 6.

Synthesis

So far, this section has shown that a spherical gravity field can be modelled by either using a linear time-variant system or a time-variant gravity vector. In terms of landing errors, the linear time-variant system performs better. On the other hand, the time-variant gravity vector allows for the addition of a glide-slope constraint, because it allows the problem to be formulated in the landing site frame $\mathcal{L}$.

Because a glide-slope is only required for the last stage of the landing, the approach phase, it is proposed to use the linear time-variant system for the MBP and the time-variant gravity vector for the AP.

The improved performance of the linear time-variant system is important for the MBP, because the furthest distance is covered during the MBP, meaning that the curvature of the Moon will have a more significant effect. Furthermore, when the powered descent is initiated between 600 to 700 km downrange, the spacecraft is still well below the horizon as seen from the landing site. Therefore, a glide-slope would not make sense during the MBP. During the

approach phase the spacecraft travels much slower, it is only a couple of kilometres away from the landing site and it flies lower. Therefore, the curvature of the Moon has a far smaller effect and a glide-slope becomes important, favouring the time-variant gravity vector method to be used during the AP.

4.3 Transcription to the Conical Form

Solving any of the foregoing formulations of the guidance problem on board a spacecraft requires a numerical solver, which need the equations in a linear equation. However, no solver takes the problem as it is currently written. Therefore, the problem needs to be transcribed, either by hand or by a modelling language, to the conical form. Both Açikmeşe and Ploen (2007) and Gerth (2014) use a modelling language (YALMIP and CVX, respectively) to translate the problem to the conical form. This eases the prototyping process, but it also increases the computational load. Because further development of the guidance algorithm is not in the scope of this study, the guidance problem is transcribed by hand for a faster implementation.

The only element that remains unchanged for the transcription is the cost function. All other elements are transcribed in the following sections. Section 4.3.1 start with transcription of the equality constraints. This is followed by Sections 4.3.2 and 4.3.3, which transcribe the linear inequality constraints and the second-order conic constraints, respectively.

4.3.1 Equality Constraints

The current equality constraint for the final boundary condition is given in Table 4.5 as follows:

$$\mathbf{E}_{\mathbf{x}(N)}\mathbf{X} = \begin{bmatrix}\mathbf{r}_f^T & \dot{\mathbf{r}}_f^T\end{bmatrix}^T \tag{4.85}$$

For a solver, when solving for $\mathbf{U}$, this needs to be cast to the form (Domahidi et al., 2013):

$$\mathbf{A}\mathbf{U} = \mathbf{b} \tag{4.86}$$

This shows that all terms linear with the stacked control vector $\mathbf{U}$ need to be grouped at the left side, while all terms independent of $\mathbf{U}$ belong on the right-hand side. Equation 4.85 is

clearly not of this form, as $\mathbf{X}$ is a function of $\mathbf{U}$. From Section 4.1.5:

$$\mathbf{X} = \mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U} \tag{4.87}$$

Where $\mathbf{X}_h$ is the homogeneous solution to the stacked state vector with the gravity particular solution superimposed:

$$\mathbf{X}_h = \mathbf{S}_\mathbf{X}\mathbf{X}_0 + \mathbf{S}_\mathbf{U}\mathbf{G} \tag{4.88}$$

Substituting Equation 4.87 into Equation 4.85 yields the following:

$$\mathbf{E}_{\mathbf{x}(N)}\left(\mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U}\right) = \begin{bmatrix}\mathbf{r}_f^T & \dot{\mathbf{r}}_f^T\end{bmatrix}^T \tag{4.89}$$

Moving the terms independent of $\mathbf{U}$ to the right-hand side of the equality yields the final form:

$$\mathbf{E}_{\mathbf{x}(N)}\mathbf{S}_\mathbf{U}\mathbf{U} = \begin{bmatrix}\mathbf{r}_f^T & \dot{\mathbf{r}}_f^T\end{bmatrix}^T - \mathbf{E}_{\mathbf{x}(N)}\mathbf{X}_h \tag{4.90}$$

Equation 4.90 is the conical form in which the numerical solver accepts the final boundary conditions.

4.3.2 Linear Inequality Constraints

Linear inequality constraints are the slack-variable constraints, mass constraints and the attitude constraints. The form in which numerical solvers accept linear inequality constraints is given Domahidi et al. (2013):

$$\mathbf{A}\mathbf{U} \leq \mathbf{b} \tag{4.91}$$

Because this form is very similar to the required form of the equality constraints in Section 4.3.1, the transcription follows the same procedure with the substitution of the following relation for the stacked state vectors:

$$\mathbf{X} = \mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U} \tag{4.92}$$

Slack Variable Constraint

The current constraint on the slack variable is as follows, From Table 4.5:

$$\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - (\mathbf{E}_z\mathbf{X} - \mathbf{z}_{1N})\right] \leq \mathbf{E}_\sigma\mathbf{U} \leq \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - (\mathbf{E}_z\mathbf{X} - \mathbf{z}_{2N})\right] \tag{4.93}$$

Substitution of Equation 4.92 yields:

$$\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - (\mathbf{E}_z(\mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U}) - \mathbf{z}_{1N})\right] \leq \mathbf{E}_\sigma\mathbf{U} \leq \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - (\mathbf{E}_z(\mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U}) - \mathbf{z}_{2N})\right] \tag{4.94}$$

Writing the lower and upper limit as two separate inequalities and working out the brackets yields:

$$\begin{cases} \boldsymbol{\mu}_{1N}\mathbf{1}_N - \boldsymbol{\mu}_{1N}\mathbf{E}_z\mathbf{X}_h - \boldsymbol{\mu}_{1N}\mathbf{E}_z\mathbf{S}_\mathbf{U}\mathbf{U} + \boldsymbol{\mu}_{1N}\mathbf{z}_{1N} \leq \mathbf{E}_\sigma\mathbf{U} \\ \mathbf{E}_\sigma\mathbf{U} \leq \boldsymbol{\mu}_{2N}\mathbf{1}_N - \boldsymbol{\mu}_{2N}\mathbf{E}_z\mathbf{X}_h - \boldsymbol{\mu}_{2N}\mathbf{E}_z\mathbf{S}_\mathbf{U}\mathbf{U} + \boldsymbol{\mu}_{2N}\mathbf{z}_{2N} \end{cases} \tag{4.95}$$

Subsequently writing all terms of $\mathbf{U}$ to the left-hand side while writing the other terms to the right-hand side of the inequality yields the following, which is the form accepted by ECOS.

$$\begin{cases} -(\mathbf{E}_\sigma + \boldsymbol{\mu}_{1N}\mathbf{E}_z\mathbf{S}_\mathbf{U})\mathbf{U} \leq -\boldsymbol{\mu}_{1N}(\mathbf{1}_N - \mathbf{E}_z\mathbf{X}_h + \mathbf{z}_{1N}) \\ (\mathbf{E}_\sigma + \boldsymbol{\mu}_{2N}\mathbf{E}_z\mathbf{S}_\mathbf{U})\mathbf{U} \leq \boldsymbol{\mu}_{2N}(\mathbf{1}_N - \mathbf{E}_z\mathbf{X}_h + \mathbf{z}_{2N}) \end{cases} \tag{4.96}$$

Mass Constraint

The transcription of the mass constraint is very similar to that of the slack-variable constraint. Luckily however, the mass constraint contains less variables. From Table 4.5:

$$\mathbf{z}_{2N} \leq \mathbf{E}_z\mathbf{X} \leq \mathbf{z}_{1N} \tag{4.97}$$

Substituting Equation 4.92 for $\mathbf{X}$ yields:

$$\mathbf{z}_{2N} \leq \mathbf{E}_z(\mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U}) \leq \mathbf{z}_{1N} \tag{4.98}$$

Writing the lower and upper limits as separate inequalities again, yields:

$$\begin{cases} \mathbf{z}_{2N} \leq \mathbf{E}_z \mathbf{X}_h + \mathbf{E}_z \mathbf{S}_\mathbf{U} \mathbf{U} \\ \mathbf{E}_z \mathbf{X}_h + \mathbf{E}_z \mathbf{S}_\mathbf{U} \mathbf{U} \leq \mathbf{z}_{1N} \end{cases} \tag{4.99}$$

Subsequently rearranging all terms to their required sides of the inequality, yields the final expression:

$$\begin{cases} -\mathbf{E}_z \mathbf{S}_\mathbf{U} \mathbf{U} \leq -\mathbf{z}_{2N} + \mathbf{E}_z \mathbf{X}_h \\ \mathbf{E}_z \mathbf{S}_\mathbf{U} \mathbf{U} \leq \mathbf{z}_{1N} - \mathbf{E}_z \mathbf{X}_h \end{cases} \tag{4.100}$$

Attitude Constraint

The attitude constraint was thus far defined by Table 4.5 as:

$$\hat{\mathbf{n}}^T \mathbf{E}_{\boldsymbol{\tau}} \mathbf{E}_{k,\mathbf{U}} \mathbf{U} \geq \mathbf{e}_\sigma^T \mathbf{E}_{k,\mathbf{U}} \mathbf{U} \cos\theta, \quad k = 1, \cdots, N \tag{4.101}$$

Because both terms in the above inequality contain the stacked control vector $\mathbf{U}$, both terms of the inequality are moved to the same side to obtain the required form for the numerical solvers:

$$(\mathbf{e}_\sigma^T \mathbf{E}_{k,\mathbf{U}} \cos\theta - \hat{\mathbf{n}}^T \mathbf{E}_{\boldsymbol{\tau}} \mathbf{E}_{k,\mathbf{U}}) \mathbf{U} \leq 0, \quad k = 1, \cdots, N \tag{4.102}$$

4.3.3 Second-Order Conic Constraints

Up to this point, the transcription of the guidance problem only involved rearranging terms. The true reason for the transcription to be required, is because of the thrust and glide-slope constraints. Both of these constraints are currently written as an SOCP:

$$\|\mathbf{A}\mathbf{U} + \mathbf{b}\| \leq \mathbf{c}^T \mathbf{U} + d \tag{4.103}$$

Numerical solvers however, require these constraints to be first-order conic Domahidi et al. (2013):

$$\mathbf{G}\mathbf{U} \preceq_K \mathbf{h} \tag{4.104}$$

Or in other words, $\mathbf{h} - \mathbf{GU}$ needs to be on cone K:

$$\mathbf{h} - \mathbf{GU} \in K \tag{4.105}$$

Ben-Tal and Nemirovski (2001) give the transcription of Equation 4.103 to the form of Equation 4.104 as follows, where cone K is of the same dimension as vector $\mathbf{h}$:

$$-\begin{bmatrix}\mathbf{c}^T\\ \mathbf{A}\end{bmatrix}\mathbf{U} \preceq_K \begin{bmatrix}d\\ \mathbf{b}\end{bmatrix} \tag{4.106}$$

Thrust Constraint

The thrust constraint up to this point, as given by Table 4.5:

$$\|\mathbf{E}_{\boldsymbol{\tau}}\mathbf{E}_{k,\mathbf{U}}\mathbf{U}\| \leq \mathbf{e}_{\sigma}^T\mathbf{E}_{k,\mathbf{U}}\mathbf{U}, \quad k = 1, \cdots, N \tag{4.107}$$

Using Equation 4.106 to transcribe the SOCP to a conic constraint yields:

$$-\begin{bmatrix}\mathbf{e}_{\sigma}^T\mathbf{E}_{k,\mathbf{U}}\\ \mathbf{E}_{\boldsymbol{\tau}}\mathbf{E}_{k,\mathbf{U}}\end{bmatrix}\mathbf{U} \preceq_K \mathbf{0}_{4\times 1}, \quad k = 1, \cdots, N \tag{4.108}$$

As the four dimensional zero-vector shows, cone K is four dimensional. Another way of seeing this, is that σ is a scalar and $\boldsymbol{\tau}$ is a three dimensional vector, adding up to a four dimensional problem.

Glide-Slope Constraint

The glide-slope constraint is given in Table 4.5 as:

$$\|\mathbf{E}_{\mathbf{xy}}\mathbf{E}_{k,\mathbf{X}}\mathbf{X}\| \leq \mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{X}, \quad k = 1, \cdots, N \tag{4.109}$$

Substituting Equation 4.92, which reads $\mathbf{X} = \mathbf{X}_h + \mathbf{S}_{\mathbf{U}}\mathbf{U}$, into the above equation yields:

$$\|\mathbf{E}_{\mathbf{xy}}\mathbf{E}_{k,\mathbf{X}}\mathbf{S}_{\mathbf{U}}\mathbf{U} + \mathbf{E}_{\mathbf{xy}}\mathbf{E}_{k,\mathbf{X}}\mathbf{X}_h\| \leq \mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{S}_{\mathbf{U}}\mathbf{U} + \mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{X}_h, \quad k = 1, \cdots, N \tag{4.110}$$

Applying the transcription method of Equation 4.106 results in the following notation:

$$-\begin{bmatrix}\mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{S}_\mathbf{U}\\ \mathbf{E}_\mathbf{xy}\mathbf{E}_{k,\mathbf{X}}\mathbf{S}_\mathbf{U}\end{bmatrix}\mathbf{U}\preceq_K\begin{bmatrix}\mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{X}_h\\ \mathbf{E}_\mathbf{xy}\mathbf{E}_{k,\mathbf{X}}\mathbf{X}_h\end{bmatrix},\quad k=1,\cdots,N \tag{4.111}$$

Because the glide-slope constraint compares the magnitude of a two dimensional $\mathbf{xy}$-vector to a scalar, cone K is three dimensional in this case.

Synthesis

Combining all transcriptions of Section 4.3, yields the fully transcribed guidance problem in the conical form, as shown in Table 4.8. Keep in mind however, that the mass constraint is redundant to the thrust constraint and can be omitted, as discussed in Section 4.1.2. An overview of all variables in these equations is provided in Appendix B.

Table 4.8: Guidance problem in conical form.

Minimize	$\mathbf{e}_{N\sigma\Delta t}^T\mathbf{U}$	*Cost function*
Subject to	$-\begin{bmatrix}\mathbf{e}_\sigma^T\mathbf{E}_{k,\mathbf{U}}\\ \mathbf{E}_{\boldsymbol{\tau}}\mathbf{E}_{k,\mathbf{U}}\end{bmatrix}\mathbf{U}\preceq\mathbf{0}_{4\times1},\quad k=1,\cdots,N$	*Thrust constraint*
	$\begin{cases}-(\mathbf{E}_\sigma+\boldsymbol{\mu}_{1N}\mathbf{E}_z\mathbf{S}_\mathbf{U})\mathbf{U}\leq-\boldsymbol{\mu}_{1N}(\mathbf{1}_N-\mathbf{E}_z\mathbf{X}_h+\mathbf{z}_{1N})\\ (\mathbf{E}_\sigma+\boldsymbol{\mu}_{2N}\mathbf{E}_z\mathbf{S}_\mathbf{U})\mathbf{U}\leq\boldsymbol{\mu}_{2N}(\mathbf{1}_N-\mathbf{E}_z\mathbf{X}_h+\mathbf{z}_{2N})\end{cases}$	*Slack variable constraints*
	$\begin{cases}-\mathbf{E}_z\mathbf{S}_\mathbf{U}\mathbf{U}\leq-\mathbf{z}_{2N}+\mathbf{E}_z\mathbf{X}_h\\ \mathbf{E}_z\mathbf{S}_\mathbf{U}\mathbf{U}\leq\mathbf{z}_{1N}-\mathbf{E}_z\mathbf{X}_h\end{cases}$	*Mass constraints*
	$-\begin{bmatrix}\mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{S}_\mathbf{U}\\ \mathbf{E}_\mathbf{xy}\mathbf{E}_{k,\mathbf{X}}\mathbf{S}_\mathbf{U}\end{bmatrix}\mathbf{U}\preceq\begin{bmatrix}\mathbf{e}_d^T\mathbf{E}_{k,\mathbf{X}}\mathbf{X}_h\\ \mathbf{E}_\mathbf{xy}\mathbf{E}_{k,\mathbf{X}}\mathbf{X}_h\end{bmatrix},\quad k=1,\cdots,N$	*Glide-slope constraint*
	$(\mathbf{e}_\sigma^T\mathbf{E}_{k,\mathbf{U}}\cos\theta-\hat{\mathbf{n}}^T\mathbf{E}_{\boldsymbol{\tau}}\mathbf{E}_{k,\mathbf{U}})\mathbf{U}\leq0,\quad k=1,\cdots,N$	*Attitude constraint*
	$\mathbf{E}_{\mathbf{x}(N)}\mathbf{S}_\mathbf{U}\mathbf{U}=\begin{bmatrix}\mathbf{r}_f^T & \dot{\mathbf{r}}_f^T\end{bmatrix}^T-\mathbf{E}_{\mathbf{x}(N)}\mathbf{X}_h$	*Final boundary*

Practical note Table 4.8 shows the mathematical concept of the convex guidance algorithm. This problem can directly be programmed as stated in the table, but the computational effort can be reduced significantly after carefully considering the equations. For example, matrix $\mathbf{E}_{k,\mathbf{X}}$ extracts state vector $\mathbf{x}(k)$ from the stacked states $\mathbf{X}$. Instead of constructing the $\mathbf{E}_{k,\mathbf{X}}$ matrix (of dimension 7 by $7N$) and multiplying this with $\mathbf{X}_h$ or $\mathbf{S}_\mathbf{U}$, it is far more efficient to directly call the subvector from $\mathbf{X}_h$ or submatrix from $\mathbf{S}_\mathbf{U}$ (even if $\mathbf{E}_{k,\mathbf{X}}$ is programmed as a sparse matrix). Considering that the $\mathbf{E}_{k,\mathbf{X}}$ matrix would need to be constructed N times for each trajectory, it is easy to see how this consumes a lot of time. The exact same principle holds for the $\mathbf{E}_{k,\mathbf{U}}$ matrix and most other extraction matrices or vectors $\mathbf{E}$ and $\mathbf{e}$. The time gain strongly depends on the number of temporal nodes N, but this method can easily reduce the computational effort by 90% (this only considers the problem construction, not including the solver time).

4.4 Overview

This section provides a quick overview of the constraints used for the MBP and the AP. The initial- and final-boundary conditions of the trajectory are given in Section 2.4.

During the MBP, the guidance algorithm as given by Table 4.4 is used in combination with the linear time-variant system of Section 4.2.3 to model a spherical gravity field. The linear time-variant system is used during the MBP, because it is a more accurate model than the time-variant gravity vector and there is no need for the definition of a glide-slope.

For the AP, the guidance algorithm as given by Table 4.5 is employed in combination with the time-variant gravity vector of Section 4.2.3. The time-variant gravity vector models a spherical gravity field while it allows the definition of a glide-slope in the landing site frame. Aside from the additional glide-slope constraint, also an attitude constraint is enforced during the AP to point the spacecraft's navigational sensors in the direction of the landing site.

5 Guidance Verification

To verify the transcription of the guidance problem and the implementation of the algorithm in the C++ simulator (discussed in Appendix A), a trajectory provided by Açikmeşe and Ploen (2007) is optimized. The optimization problem used by Açikmeşe and Ploen (2007) concerns a landing on the surface of Mars, for which Table 5.1 shows the exact parameters used. Note that the spacecraft is moving away from the targeted landing site (the origin) as if it has overshot it.

Açikmeşe and Ploen (2007) solved the guidance problem of Table 4.5 using YALMIP (in MATLAB) as a modelling language to transcribe the problem. YALMIP subsequently called the SeDuMi solver to solve the transcribed problem. For this research, the transcribed problem of Table 4.8 was used to directly optimize the trajectory using ECOS in C++. Appendix A discusses the architectural design of the simulator.

Figures 5.1 and 5.3 show the results obtained by Açikmeşe and Ploen (2007), while Figures 5.2 and 5.4 show the results obtained through the C++ implementation. One can directly observe that the obtained figures are very close to the reference figures of Açikmeşe and Ploen (2007). Furthermore, Açikmeşe and Ploen (2007) obtained a fuel cost of 399.5 kg, which agrees with the fuel use shown in Figure 5.2. Note however, that Figures 5.2 and 5.4 do not include $\mathbf{x}_0$, where Figures 5.1 and 5.3 do. Also notice that Açikmeşe and Ploen (2007) define the x-axis as vertical, where the implementation for this research defines the z-axis as vertical. Further note that, although Açikmeşe and Ploen (2007) model a Mars landing, no atmospheric effects are modelled.

Aside from these two small notes, there is only one true difference in the results. Figure 5.3 shows that the attitude θ at the last time step snaps from around 50° to 0°. This sudden change in attitude is also visible in the acceleration and thrust sub-figures. This is caused by the fact the Açikmeşe and Ploen (2007) imposed a final boundary condition on the attitude of the spacecraft, requiring it to be vertical. However, because no rotational dynamics are modelled by the guidance algorithm and no further attitude constraints were imposed during the flight, the attitude is free to change instantaneously. For the C++ implementation in this research,

the final boundary condition for the attitude of the spacecraft was omitted. Therefore, Figure 5.4 does not show this instantaneous change in attitude at touch-down. This final boundary condition for the attitude was deliberately omitted, because it does not improve the validity of the optimal trajectory: 1) it is unrealistic to change the attitude from 50° to 0° within a second and 2) as Figures 5.3 and 5.4 show, the constraint has no significant effect on the rest of the trajectory.

Açikmeşe and Ploen (2007) do not explicitly address the problem of instantaneous attitude changes in their paper. However, they mention including attitude dynamics in the guidance problem as part of future work.

Overall, Figures 5.1 through 5.4 verify both the transcription method of Section 4.3 and the C++ implementation of convex guidance. Furthermore, the results show the potential of the guidance algorithm by optimizing an extreme scenario, where the spacecraft has overshot the landing site and needs to reverse its horizontal speed.

Table 5.1: Verification problem.

t_f	$= 81$ s
$\mathbf{r}_0$	$= \begin{bmatrix} 2 & 0 & 1.5 \end{bmatrix}^T$ km
$\dot{\mathbf{r}}_0$	$= \begin{bmatrix} 100 & 0 & -75 \end{bmatrix}^T$ m/s
$\mathbf{r}_f$	$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ km
$\dot{\mathbf{r}}_f$	$= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ m/s
$\mathbf{g}$	$= \begin{bmatrix} 0 & 0 & -3.7114 \end{bmatrix}^T$ m/s^2
g_0	$= 9.807$ m/s^2
α	$= 4°$
m_0	$= 1905$ kg
I_{sp}	$= 200.48$ s
T	$= 16.573$ kN
T_l	$= 0.3T$
T_h	$= 0.8T$

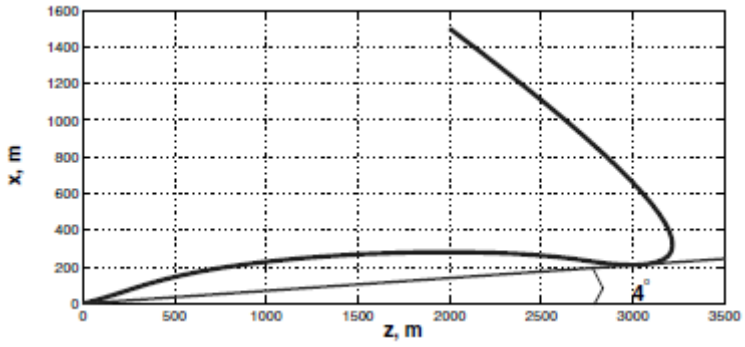


Figure 5.1: The spacecraft's trajectory as provided by Açikmeşe and Ploen (2007).

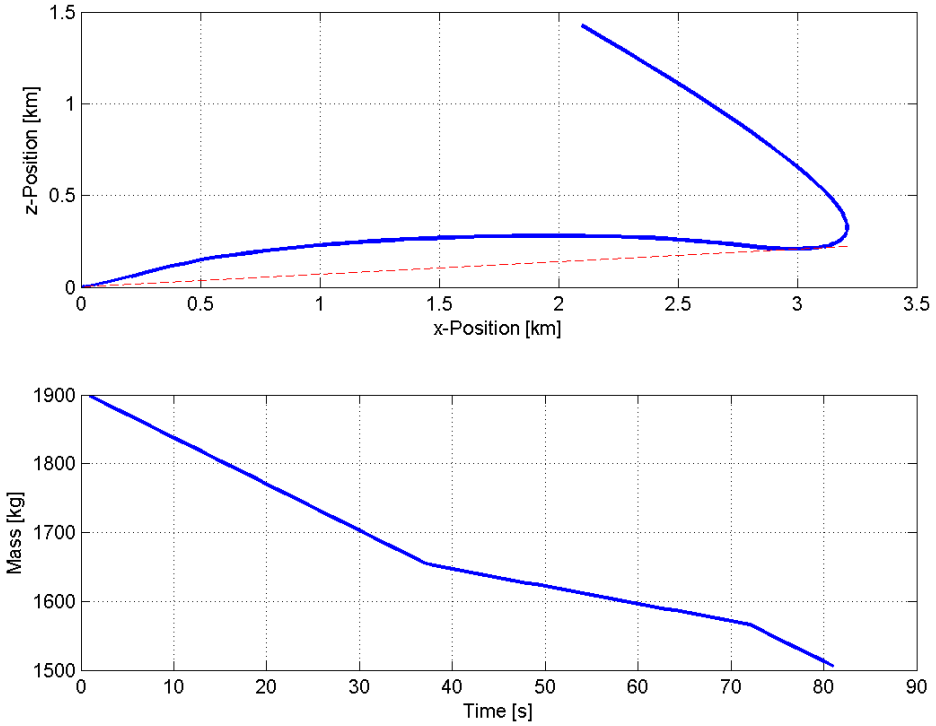


Figure 5.2: The spacecraft's trajectory and mass during the flight.

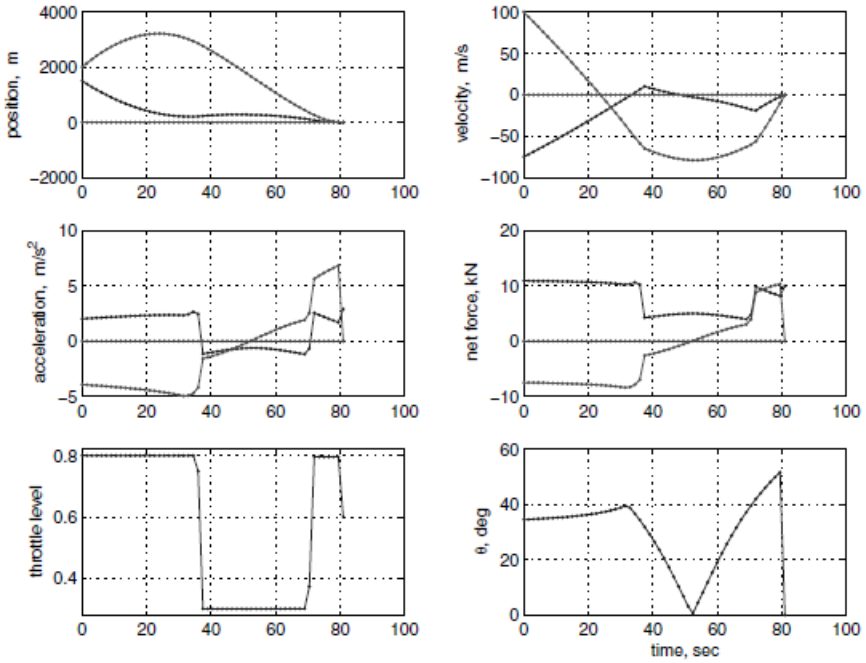


Figure 5.3: The spacecraft's parameters during the flight as provided by Açikmeşe and Ploen (2007).

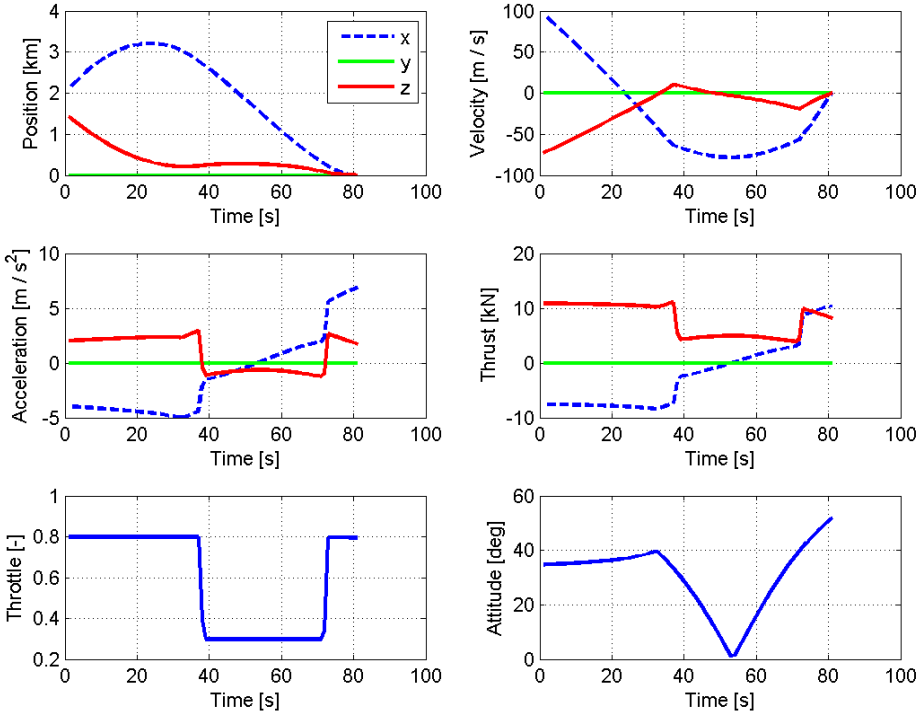


Figure 5.4: The spacecraft's parameters during the flight.

6 Guidance Error Quantification

The guidance algorithm receives the estimated position, velocity, attitude and angular rate of the spacecraft from the navigation algorithm, as shown in Figure 6.1. Using the position and velocity, the guidance algorithm determines the optimal trajectory to the allocated landing site. The trajectory is optimised for fuel usage while conforming to a set of constraints. During the flight, the estimated position of the spacecraft is compared to the optimal trajectory, and corrective actions are imposed on the nominal control values. The resulting attitude is forwarded to the attitude controller.

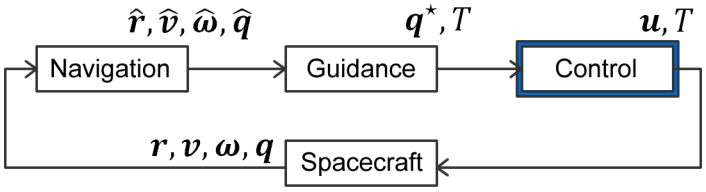


Figure 6.1: Closed guidance, navigation and control loop.

In the derivation of convex guidance in Chapter 4, many assumptions have been made to simplify the trajectory optimisation problem. Because every simplification introduces an error to the result, this section investigates the magnitude of these errors.

Discretisation

When dynamics are calculated numerically, they are often discretised. The resulting errors depend on the problem, the discretisation method and the step size used. To determine the discretisation errors, the guidance system is implemented in open loop. This means that all control commands are executed as they are provided by the guidance algorithm and no corrective manoeuvres are performed. Because the simulator uses a highly accurate integrator, any deviations from the optimal trajectory are caused by discretisation errors in the trajectory optimisation by convex guidance.

Main Braking Phase For running the simulation, the MBP as described in Section 2.4 is used. Figure 6.2 shows the relation of the discretisation step to the trajectory accuracy, computational time and computational memory. The discretisation step is the step size used by the guidance algorithm to linearise the spherical gravity field. The position error refers to the magnitude of the deviation from the AG, as defined by Section 2.4. The execution time is the total time ECOS[1] (Domahidi et al., 2013) uses to optimize the problem (Note that this is an iterative process, as described in Section 4.2.3. For $\Delta t = 5\,\mathrm{s}$, a number of 5 iterations are performed in 0.852 seconds). The memory usage refers to the peak amount of memory used while ECOS solves the problem.

Figure 6.2 and Table 6.1 show that lowering the step size of the guidance algorithm reduces the position error at the AG in a linear fashion. However, they also show that lowering the step size increases the computational time exponentially. For $\Delta t = 5\,\mathrm{s}$, optimization takes 0.852 seconds. For $\Delta t = 2\,\mathrm{s}$, it takes 6.51 seconds. And ultimately for $\Delta t = 1\,\mathrm{s}$, optimization takes 52.2 seconds. Because it concerns optimization of the optimal MBP trajectory from Section 2.4, each optimization in Figure 6.2 consisted of 5 iterations (5 calls to ECOS). The increase in computational time is purely caused by the increase of the problem size, not from additional iterations. These calculations were performed on an Intel i7-3630QM CPU at 2.40 GHz with 8GB of DDR3 SDRAM. Furthermore, note that for $\Delta t = 5\,\mathrm{s}$ ECOS uses about 34 MB of memory, while this grows rapidly to about 1 GB for $\Delta t = 1\,\mathrm{s}$.

Table 6.1: Position error versus computational effort for the main braking phase.

Δt	**5 s**	**4 s**	**3 s**	**2 s**	**1 s**
Execution	0.852 s	1.305 s	2.612 s	6.514 s	52.152 s
Memory	34 MB	51 MB	89 MB	258 MB	1 GB
Error	11.55 m	9.19 m	6.85 m	4.54 m	2.01 m

The error at AG of 11 to 12 metres differs from the result of Gerth (2014), who found an error of just over 6 metres. This difference might be explained by numerical errors, as discussed in Section A.3. Considering the limited computational power onboard of a spacecraft, it is desirable to use $\Delta t = 5\,\mathrm{s}$. Gerth (2014) has researched the sensitivity of the AG conditions and found that an offset of 11 to 12 metres does not significantly change the required propellant mass and does not pose any threat to the remainder of the landing. Because the consequential error of 11 to 12 metres at the AG is acceptable, it is advised to minimize the computational time instead of using valuable seconds to reduce the trajectory error by a couple of metres. The velocity error at the AG approximates 0.05 m/s. Gerth (2014) also shows that this error does not significantly influence the further landing. Therefore, $\Delta t = 5\,\mathrm{s}$ is used for the guidance algorithm during the MBP for the remainder of this study.

[1] Embedded Conic Solver: `https://www.embotech.com/ECOS`

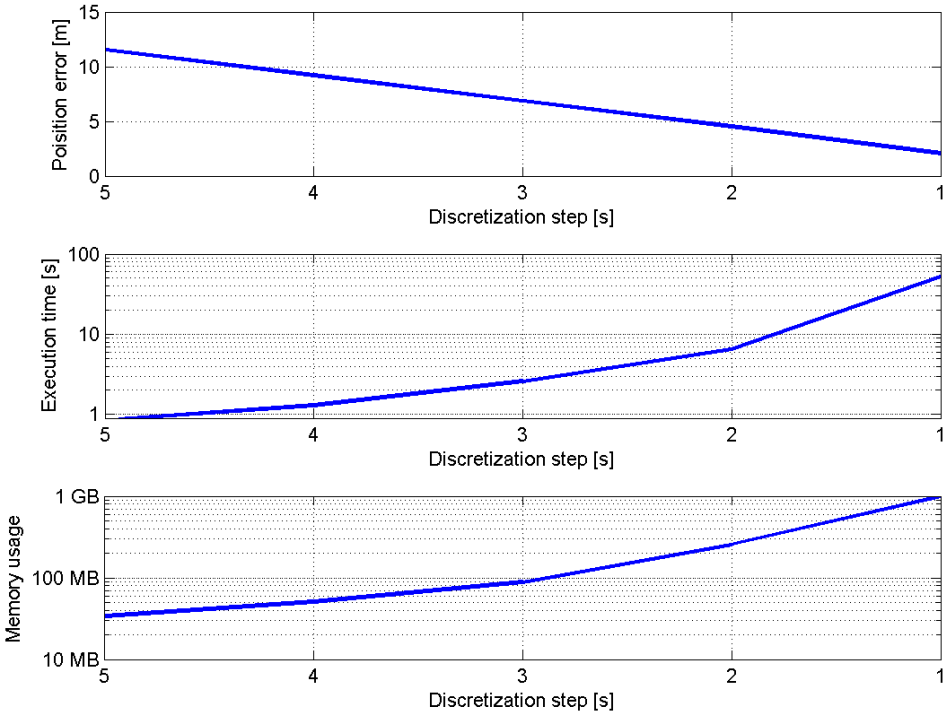


Figure 6.2: Position error and computational expenses for a range of discretization steps.

Approach Phase Because the AP lasts much shorter than the MBP, a relatively small time step can be used from a computational point of view. Simulating the AP as it has been defined in Section 2.4, the optimization only takes 0.034 seconds of execution time (with an Intel i7-3630QM CPU at 2.40 GHz) for $\Delta t = 1$ s. This discretisation step yields errors in the order of centimetres at the TG. Using $\Delta t = 0.5$ s increases the execution time to 0.16 seconds, while the landing error remains in the order of centimetres. Further lowering the step size to $\Delta t = 0.1$ s significantly increases the execution time to 8.6 seconds. Also the memory usage spikes from single or tens of megabytes to well over half a gigabyte, as shown in Table 6.2.

From the results in Table 6.2, $\Delta t = 1$ s is considered the most realistic option for use onboard of a spacecraft. This discretisation step produces an error in the order of centimetres, but lowering the discretisation step reduces the error to a limited extend while increasing the computational efforts significantly.

Note that the position accuracy greatly exceeds that of the MBP, as shown in Section 6. Gerth (2014) showed that errors in the order of metres for the MBP pose no threat to the mission, because the spacecraft is still at over 1 km altitude. However, for the AP a sub-metre accuracy can be important to avoid hazards near the landing site.

Table 6.2: Position error versus computational effort for the approach phase.

Δt	**1 s**	**0.5 s**	**0.1 s**
Execution	0.034 s	0.160 s	8.642 s
Memory	5 MB	17 MB	642 MB
Error	5.42 cm	2.83 cm	-

Lunar Rotation

Even though the Moon is a relatively small and slow rotating body, the Lunar surface at the equator still moves at approximately 4.6 m/s (assuming the Moon's sideral time corresponds to its orbital period of 27.321661 days, as provided by Lissauer and de Pater (2013)), which is not modelled by convex guidance.

Main Braking Phase For the MBP, the guidance algorithm defines the Approach Gate (AG) as a stationary point in inertial space. However, the AG moves through inertial space, because of the Lunar rotation. As a result, the AG has moved a significant distance in the time it takes the spacecraft to fly the MBP. Assuming a flight time of 660 seconds (as defined in Section 2.4), the AG has moved about 3 km, meaning that the spacecraft will still be 3 km away in downrange when it is supposed to reach AG. This example assumes the landing site to be located near the equator. Having a landing site closer to the poles would reduce this deviation to zero. Figure 6.3 shows the MBP position errors at AG, depending on the latitude of the landing site.

Not modelling the Lunar rotation in the guidance algorithm also introduces a velocity error. Because the surface moves at 4.6 m/s, not taking the rotation into account will consequently result in a 4.6 m/s velocity error at the AG. Also this error reduces to zero when landing close to the Moon's poles. Figure 6.3 shows the MBP velocity errors at AG, depending on the latitude of the landing site.

A position error of 3 km at AG is a significant and unacceptable deviation. Gerth (2014) shows that for such deviations, no solutions can be found for the Approach Phase (AP). Luckily, it is very easy to incorporate a model of the Lunar rotation into the guidance algorithm. Because the time of flight is provided as a fixed value when the trajectory is optimized, this time of flight can be used to determine the displacement of the AG during the MBP. Taking a time of flight t_f of 660 seconds for example: instead of setting the final conditions to the position of the AG at t_0, the inertial position of the AG at $t = 660\ \mathrm{s}$ is used for the final condition.

Because the velocity at AG - as defined in Section 2.4 - represents the spacecraft's velocity

with respect to the Lunar surface, a correction of the AG velocity in inertial space is required. The velocity of the AG itself, at t_f (which is 4.6 m/s at the equator), is added to the spacecraft's AG-velocity in inertial space to define the final conditions of the optimization problem.

By aiming at the location where the AG will be after t_f seconds, instead of aiming at the location where it is at the moment of optimization, all errors at the AG are removed. And as a consequence, the required ΔV is reduced slightly because of the increased target velocity at the AG.

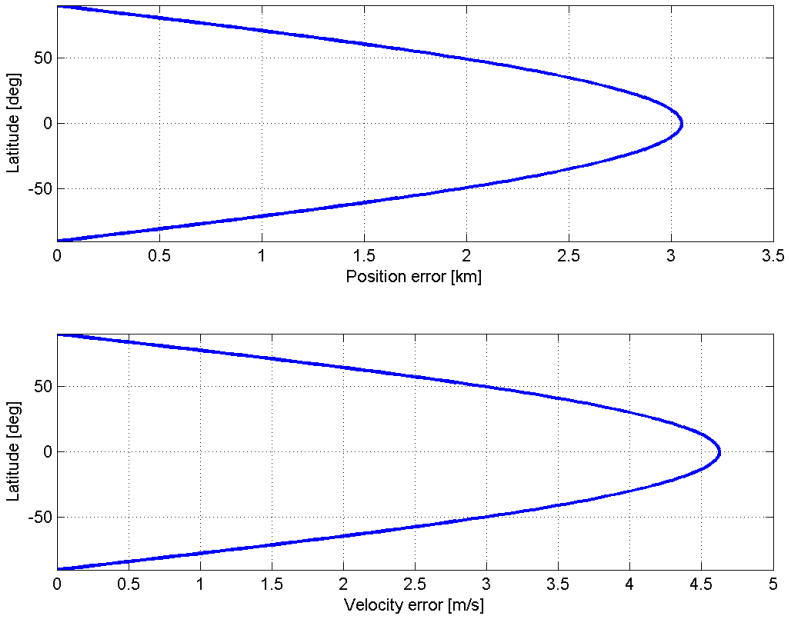


Figure 6.3: Position and velocity errors at AG due to not modelling the Lunar rotation for the MBP for a set of landing site latitudes.

Approach Phase Because the guidance algorithm optimizes the AP in the landing site frame $\mathcal{L}$, the Lunar rotation has a far less significant effect for the AP than it has for the MBP. For example, because the velocity is expressed with respect to the landing site, there is no 4.6 m/s offset in downrange velocity as there was for the MBP. All errors for the AP are introduced by the fact that the $\mathcal{L}$-frame moves through inertial space in a circular motion and slowly rotates at the same time. This motion and rotation of the $\mathcal{L}$-frame with respect to the $\mathcal{I}$-frame cause apparent accelerations in the $\mathcal{L}$-frame, which are not modelled by the guidance algorithm.

Simulating the AP as defined by Section 2.4 yields a position error of 10.5 cm and a velocity error of 5.87 mm/s at the Terminal Gate (TG), both fully caused by the Lunar rotation. Even though the apparent accelerations in the $\mathcal{L}$-frame - caused by the Lunar rotation - could easily be introduced to the guidance algorithm using vector $\mathbf{R}$ in Equation 6.1, there is no need to

do so, because the errors are sufficiently small compared the to required landing accuracy of Section 2.5.

$$\mathbf{X} = \mathbf{S_X X}_0 + \mathbf{S_U U} + \mathbf{S_U G} + \mathbf{S_U R} \tag{6.1}$$

3rd Body Perturbations

Because the spacecraft flies close the Lunar surface, the Moon's gravitational pull is by far the most significant. However, because of the relative small size of the Moon, other celestial bodies may influence the gravitational field to an observable extend, which is not modelled by convex guidance. For example, because of the proximity of the Earth, it perturbs the spacecraft by 10^{-5} m/s^2. The Sun is much farther away than the Earth, but because of its size it still perturbs the spacecraft by 10^{-7} m/s^2. Even though the Earth's gravitational perturbation is about 0.001% of the Moon's gravitational acceleration, a 720 second integration of the perturbation yields a deviation in the order of metres. Integration of the Sun's perturbing acceleration yields a deviation in the order of centimetres. No further celestial bodies are considered, because even Jupiter only produces perturbations in the order of 10^{-12} m/s^2 at its closest approach. The exact equations governing these physics have been discussed in Section 3.5.

The direction of these perturbations changes over time, depending on the geometric composition of the Solar System at that time. The Earth is always close to the Moon's prime meridian and equator. The Sun on the other hand, can be over any longitude of the Moon. For simulating the effects of third-body perturbations, the composition of the Solar System is used on the dates of January 1 of the years 2000 to 2018. This provides a good variety of conditions for the perturbing forces. Figure 6.4 shows the direction of the Sun with respect to the Moon's prime meridian, as seen from the Lunar north pole. The blue circles represent the position of the Earth for the same range of dates. Figure 6.5 provides a closer view of the relative Earth positions, as seen from top-down and side-on[2].

Because the Earth is always approximately over the same Lunar coordinates (due to the tidal lock), the influence of the perturbations also depend on the coordinates of the landing site. Therefore, the effects on the landing performance is investigated for a set of coordinates. Simulations are run for a set of landing sites and epoch combinations (5x19). Table 6.3 shows the average position and velocity errors obtained from these simulations as vectors. Table 6.4 shows the magnitude of the largest errors obtained from these simulations.

[2]Sun and Earth ephemerides obtained from NASA JPL's HORIZONS system at `http://ssd.jpl.nasa.gov/horizons.cgi`.

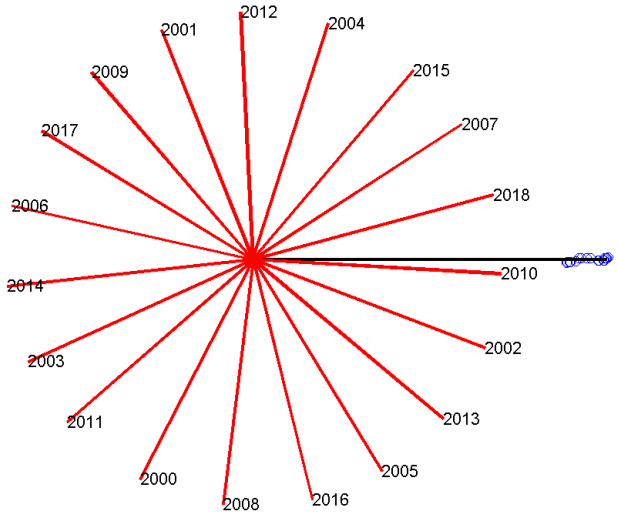


Figure 6.4: Directions of the Sun (red lines) and positions of the Earth (blue circles) on the first of January from 2000 to 2018, with respect to the Moon's prime meridian, as seen from the Lunar north pole.

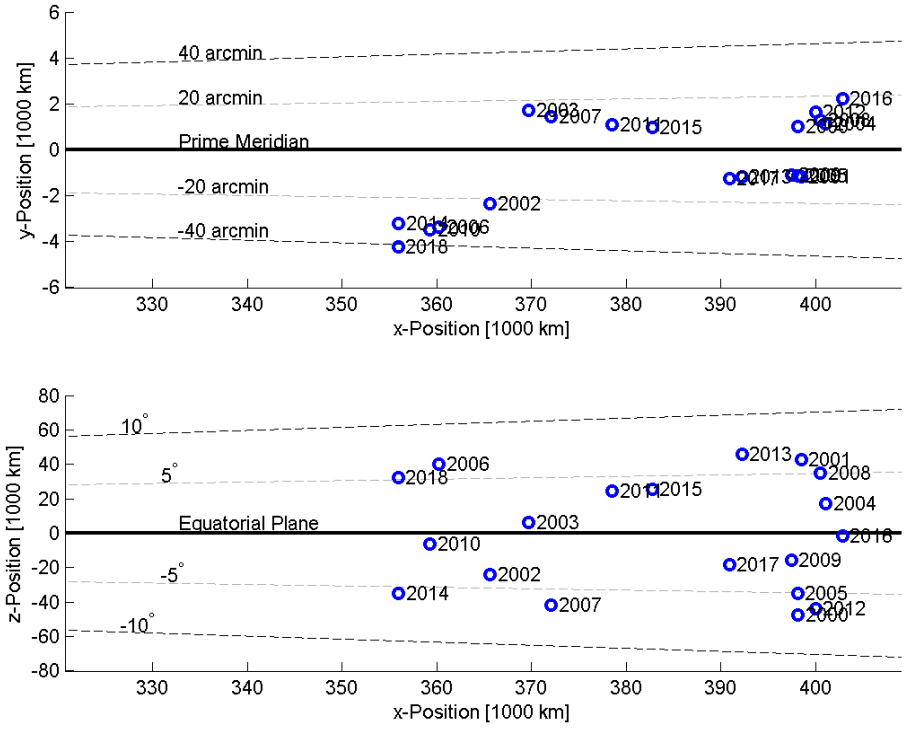


Figure 6.5: Earth's position on January first for a range of years, expressed in the Corotating frame $\mathcal{C}$.

Table 6.3 shows that landing at 0° or 180° longitude produces the largest errors. Furthermore, it is interesting to note that when landing at 0° or 180° longitude the spacecraft ends up

too high and overshoots the target, while landing at 90° or 270° longitude makes the spacecraft end up too low and end up short of the target. For the north pole, the spacecraft ends up lower than targeted and overshoots the gate.

Overall, Tables 6.3 and 6.4 show that the position errors are in the order of metres and the velocity errors are in the order of tens of mm/s for the MBP. For the AP, the position errors are in the order of centimetres while the velocity errors are about mm/s.

Table 6.3: Average position and velocity errors at AG and TG.

		Main Braking Phase		**Approach Phase**	
Lat.	**Lon.**	**Position error**	**Velocity error**	**Position error**	**Velocity error**
		m	mm/s	m	mm/s
0°	0°	$\begin{bmatrix} -0.005 \\ 0.439 \\ 5.625 \end{bmatrix}$	$\begin{bmatrix} -0.016 \\ 0.747 \\ 18.121 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \\ 0.033 \end{bmatrix}$	$\begin{bmatrix} -0.001 \\ -0.004 \\ 1.266 \end{bmatrix}$
0°	90°	$\begin{bmatrix} 0.007 \\ -0.937 \\ -2.831 \end{bmatrix}$	$\begin{bmatrix} 0.014 \\ -1.744 \\ -9.156 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \\ -0.017 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.008 \\ -0.643 \end{bmatrix}$
0°	180°	$\begin{bmatrix} 0.005 \\ 0.427 \\ 5.553 \end{bmatrix}$	$\begin{bmatrix} 0.016 \\ 0.747 \\ 18.121 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \\ 0.033 \end{bmatrix}$	$\begin{bmatrix} 0.001 \\ -0.004 \\ 1.266 \end{bmatrix}$
0°	270°	$\begin{bmatrix} 0.001 \\ -0.961 \\ -2.795 \end{bmatrix}$	$\begin{bmatrix} 0.001 \\ -1.826 \\ -9.063 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \\ -0.017 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ -0.001 \\ -0.638 \end{bmatrix}$
90°	0°	$\begin{bmatrix} -0.009 \\ 0.512 \\ -2.785 \end{bmatrix}$	$\begin{bmatrix} -0.029 \\ 1.005 \\ -9.032 \end{bmatrix}$	$\begin{bmatrix} 0.000 \\ 0.000 \\ -0.017 \end{bmatrix}$	$\begin{bmatrix} -0.003 \\ 0.000 \\ -0.632 \end{bmatrix}$

Table 6.4: Worst position and velocity errors at AG and TG.

		Main Braking Phase		**Approach Phase**	
Lat.	**Lon.**	**Position error**	**Velocity error**	**Position error**	**Velocity error**
		m	mm/s	m	mm/s
0°	0°	7.009	22.754	0.041	1.595
0°	90°	3.651	11.463	0.021	0.791
0°	180°	6.907	22.451	0.041	1.575
0°	270°	3.653	11.479	0.021	0.790
90°	0°	3.571	11.440	0.021	0.805

Irregular Gravity Field

The gravity field of the Moon is just as irregular as its surface. Figure 6.6 shows the deviation of the gravity field from the nominal spherical model, where red indicates additional gravitational pull and blue represents a sub-nominal gravitational attraction. The largest of craters are clearly visible in the gravitational field of the Moon, making the gravity field highly irregular. Although Figure 6.6 shows the Lunar gravity field to be very irregular, convex guidance assumes a perfect spherical gravity field, mainly out of concern for computational effort. For the generation of Figure 6.6, the gravity model provided by NASA's GRAIL mission was used[3].

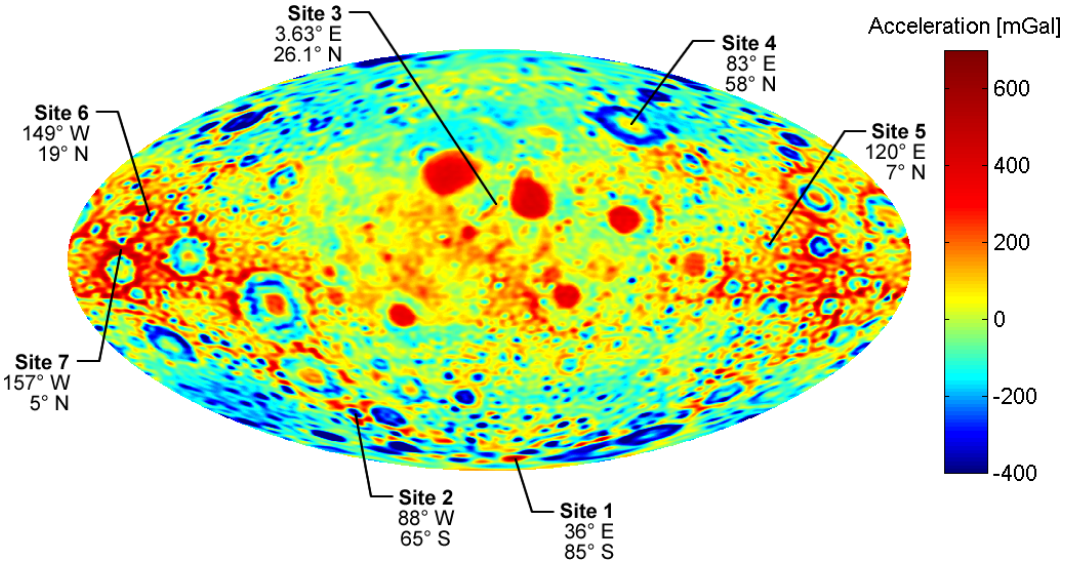


Figure 6.6: Spherical harmonics model of the Moon's gravity field up to degree and order 140, where the 7 test landing sites are indicated.

How to model gravitational irregularities has been discussed in Section 3.5. The gravity field can theoretically be modelled exactly by using an infinite degree and order. Practically, it is impossible to use an infinite sum and a truncation point needs to be determined. To determine the required degree and order of the gravity model, simulations are run for 7 landing sites. For every landing site, simulations are performed from a 2x2 gravity field up to a 150x150 field. For each run, the deviation is compared to the landing trajectory in a perfect spherical gravity field.

In selecting the 7 landing sites of Figure 6.6, the Mollweide projection is used to prevent size distortions near the Lunar poles. The shapes are slightly influenced, as they are slanted towards the poles, but the relative sizes (surface area) of features remain to scale throughout the map.

[3]Model freely available at `http://pds-geosciences.wustl.edu/dataserv/gravity_models.htm`.

Landing site 1 is located at a gravity hot-spot, close to two areas of considerably lower gravity. Landing site 2 is located at a low-gravity area, inside a ring of higher gravity. Landing site 3 is the landing site of Apollo 15, near a ridge of higher gravity. This relatively thin ridge only appears when using a high degree model. Landing site 4 is located on a high-gravity spot surrounded by a low-gravity ring. Site 5 is located in a low-gravity pit in a high-gravity area. Site 6 is also located in a low-gravity pit surrounded by a high-gravity area, but also close to a bigger crater. Finally, landing site 7 is located on a high-gravity ridge in between two fairly sized craters, where the gravity-differential with the south crater is largest.

Figure 6.7 shows the change in position - for landing site 1 - caused by the order and degree of the spherical harmonics used, with respect to a perfect spherical gravity field. The figure shows that the influence strongly fluctuates at lower degrees. For the MBP the deviation only starts to stabilise at a gravity field of degree and order 100. For the AP this is even worse and the deviation does not stabilise until degree and order 140. Landing site 1 is shown in Figure 6.7, because it is the worst case compared to the other landing sites. For the other sites the simulations either stabilised at the same or at lower degrees and orders, as shown in Appendix C. Therefore, it is decided to use a gravity field of degree and order 100 during the MBP and 140 during the AP.

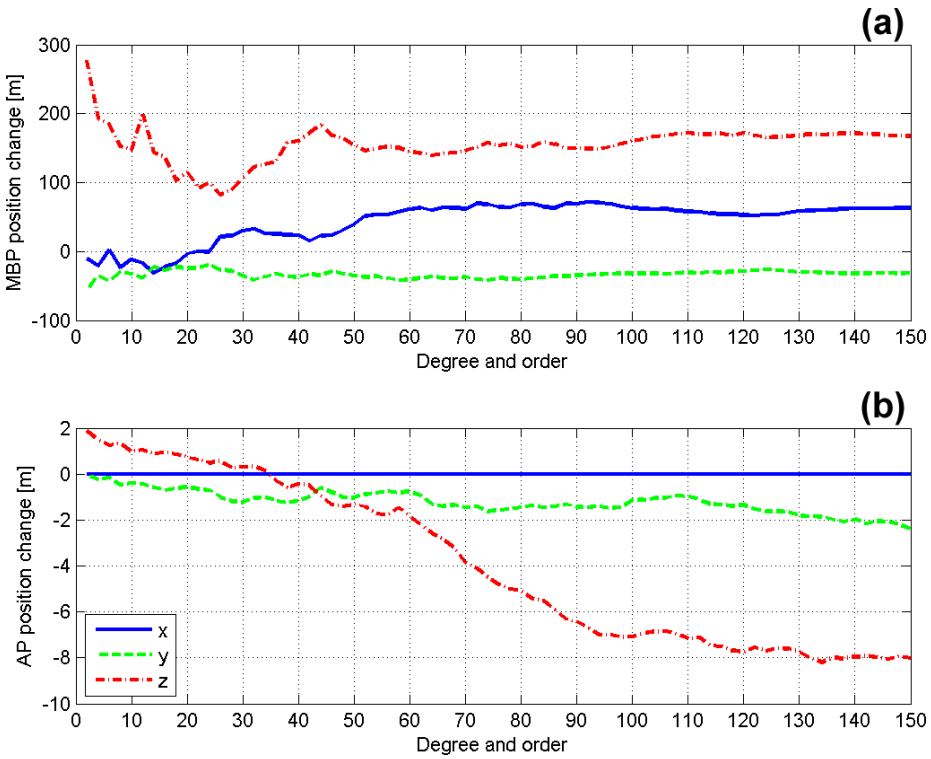


Figure 6.7: Change in landing position for landing site 1, due to modelling a higher degree gravity field. The change is with respect to a perfect spherical gravity field. **a**: Contribution from the main braking phase. **b**: Contribution from the approach phase.

Because of numerical limitations, Figure 6.7 only goes up to degree and order 150. Computation of the Legendre polynomials involve the factorial of the degree of the gravity field, see Equations 3.23c and 3.24. The largest factorial that can be represented by the double-precision floating-point format is $170! \approx 7.257 \cdot 10^{306}$. Beyond the factorial of 170, double-precision floating-point variables overflow and return *infinity*. However, because of mathematical operations and depending on the landing coordinates, double-precision variables regularly overflow when modelling lower degree and order gravity fields, too. Therefore, degree and order 150 is the highest accuracy at which the current implementation can be used, without regularly overflowing variables.

Overflowing of double-precision floats, can be prevented by using the quadruple-precision floating-point format, which allows number representations up to 10^{4932}. In that case the highest possible factorial is $1754! \approx 1.979 \cdot 10^{4930}$ (which is well beyond the full degree and order of any current Lunar gravity model). However, the quadruple-precision floating-point format is not supported by C++ by standard at this point.

Global results

To get a good sense of the influence of an irregular gravity field on the landing performance, landings are simulated for the entire Lunar surface. A gird is placed over the Moon with a resolution of 1° in both longitude and latitude. The trajectory as described by Section 2.4 is flown in each simulation.

Figure 6.8 shows the errors resulting from the irregularities in the gravity field for the main braking phase and the approach phase. The sub-figures show the individual x, y and z-components of the position error. Note that the z-sub-figures in Figure 6.8 are approximately the inverse of Figure 6.6. Additional gravity (red in Figure 6.6) results in a lower altitude (blue in Figure 6.8) and vice versa. Also note that the largest position error is in the z-direction. The x and y-components show smaller errors. Furthermore, the maps are shown in a square projection to easier read the results in terms of longitude and latitude.

It is also interesting to note that the MBP figures seem to be a smudged version of the AP figures and that many small bumps in the gravity field have been smoothened out. This is caused by the fact that during the MBP, the spacecraft travels almost 630 km (5.8% of the Moon's circumference) at high speeds, while during the AP the spacecraft only travels 1.3 km at relatively low speeds. Because the spacecraft is simulated to orbit in an eastward direction, the gravity field features are also stretched in eastward direction. Furthermore, because landing at large (either positive or negative) latitudes requires the descent orbit to have a significant inclination, the streaks are also pointing away from the equator. This is best

visible in the z-components of Figure 6.8, because it has the highest contrast.

Similar figures as Figure 6.8 are obtained for the velocity errors. These plots are almost identical, except that the colour values range from -2 m/s to 2 m/s for the MBP and from -0.2 m/s to 0.2 m/s for the AP. Because the velocity errors show the same exact pattern, they are not shown here.

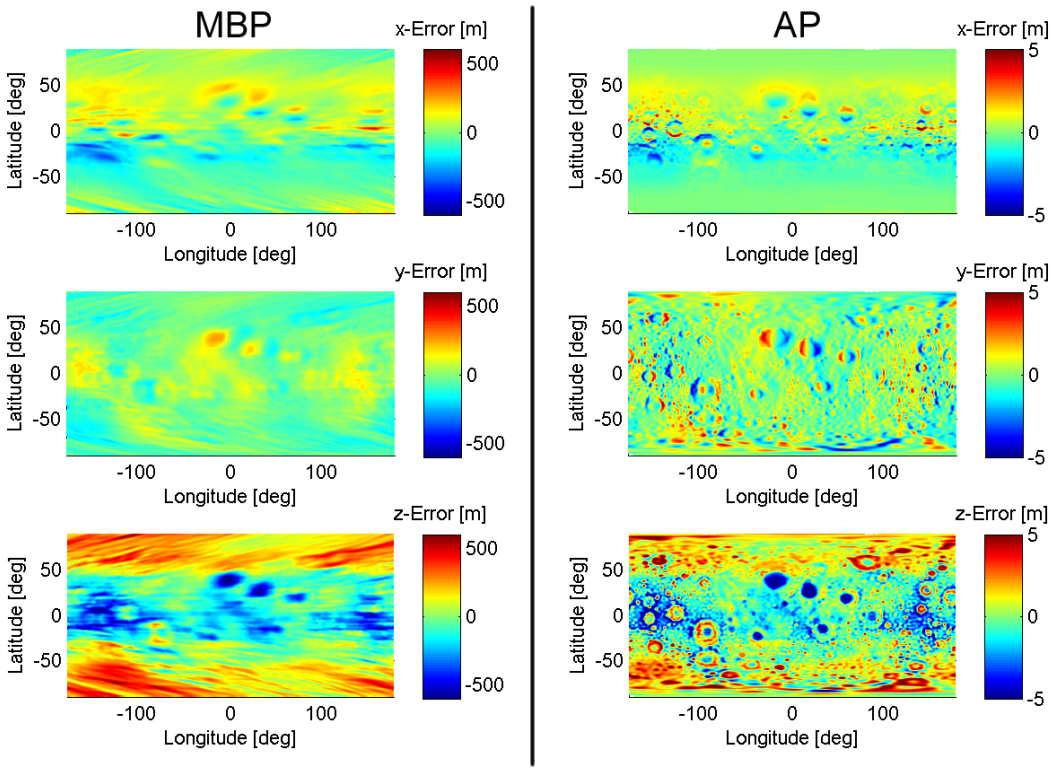


Figure 6.8: Components of the position error caused during the MBP and the AP. **x** = South. **y** = East. **z** = Zenith.

Figure 6.9b shows the magnitude of the position error for the MBP, while Figure 6.9c shows the magnitude of the position error for the AP. Figure 6.9a shows the perturbing forces again for a 140x140 gravity field, but this time in a square projection. Sub-figure *a* is added to show that the errors closely resemble the irregularities in the gravitational field. Please note however, that for sub-figure *a* green is zero and blue is negative, while for sub-figures *b* and *c* green is positive and blue is zero (because there is no negative magnitude). Figure 6.9 again shows the sub-figure *b* for the MBP is a smudged version of sub-figure *c* for the AP, where the streak point eastward and away from the equator.

The magnitude plots of the velocity errors are again omitted because they are almost identical to the position error plots in Figure 6.9. The only true differences are the colour scales, which go up to 2 m/s for the MBP and 0.25 m/s for the AP.

In conclusion, Figure 6.8 shows that the biggest component of the errors are in the local vertical direction. It further shows that at the Lunar poles the spacecraft usually ends up too high, while close to the equator it tends to end up closer to the ground than intended.

Figure 6.9 shows that the maximum position error is 737 m for the MBP, while the mean position error is 212 m. For the AP the maximum position error is 8.70 m, with a mean position error of 2.00 m. The maximum velocity error for the MBP is 2.37 m/s, with a mean of 0.67 m/s. For the AP the maximum velocity error is 0.338 m/s with a mean value of 0.077 m/s.

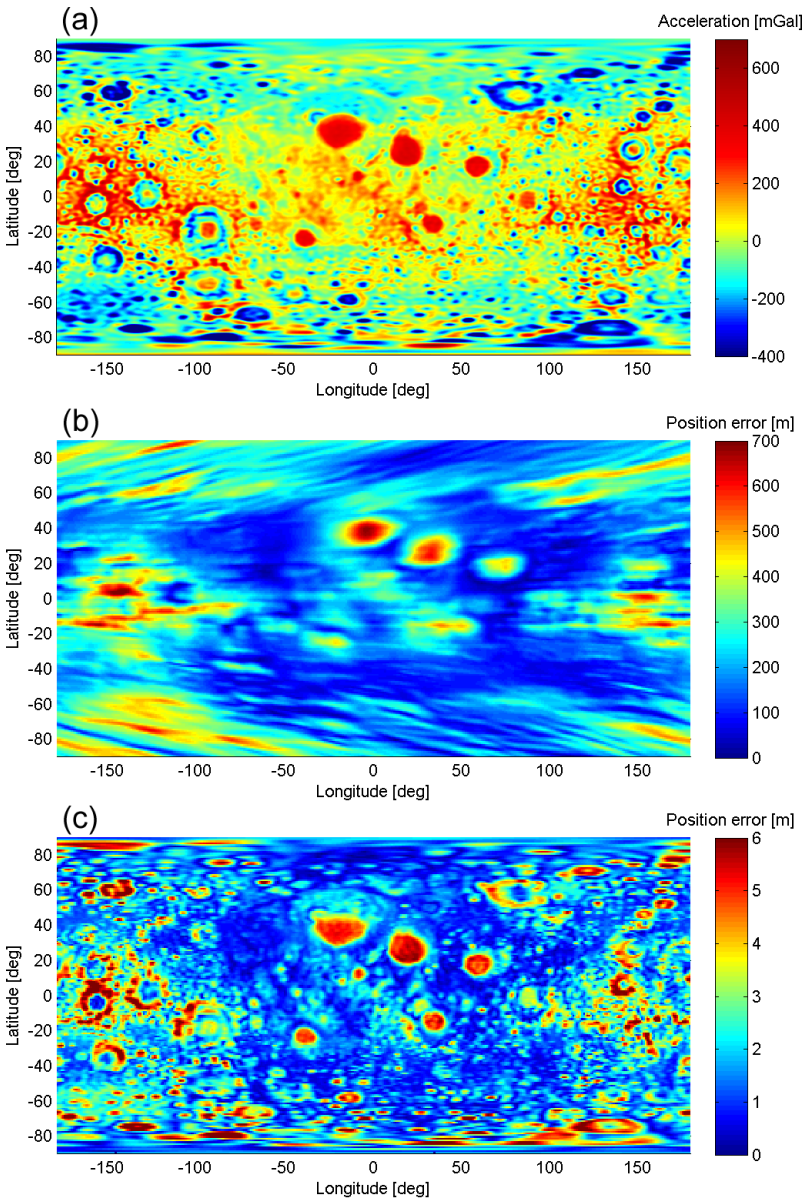


Figure 6.9: **a**: Perturbations in gravitational field up to degree and order 140. **b**: Magnitude of poisition error for the MBP. **c**: Magnitude of position error for the AP.

7 Control

The control algorithm of a spacecraft receives manoeuvre commands from the guidance system in the form of thrust vectors. The control algorithm needs to orient the spacecraft to align with the thrust vector and activate the thrusters to match the magnitude of the thrust vector. At the same time, the control algorithm also makes corrections when the spacecraft deviates from the optimised path.

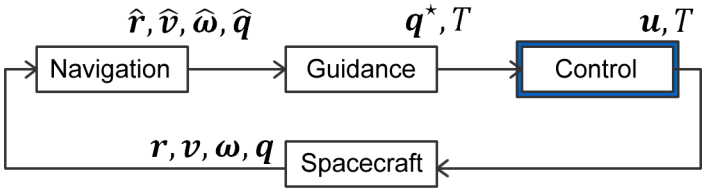


Figure 7.1: Closed guidance, navigation and control loop.

7.1 Attitude Controller

This study is not looking into advanced attitude control algorithms, but it is investigating the performance of convex guidance in a GNC-loop instead. Therefore, to establish a baseline on the influence of convex guidance on attitude control, only a simple Proportional-Derivative (PD) controller is used. Because the attitude of the spacecraft is represented by a quaternion, as discussed in Section 3.3, a quaternion feedback controller is used.

7.1.1 Quaternion Feedback

A Quaternion Feedback (QF) controller is a linear PD controller that utilizes quaternions for the expression of the spacecraft's attitude instead of Euler angles. A short derivation follows, as provided by Wie et al. (1988) and Wie (1998). This approach considers Euler's equations

of rotational motion about a body-fixed axis system, as shown in Equation 7.1a.

$$\mathbf{I}\dot{\boldsymbol{\omega}} = -\boldsymbol{\Omega}\mathbf{I}\boldsymbol{\omega} + \mathbf{u} \tag{7.1a}$$

$$\boldsymbol{\Omega} = [\boldsymbol{\omega}\times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{7.1b}$$

Here, $\mathbf{I}$ represents the spacecraft's inertia matrix, $\boldsymbol{\omega}$ represents the angular velocity vector, $\mathbf{u}$ represents the control torque vector and $\boldsymbol{\Omega}$ is a skew-symmetric matrix that replaces the cross-product of $\boldsymbol{\omega}$. In this equation, $\boldsymbol{\Omega}\mathbf{I}\boldsymbol{\omega}$ is the gyroscopic coupling torque.

By using eigenaxis rotations, any change of attitude can be represented by a single rotation, which in this case is expressed by a quaternion of rotation. The quaternion of rotation has been defined in Section 3.3, but is repeated below for convenience.

$$\bar{\mathbf{q}}(\mathbf{e},\theta) = \begin{bmatrix} \mathbf{q}(\mathbf{e},\theta) \\ q_4(\theta) \end{bmatrix} = \begin{bmatrix} \mathbf{e}\sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{7.2a}$$

$$\mathbf{e} = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^T \tag{7.2b}$$

Here, $\mathbf{e}$ is a unit vector representing the eigenaxis and θ is the angle of rotation around the eigenaxis.

In controlling the attitude of a spacecraft, the error in the attitude is expressed as an quaternion, which is attempted to reduce to zero angle. For this expression, the error quaternion $\begin{bmatrix} \mathbf{q}_e^T & q_{e_4} \end{bmatrix}^T$ is obtained from the commanded attitude quaternion $\begin{bmatrix} \mathbf{q}_c^T & q_{c_4} \end{bmatrix}^T$ and the actual attitude quaternion $\begin{bmatrix} \mathbf{q}^T & q_4 \end{bmatrix}^T$ as follows:

$$\bar{\mathbf{q}}_e = \begin{bmatrix} q_{c_4}\mathbf{I}_{3\times 3} - [\mathbf{q}_c\times] & -\mathbf{q}_c \\ \mathbf{q}_c^T & q_{c_4} \end{bmatrix} \bar{\mathbf{q}} \tag{7.3}$$

$$\begin{bmatrix} q_{e_1} \\ q_{e_2} \\ q_{e_3} \\ q_{e_4} \end{bmatrix} = \begin{bmatrix} q_{c_4} & q_{c_3} & -q_{c_2} & -q_{c_1} \\ -q_{c_3} & q_{c_4} & q_{c_1} & -q_{c_2} \\ q_{c_2} & -q_{c_1} & q_{c_4} & -q_{c_3} \\ q_{c_1} & q_{c_2} & q_{c_3} & q_{c_4} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{7.4}$$

Using these definitions, a linear state feedback controller is defined by Wie et al. (1988), as shown in Equation 7.5, where $\mathbf{K}$ and $\mathbf{D}$ are gain matrices and $\mathbf{u}$ is the control torque vector.

$$\mathbf{u} = \boldsymbol{\Omega}\mathbf{I}\boldsymbol{\omega} - \mathbf{D}\boldsymbol{\omega} - \mathbf{K}\mathbf{q}_e \tag{7.5}$$

Because the error quaternion $\mathbf{q}_e$ represents a rotation and $\boldsymbol{\omega}$ is the angular-rate vector of the spacecraft, this linear state feedback controller is a PD controller.

Wie et al. (1988) continue from Equation 7.5 to show how the parameters $\mathbf{K}$ and $\mathbf{D}$ of the controller are selected. They set $\mathbf{K} = k\mathbf{I}$ and $\mathbf{D} = d\mathbf{I}$ to achieve eigenaxis rotations, where k and d are positive scalars. An eigenaxis rotation is the most efficient way of rotating from one orientation to the next. Furthermore, using the definition for the quaternion from Equation 7.2 ($\mathbf{q} = \mathbf{e}\sin(\theta/2)$), yields the following expression for the control torque:

$$\mathbf{u} = \boldsymbol{\Omega}\mathbf{I}\boldsymbol{\omega} - d\mathbf{I}\boldsymbol{\omega} - k\mathbf{I}\mathbf{e}\sin(\theta/2) \tag{7.6}$$

Substituting this expression for the control torque into the equation of rotational motion (Equation 7.1a), yields the equality of Equation 7.7.

$$\mathbf{I}\dot{\boldsymbol{\omega}} = -d\mathbf{I}\boldsymbol{\omega} - k\mathbf{I}\mathbf{e}\sin(\theta/2) \tag{7.7}$$

Notice that the gyroscopic coupling torque terms $\boldsymbol{\Omega}\mathbf{I}\boldsymbol{\omega}$ of the two equations have cancelled. This means that the controller perfectly counteracts the coupled torques. However, in practice the gyroscopic coupling torque is never fully known. Therefore, this becomes a small error source.

Because the attitude correction concerns an eigenaxis rotation, $\boldsymbol{\omega} = \dot{\theta}\mathbf{e}$ and Equation 7.7 can be rewritten as follows:

$$\left(\ddot{\theta} + d\dot{\theta} + k\sin(\theta/2)\right)\mathbf{I}\mathbf{e} = \mathbf{0} \tag{7.8}$$

Because $\mathbf{I}\mathbf{e} \neq \mathbf{0}$, the factor $\ddot{\theta} + d\dot{\theta} + k\sin(\theta/2)$ must be zero instead. Furthermore, approximating the sine of small attitude errors by its angle, $\sin(\theta/2) \approx \theta/2$, yields the expression of Equation 7.9.

$$\ddot{\theta} + d\dot{\theta} + k\theta/2 = 0 \tag{7.9}$$

A second order system can be written as:

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0 \tag{7.10}$$

Here, ζ is the damping ratio, for which the ideal ratio is used $\zeta = \frac{1}{2}\sqrt{2}$. Furthermore, ω_n is the natural frequency of the undamped system. Comparing Equations 7.9 and 7.10, the definitions of d and k become obvious as shown in Equations 7.11a and 7.11b.

$$d = 2\zeta\omega_n \tag{7.11a}$$

$$k = 2\omega_n^2 \tag{7.11b}$$

The only variable that still needs to be determined is the natural frequency ω_n. By rewriting the natural frequency to the settling time, a more intuitive parameter can be tuned. Nise (2008) approximates the settling time of the system by $t_{set} = 4/(\zeta\omega_n)$. Rewriting $t_{set} = 4/(\zeta\omega_n)$, obtains Equation 7.12. Substitution of this result into Equations 7.11a and 7.11b, yield direct expressions for d and k as a function of the desired settling time and the damping ratio:

$$\omega_n = \frac{4}{\zeta t_{set}} \tag{7.12}$$

$$d = \frac{8}{t_{set}} \tag{7.13}$$

$$k = \frac{32}{\zeta^2 t_{set}^2} \tag{7.14}$$

Remember that d and k were used to define $\mathbf{D}$ and $\mathbf{K}$, which where the gain matrices of the controller of Equation 7.5 (repeated below).

$$\mathbf{u} = \mathbf{\Omega I}\boldsymbol{\omega} - \mathbf{D}\boldsymbol{\omega} - \mathbf{K}\mathbf{q}_e$$

$$\mathbf{D} = d\mathbf{I}$$

$$\mathbf{K} = k\mathbf{I}$$

7.1.2 Results

It is desirable to have a small settling time, for quick attitude control. It was found that $t_{set} = 0.5$ s gives good performance in tracking the attitude. However, for movement over large angles, a longer settling time can be expected. Therefore, when the angle is greater than 1°, a settling time of $t_{set} = 4$ s is used. These parameters are used for the remainder of this study. Furthermore, the attitude controller runs at a frequency of 30 Hz.

To determine the influence of convex guidance on the attitude control problem, simulations are run where the attitude controller is commanded to track the attitude of the optimal trajectory, as it is optimised by convex guidance. To be clear, at this stage there are no corrections made by a trajectory tracker when the spacecraft deviates from the optimal path.

Because a zero-order hold function was assumed for the control vector (specific thrust $\boldsymbol{\tau}$) in Section 4.1.3, the optimal attitude is discontinuous. Therefore, the spacecraft is commanded to hold a specific attitude for a short time, after which it is commanded to instantly change its attitude, holding the new attitude for a short time as well before instantly changing again, as shown in Figure 7.2. This figure shows that the spacecraft has to change its angular rate repeatedly and as a result, lags behind the command. Note that the magnitude of thrust remains constant during the time period shown in Figure 7.2. The z-axis acceleration increases because the spacecraft slowly up rights itself, increasing the thrust's z-component.

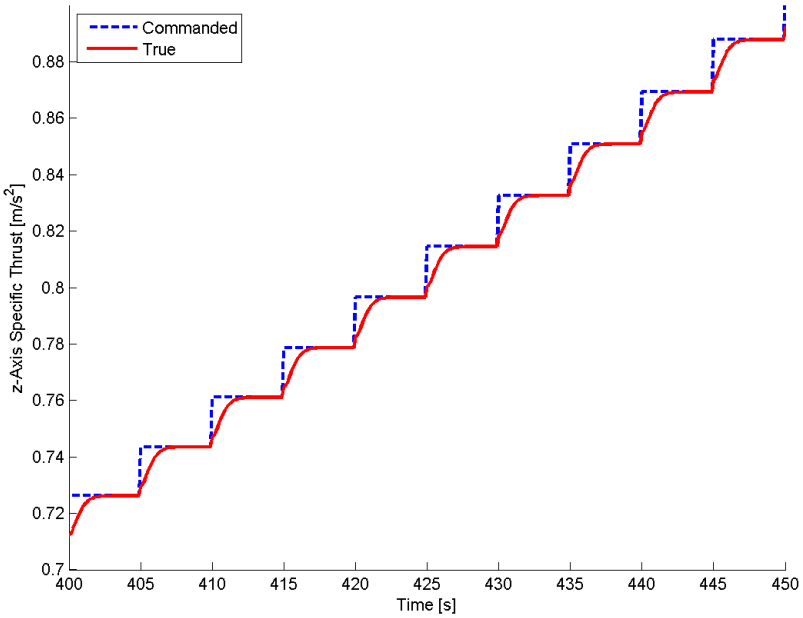


Figure 7.2: Spacecraft following a discontinuous command attitude. This figure shows the z-axis of the specific thrust increasing, as the spacecraft slowly uprights itself.

Having to increase and decrease the angular rate of the spacecraft repeatedly requires a lot of propellant. To reduce the control effort, the discontinuous attitude command is linearly interpolated at the mid-points of a command, as shown in Figure 7.3. By interpolating through the mid-points, the errors made at the beginning of a command approximately cancel the errors made at the end of a command. Due to this interpolation, the commanded attitude is much smoother, as the spacecraft does not have to build and break its angular momentum repeatedly. Because the spacecraft can maintain a near constant angular rate, the interpolated attitude command uses less than 2% of the attitude control effort compared to the discontinuous attitude commands. This is a very significant save, from only a small alteration.

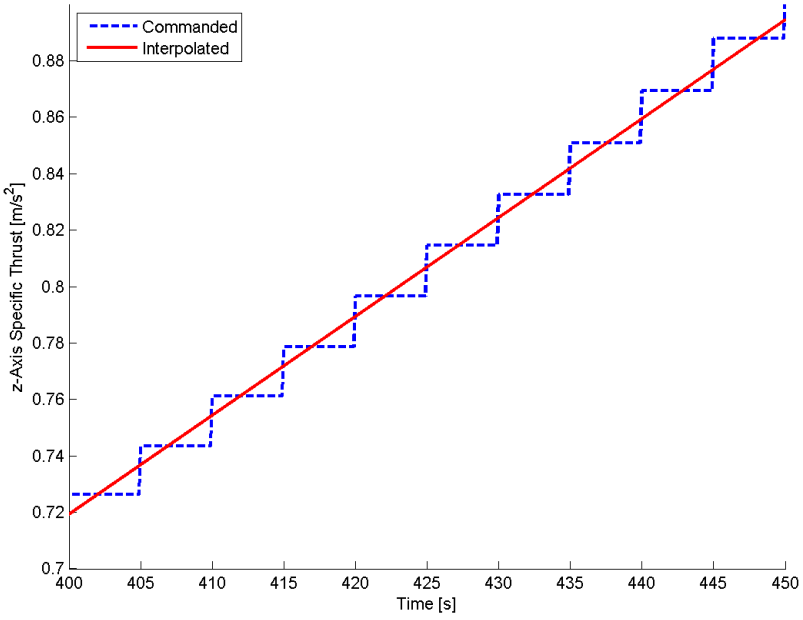


Figure 7.3: Spacecraft following the interpolated version of the discontinuous command attitude. This figure shows the z-axis of the specific thrust increasing, as the spacecraft slowly uprights itself.

Instead of interpolating the optimal control, the zero-order hold function in convex guidance can also be replaced by a higher order function. For example, Bhagat (2016) uses the trapezoidal rule to piecewise linearise the optimal control function.

Another way to resolve the discontinuous nature of the optimal control, is to reduce the discretisation step in the guidance algorithm (which is 5 seconds for the MBP and 1 second for the AP). However, Section 6 shows that reducing these step sizes dramatically increases the computational loads. Therefore, this is not considered a viable option in this case.

In conclusion, because it is not feasible to increase the frequency of the guidance algorithm for this problem, it is advised to either interpolate the optimal control, or to change the optimal

control to a first-order function in the convex guidance algorithm. Due to time constraints, the interpolation method is used for the remainder of this study.

7.2 Trajectory Tracker

Thus far, all simulations relied on an open loop. The spacecraft executed the control commands directly provided by convex guidance, as being the optimal trajectory. This way, the errors of discretisation, Lunar rotation, third body perturbations, gravity field irregularities and attitude dynamics have been determined. The next step is to develop a trajectory tracker to make corrections when the spacecraft starts to deviate from the optimal trajectory.

7.2.1 Controller

This study is not looking into advanced tracking algorithms, but it is investigating the performance of convex guidance in a GNC-loop instead, of which the trajectory tracker is a small part. Therefore, to establish a baseline on the influence of convex guidance on trajectory tracking, the tracker is based on a PD-controller. In order to do so, the state-space system of Equation 7.15 is used, where the state is defined as $\mathbf{x} = \begin{bmatrix} \mathbf{r}^T & \dot{\mathbf{r}}^T \end{bmatrix}^T$.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{7.15}$$

The system matrix is given by Equation 7.16, where μ is the Lunar gravitational parameter and r is distance of the spacecraft with respect to the Moon's centre of mass. It was found that r can either be set to the radius of the initial condition, or the radius of the targeted landing site without having a significant effect on the controller. Therefore, $\mathbf{A}$ is assumed to be constant.

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \frac{-\mu}{r^3}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times6} \tag{7.16}$$

Control vector $\mathbf{u}$ is defined as an acceleration (as discussed for convex guidance in Chapter 4). This ensures that the eigenvalues do not change during the flight, fixing the characteristics of the system. By defining $\mathbf{u}$ as an acceleration, input matrix $\mathbf{B}$ is given as follows:

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{I}_{3\times3} \end{bmatrix} \in \mathbb{R}^{6\times3} \tag{7.17}$$

With $\mathbf{A}$ assumed to be constant, Equation 7.15 is a linear equation. Developing a perturbation equation from this yields the following expression:

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \tag{7.18}$$

Here, $\Delta\mathbf{x}$ is the state deviation and $\Delta\mathbf{u}$ is the control effort. Matrices $\mathbf{A}$ and $\mathbf{B}$ are given by Equations 7.16 and 7.17, respectively.

A PD-controller is constructed by defining a feedback loop, where $\mathbf{K}$ is the gain matrix:

$$\Delta\mathbf{u} = -\mathbf{K}\Delta\mathbf{x} \tag{7.19}$$

Control effort $\Delta\mathbf{u}$ is added to the optimal control $\mathbf{u}_0$ from the guidance algorithm, creating the combined control vector $\mathbf{u}$, which is forwarded to the attitude controller:

$$\mathbf{u} = \mathbf{u}_0 + \Delta\mathbf{u} \tag{7.20}$$

To optimise gain matrix $\mathbf{K}$, Gopal (1993) defines quadratic cost function J:

$$J = \int_0^\infty \left(\Delta\mathbf{x}^T\mathbf{Q}\Delta\mathbf{x} + \Delta\mathbf{u}^T\mathbf{R}\Delta\mathbf{u}\right) dt \tag{7.21}$$

For this cost function, $\mathbf{Q}$ and $\mathbf{R}$ are weight matrices that give an individual weight to all elements of state deviation $\Delta\mathbf{x}$ and control effort $\Delta\mathbf{u}$. The cost function seeks to balance the state errors and control effort. Bryson and Ho (1975) propose to use the maximum allowable state deviation for defining $\mathbf{Q}$ and the maximum allowable control effort for setting $\mathbf{R}$:

$$\mathbf{Q} = \mathrm{diag}\left\{\frac{1}{\Delta \mathrm{x}_{1_{\max}}^2} \quad \frac{1}{\Delta \mathrm{x}_{2_{\max}}^2} \quad \cdots\right\} \tag{7.22}$$

$$\mathbf{R} = \mathrm{diag}\left\{\frac{1}{\Delta \mathrm{u}_{1_{\max}}^2} \quad \frac{1}{\Delta \mathrm{u}_{2_{\max}}^2} \quad \cdots\right\} \tag{7.23}$$

For setting $\mathbf{Q}$, the mission requirements of Section 2.5 are used, yielding:

$$\mathbf{Q} = \mathrm{diag}\left\{\frac{1}{1.5^2} \quad \frac{1}{1.5^2} \quad \frac{1}{1.0^2} \quad \frac{1}{1.0^2} \quad \frac{1}{1.0^2} \quad \frac{1}{1.0^2}\right\} \tag{7.24}$$

As detailed in Section 2.4, the thrusters can deliver a maximum of 3,820 N of thrust. 95% of this thrust is allocated to the guidance algorithm. Because convex guidance returns bang-bang control, all of the allowed 95% of thrust is used for most of the time. In fact, the guidance algorithm only uses low-thrust for 52 seconds of the flight, which is just over 7% of the total flying time. This assumes the trajectory given in Section 2.4.

As a result, only 5% (191 N) of this thrust is available to the tracking algorithm. Assuming the spacecraft has a mass of 1,800 kg at the Powered Descent Initiation (PDI), the maximum achievable corrective acceleration is 0.106 m/s^2. This maximum acceleration figure is a magnitude. Therefore, to evenly divide the total value over the three spatial axes, the limit is multiplied with $\sqrt{\frac{1}{3}}$. This yields the following $\mathbf{R}$ matrix:

$$\mathbf{R} = \text{diag}\left\{ \frac{1}{\left(0.106\sqrt{\frac{1}{3}}\right)^2} \quad \frac{1}{\left(0.106\sqrt{\frac{1}{3}}\right)^2} \quad \frac{1}{\left(0.106\sqrt{\frac{1}{3}}\right)^2} \right\} \tag{7.25}$$

Substituting the definition of the control vector (Equation 7.19) into the cost-function (Equation 7.21), yields:

$$J = \int_0^\infty \Delta\mathbf{x}^T \left(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}\right) \Delta\mathbf{x} dt \tag{7.26}$$

Gopal (1993) equates the cost-function to the negative of the Lyapunov function in Equation 7.27:

$$\mathbf{V}(\Delta\mathbf{x}) = \Delta\mathbf{x}^T\mathbf{P}\Delta\mathbf{x} \tag{7.27}$$

such that:

$$\Delta\mathbf{x}^T \left(\mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K}\right) \Delta\mathbf{x} = -\frac{d}{dt}\Delta\mathbf{x}^T\mathbf{P}\Delta\mathbf{x} \tag{7.28}$$

Using the definition of the perturbation equation in Equation 7.18, Gopal (1993) is able to rewrite Equation 7.28 to the following matrix Riccati equation:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{7.29}$$

After solving Equation 7.29 for $\mathbf{P}$ (using a build-in function in MATLAB), the optimal gain matrix $\mathbf{K}$ is obtained through Equation 7.30:

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{7.30}$$

Optimisation of the cost function through the Riccati equation yields the following gain matrix:

$$\mathbf{K} = \begin{bmatrix} 0.0408 & 0 & 0 & 0.2923 & 0 & 0 \\ 0 & 0.0408 & 0 & 0 & 0.2923 & 0 \\ 0 & 0 & 0.0613 & 0 & 0 & 0.3554 \end{bmatrix} \tag{7.31}$$

Using these gains in the state-space system of Equation 7.15, returns the following eigenvalues for $\mathbf{A} - \mathbf{BK}$:

$$\lambda_{1,2} = -0.146 \pm 0.140i$$

$$\lambda_{3,4} = -0.146 \pm 0.140i$$

$$\lambda_{5,6} = -0.178 \pm 0.172i$$

Because the real parts of all eigenvalues are negative, the system is stable. Equation 7.17 shows that $\mathbf{B}$ does not change over time. On the other hand, matrix $\mathbf{A}$ of Equation 7.16 does change with time. However, analysis shows that this change is so small that the eigenvalues practically remain constant throughout the flight.

7.2.2 Results

To determine the base performance of the trajectory tracker, ideal conditions and ideal navigation are assumed. The only error sources are the discretisation of the optimal trajectory by the guidance algorithm and the attitude dynamics.

Running the trajectory tracker at 10 Hz and the attitude controller at 30 Hz, the control loop is able bring the spacecraft to within 0.12 metres of the target at the TG, with an offset in velocity of about 3 mm/s. The corrections of the trajectory tracker introduced an additional

0.51 m/s ΔV and an additional 0.16 kg of propellant, bringing the total propellant used to 808.4 kg. It is worth noting that the predominant component of the landing error is in the downrange direction. Because the attitude controller does not perfectly track the attitude, the spacecraft slightly lags behind its commanded manoeuvre to up right itself over the course of the landing. Therefore, the downrange component of the thrust is slightly larger than intended. This makes the spacecraft slow down too much, making it land short of the target. The trajectory tracker partially corrects for this, but because the position margin was set to 1.5 metres in Section 7.2.1, the trajectory tracker only reduces this error to 0.12 metres. This is no problem, because it meets the requirements set in Section 2.5.

As Section 7.1.2 discussed, an interpolated version of the optimal attitude is used to reduce the required control effort. However, the trajectory tracker is trying to follow a path that is an integration of a zero-order hold attitude, while the attitude controller tries to fly a piecewise linear attitude. This causes a mismatch in the trajectory, as shown in Figure 7.4 (note that random numbers were used for Figure 7.4 for an illustrative purpose). The upper graph in this figure shows the discontinuous attitude that is returned by the convex guidance algorithm, and the interpolated version that is flown by the attitude controller. The lower graph shows the integration - the velocity - that results from the accelerations. It is clear that the two trajectories do not fully line up. The trajectory tracker detects this deviation from the optimal trajectory and performs corrective manoeuvres, effectively trying to re-introduce the discontinuous attitude. However, because the trajectory tracker allows for a margin around the optimal path, the control effort is still 70% lower, compared to directly using the discontinuous optimal control from the guidance algorithm. Directly using a first-order function for the optimal control in convex guidance is expected to provide better performance, because in that case the commanded attitude for the attitude controller would directly corresponds with the optimal trajectory.

Remember that reducing the step size of convex guidance is not an option in this case because of the dimension of the problem, as shown in Section 6. Therefore, it would be interesting to study the performance of convex guidance when a first-order function is used for the optimal control, instead of a zero-order hold function. Due to time constraints, the higher order function for the control vector is not implemented into the convex guidance algorithm. Instead, the interpolated attitude is continued to be used for the remainder of this study.

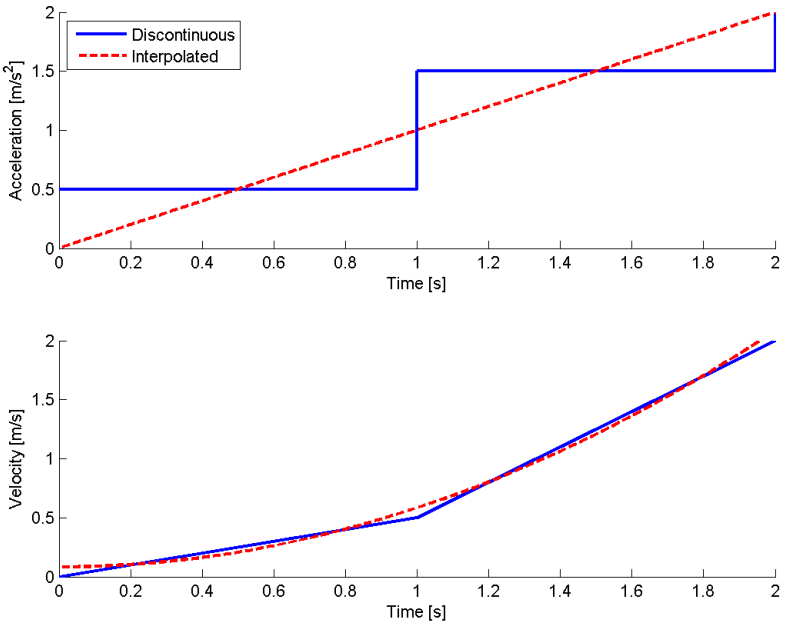


Figure 7.4: Difference in acceleration and velocity when using the discontinuous command directly from Convex Guidance versus the interpolated command.

As convex guidance is described in Chapter 4, the landing trajectory is re-optimised at the AP. The final position and velocity of the MBP are carried over as the initial conditions for the AP trajectory. However, the spacecraft's attitude and attitude dynamics are not taken into account. As a result, there is no guarantee that the attitude has a fluent transition from the MBP to the AP. In fact, when using the MBP and AP trajectories as discussed in Section 2.4, there is a discontinuity of about 20° in the attitude at the AG. This mismatch in attitude exists, because the MBP and AP were optimised separately by Gerth (2014). First, the position and velocity at the AG were designed to deliver the spacecraft to the TG with minimal fuel. Subsequently, the PDI was designed to reach the AG efficiently. In this process, the transition in attitude was not considered. Furthermore, Gerth (2014) constrained the attitude during the AP to ensure that the landing site is within the field of view of the navigational sensors, which might contribute to the attitude discontinuity.

Theoretically, an attitude constraint could be imposed on either the final-boundary condition of the MBP or the initial-boundary condition of the AP, but such constraints will have little effect. Because no attitude dynamics are modelled and the angular rate is not limited by convex guidance, the optimal control is free to instantly change the attitude of the spacecraft at any time. Therefore, if the attitude is constraint from jumping 20° at t_0, it will make the jump at $t = 1$ instead, only postponing the problem. One could argue that the MBP and the AP should be designed to have a better transition. However, this does not take away the possibility of a re-direct to a different landing site for hazard avoidance, which might also involve a large

attitude manoeuvres.

Figure 7.5 shows the discontinuity of 20° at the AG. The bottom graph shows that there is a big jump in z-axis thrust. Note that the thrust level remains constant, but the spacecraft rotates to up right itself, generating more thrust in the z direction. The *Nominal* -line shows the optimal thrust as given by convex guidance, the *Commanded* -line shows the corrective action from the trajectory tracker and the *True*-line shows the actual thrust that is achieved, due to the current attitude. The figure shows that the thrust deliberately overshoots the nominal command, to compensate for its slow attitude response. The upper graph of Figure 7.5 shows how this affects the position of the spacecraft. It shows that, because the spacecraft does not have the correct attitude, the spacecraft quickly drifts off the optimal trajectory to over 5 metres. However, once the large attitude error of 20° has been compensated, the trajectory tracker is able to return the spacecraft to the optimal trajectory.

Even though the controllers are able to return the spacecraft to the optimal trajectory, this excursion cannot be the optimal solution. Therefore, it is advisable to expand convex guidance to incorporate an angular rate limiter to prevent large discontinuities and to allow for the use of attitude constraints on the boundary conditions of the trajectory. Large discontinuities need to be prevented, because they can destabilise the control system, when not tuned for.

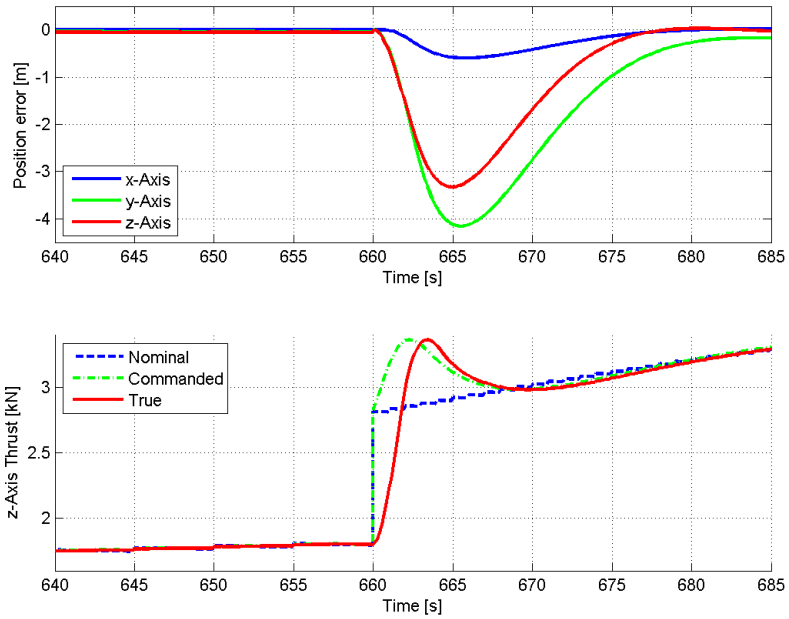


Figure 7.5: A discontinuity between the MBP and AP causes the spacecraft to drift from the optimal trajectory, but the trajectory tracker eventually corrects this deviation.

7.3 Sensitivity

After determining the performance of the control loop in Section 7.2.2, the stability of the system is tested in this section. Ideal navigation is still assumed, but other perturbations are introduced to the system.

Table 7.1 gives an overview of the impact of different perturbations on the performance of the control loop. The unperturbed performance of Section 7.2.2 is used as the baseline for reference. All cases are discussed in greater detail in the following paragraphs.

Table 7.1 shows that the control loop is robust to guidance related perturbations, and that the spacecraft is able to land at the targeted landing site while meeting the requirements of Section 2.5.

Despite the disadvantage of the discontinuous attitude during the phase transition and the zero-order hold function for the attitude in general (discussed in Section 7.2.2), the results show that a simple trajectory tracker is able to correct for most of the simplification errors made by convex guidance. For example, Section 6 showed that by neglecting higher-order gravity terms, convex guidance can make errors in the order of hundreds of metres, while

Table 7.1 shows that the trajectory tracker easily removes these errors with little effort. This practically shows that the spherical gravity model used by convex guidance is sophisticated enough for the purpose of Lunar landing. Table 7.1 further shows that Lunar rotation and third body perturbations have practically no effect on the overall landing performance, apart from the slight increase in propellant for Lunar rotation. Also for thrust misalignment, there is hardly any deterioration in landing performance. However, compensation of the thrust misalignment requires 500% control effort, with respect to the reference case.

Table 7.1: Sensitivity of the control loop to guidance related perturbations.

	Landing error		**Propellant**	**Comment**
	m	*mm/s*	*kg*	
Reference	0.12	3	808.4	
Lunar Rotation	0.12	3	+0.17	
3rd Bodies	0.12	3	+0.00	
Irregular Gravity	0.09 to 0.20	3 to 4	-0.12 to +0.33	*depending on landing site*
Mass error	0.69	14	+4.65	*10 kg error*
RCS thrust	0.12	3	+0.01	*10% error*
Thrust misalignment	0.13	4	+0.00	*0.4° misalignment*
				500% control effort

Lunar Rotation To determine the effects of Lunar rotation, a landing on the Moon's equator is simulated. The spacecraft lands 0.12 metres of target with a velocity error of 3 mm/s, exactly as the unperturbed flight in Section 7.2.2. The difference is that the spacecraft used an additional 0.17 kg. Overall this analysis shows that the control loop can easily deal with the Lunar rotation.

3rd Body Perturbations To create a worst case scenario, the spacecraft is landing at 0° longitude and 0° latitude. Section 6 showed that this yields the largest deviation from the optimal trajectory, because this creates the closest approach for the Earth. All epochs from Section 6 are simulated to include a variety of Solar-System compositions.

The simulations showed that, regardless of the epoch, the spacecraft lands 0.12 metres off target, whit a velocity error of 3 mm/s. These analyses show that third body perturbation have no significant effect on the performance of the system.

Irregular Gravity Field The influence of irregularities in the Lunar gravity field on the space-craft's landing trajectory depends on the targeted landing site. Therefore, simulations are run

for the landing sites defined in Section 6. These landing sites were picked for the strongly varying gravity field in that region.

Figure 7.6 shows the landing performance of the control loop at the seven landing sites of Section 6. The reference case is the landing performance of the control loop without gravitational perturbations.

The figure shows that the spacecraft consistently lands short of the target. As discussed in Section 7.2.2, this is caused by the spacecraft not perfectly tracking the pitch command. As the spacecraft moves from a large pitch angle to near zero pitch, the spacecraft lags behind in moving itself upright. Therefore, the thrust has a too large component in the downrange direction, causing the spacecraft to land short of its target.

Figure 7.6 further shows that for the selected landing sites, the landing only deviates from the reference case in the order of centimetres. Therefore, this figure shows that the system copes well with the irregular gravity field of the Moon. The propellant used, depends on the landing site. Where the gravitational pull is larger than nominal, more propellant is required and vice versa. In the best case for the seven defined landing sites, the spacecraft saved 0.12 kg of propellant, while spending an extra 0.33 kg in the worst case. Furthermore, the attitude controller used an additional 15% to 35% of control effort during these landings.

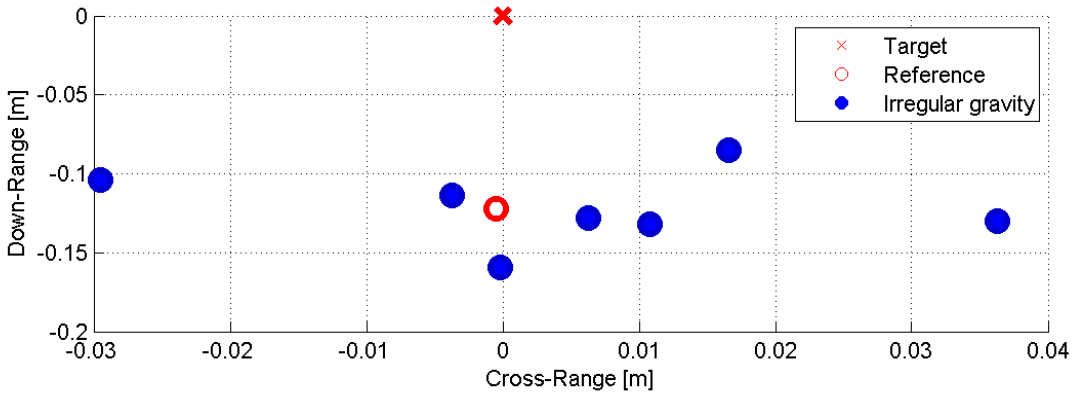


Figure 7.6: Landing positions of several simulations. **Reference**: Performance of the control loop without any perturbations. **Irregular gravity**: Performance of the control loop with an irregular gravity field as perturbation, for the 7 landing sites defined in Section 6. Note that the x- and y-axes are not on the same scale.

Mass Error When there is an error in the estimated mass of the spacecraft, there will be a mismatch between the acceleration the system is trying to produce and the accelerations that are actually generated by the thrusters. Such errors can destabilise the control system.

Simulations have shown that when the spacecraft is 10 kg overweight, the control system is able to land within 0.69 metres from the target with a velocity error of 14 mm/s. This landing error is significantly larger than the landing error the spacecraft achieves with an ideal mass. However, the landing error remains well within the requirements set in Section 2.5. An additional 4.65 kg of propellant and 1% of control effort were used to safely land the 10 kg of overweight.

RCS Thrust Error Errors in the magnitude of the attitude control torque causes a mismatch in the angular acceleration that is commanded to be produced and the angular acceleration that is actually achieved. With a $\pm 10\%$ error in applied torque, the spacecraft is still able to land 0.12 metres from the target with an 3 mm/s velocity error, while only requiring an additional 0.01 kg of propellant. However, there is an increase of about 4% in control effort. Overall the system seems very robust to such errors.

Thrust Misalignment Having the thrust misaligned with respect to the centre of mass introduces undesired torques when the main engine is ignited. With a thrust misalignment of 0.4°, a torque of 22 Nm is created when the spacecraft is at full throttle. This torque is almost 17% of the torque the spacecraft can generate with its RCS. With this misalignment the spacecraft can still land within 0.13 metres of the target with a velocity error of 4 mm/s. However, because there is a constant disturbing torque, the control effort is five times that of the reference case with ideal thrust alignment.

8 Navigation

The role of navigation is to estimate the state of the spacecraft, that is, the position, velocity and attitude. This kind of information is required by the guidance system to determine the optimal trajectory and by the control system to orient the spacecraft to assure that the optimal trajectory is followed. A multitude of sensors exist to measure one or more parameters of the spacecraft's state vector. Mathematically, the results of separate sensors can be combined to acquire accurate estimations of the spacecraft's full state vector.

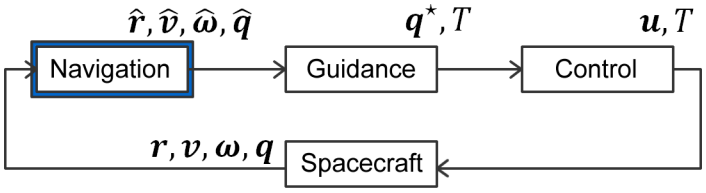


Figure 8.1: Closed guidance, navigation and control loop.

8.1 Sensors

Inertial Navigation Inertial navigation directly measures the linear acceleration and angular rate of the spacecraft. These measurements are integrated from a specified initial condition to track the position, velocity and attitude of the spacecraft. The down side of this method is that the integration sums all the errors in the raw measurements, which makes the propagation errors in position and attitude grow over time. For example, Amzajerdian et al. (2011) mention that drift errors of an inertial measurement unit can be in the order of kilometres for a Moon landing. Inertial navigation is practically always used, but at least some other kind of measurement is needed to prevent the errors from growing too large.

Groves (2013) provides a good and detailed description of inertial measurement units and their error sources. Table 8.1 shows typical accelerometer and gyroscope biases and accuracies, as provided by Groves (2013).

Table 8.1: Inertial Measurement Unit errors and accuracies based on Groves (2013). All errors and accuracies need to be applied to each axis individually.

Accelerometer	
Bias	$\mu = 0.01 \text{ mg} = 10^{-4} \text{ m/s}$
Accuracy	$\sigma = 20 \ \mu\text{g}/\sqrt{\text{Hz}} = 0.0020 \text{ m/s}^2$ (at 100 Hz)
Scaling errors	$s = 10^{-3}$
Misalignment	$m = 10^{-3}$
Gyroscopes	
Bias	$\mu = 0.001°/\text{hr} = 5 \cdot 10^{-9} \text{ rad/s}$
Accuracy	$\sigma = 0.001°/\sqrt{\text{hr}} = 2.91 \cdot 10^{-6} \text{rad/s}$ (at 100 Hz)
Scaling errors	$s = 10^{-4}$
Misalignment	$m = 10^{-3}$
Sampling rate	100 Hz

Star Tracker A star tracker uses a camera to look at the stars and compares the acquired image to an onboard star database. From this information the system can deduce the orientation of the spacecraft about all its three axes.

In early 2006, NASA's New Horizons spacecraft was launched to study Pluto, Pluto's moons and the Kuiper belt. To perform its mission, New Horizons needs to be able to autonomously determine and adjust its attitude. Therefore, the spacecraft was equipped with star trackers which were rated to have an accuracy of 7, 7 and 70 arcseconds (3σ) around their x-, y- and z-axes at 10 Hertz; where the z-axis is aligned with the star tracker's bore sight. Rogers et al. (2006) have analysed the performance of New Horizons' star tracker through received telemetry. The data represent one period of a month and another one of a month and a half. They found accuracies of 3.42, 3.32 and 32.31 arcseconds around the x-, y- and z-axes for the first period and accuracies of 4.12, 3.96 and 34.21 arcseconds around the x-, y- and z-axes for the second period. From these values, the accuracies of Table 8.2 are assumed for modelling star trackers.

Table 8.2: Assumed Star Tracker performance parameters.

Accuracy x-axis	$3\sigma = 4 \text{ acrseconds} = 1.94 \cdot 10^{-5} \text{ rad}$
Accuracy y-axis	$3\sigma = 4 \text{ acrseconds} = 1.94 \cdot 10^{-5} \text{ rad}$
Accuracy z-axis	$3\sigma = 34 \text{ acrseconds} = 1.65 \cdot 10^{-4} \text{ rad}$
Sampling rate	10 Hz

Visual Navigation With visual navigation, images are created of the Lunar surface. By comparing the images to an on-board database, landmarks can be identified and the position of the spacecraft can be estimated. Imaging the Lunar surface can either be done passively, where a camera takes photos of the surface as it is lit by the Sun, or actively, where the return-time of a pulse of laser is measured (Lidar). Passive systems are usually lighter in mass, but active systems can be used under any conditions, as they do not rely on the surface to be illuminated by the Sun.

Because automatic visual navigation systems are relatively new and still under development, it is difficult to obtain concise performance figures. Therefore, other literature is used to create a hypothetical situation. For absolute navigation, a map needs to be available for the region of the landing site. Johnson and Montgomery (2008) state that a global Lunar surface map is available, with a resolution of 100 metres per pixel. Johnson and Montgomery (2008) continue to state that the position estimation of the spacecraft can never be more accurate than the resolution of the terrain map. Therefore, for the Main Braking Phase (MBP), the noise on the position measurements is assumed to be 100 metres (1σ).

For the last stage of the landing, the Approach Phase (AP), the landing site is scanned for potential hazards. This procedure creates a detailed map of the landing site on board of the spacecraft. Based on this map a final landing location is determined, depending on what spot is deemed safest. Amzajerdian et al. (2011) discuss the performance of mapping and relative navigation requirements for NASA's ALHAT project. Amzajerdian et al. (2011) state that at Approach Gate (AG), the landing site can be mapped with a resolution of 40 centimetres. Therefore, it is assumed that relative navigation during the AP has a noise level of 40 centimetres (1σ).

Note that these figures are very rudimentary assumptions. However, the aim of this study is to determine the interaction between convex guidance and a navigation algorithm, not the absolute performance of the overall system.

8.2 State Estimators

The separate measurements performed by the individual sensors need to be fused to a coherent state estimation. This section gives an overview of several state estimators, followed by a selection of the most suitable estimator for this study.

The Kalman Filter is a very popular and well known state estimator for linear systems. The Extended Kalman Filter (EKF) is a variation of the Kalman Filter, where nonlinear systems are approximated by a first-order Taylor-series expansion. Old state estimates are propagated

over time and updated with new observations on regular intervals.

The Sigma-Point Kalman Filter (SPKF) is closely related to the EKF, but models nonlinear systems through mapping the statistical parameters using the Unscented Transform, instead of a Taylor-series expansion. A specific subgroup of SPKF is the Unscented Kalman Filter (UKF). With respect to a Taylor series expansion, the UKF reaches second-order or higher accuracies (Gross et al., 2010).

The Particle Filter (PF) is somewhat related to the SPKF, because the PF also uses points (particles) to determine the mapping of statistical parameters. However, where the sigma-points of the SPKF are set deterministically, the particles of the PF are set randomly and assigned weights (Gross et al., 2010).

The State-Dependent Riccati Estimator (SDRE) factorizes the dynamic equations into a linear system dependant on the state. This is different from the EKF, because the EKF uses model linearization to solve the Riccati equation, while an SDRE solves the Ricatti equation at each time step using system factorization (Berman et al., 2014). A significant downside, is that an infinite number of factorizations is available, and it is not known how to select the right factorization to minimize the state-estimation error (Berman et al., 2014).

In an EKF and UKF trade-off, both St-Pierre and Gingras (2004) and VanDyke et al. (2004) find that the UKF provides slightly better estimations than the EKF. The main difference however, is the convergence of the two filters. Crassidis and Markley (2003) and VanDyke et al. (2004) show a significant improvement in the convergence rate of the UKF over the EKF for large initialization errors of the estimated state. However, Rhudy et al. (2013) show that specific cases can be found for which the EKF outperforms the UKF on convergence. Furthermore, St-Pierre and Gingras (2004) add that the UKF requires significantly more computational power than the EKF due to the number of evaluations needed for the UKF.

Gross et al. (2010) compared the EKF, UKF and PF. Their study largely agrees with St-Pierre and Gingras (2004) and VanDyke et al. (2004), in that the EKF and UKF have similar performance. Gross et al. (2010) add to this that the PF performed less well in the estimation of the state than the EKF and UKF. Gross et al. (2010) further found that, in terms of computational load, the EKF out performs both the UKF (which is in agreement with St-Pierre and Gingras (2004)) and the PF.

Berman et al. (2014) performed a comparison between the EKF and the SDRE. Their study showed that the EKF performed similar, if not better than the SDRE, in terms of accuracy. The SDRE however, can provide faster convergence.

The aforementioned studies show that the EKF, UKF and SDRE all obtain similar accuracies

and outperform the PF. The UKF and SDRE can converge faster than the EKF, but in turn, the EKF is the easiest in its implementation and requires the least computational power.

Because the navigation system is only a small part of this study, the EKF is selected for the spacecraft's state estimation. It is an easy and computationally light algorithm that performs well enough. It is considered a good algorithm to set a baseline for the interaction between convex guidance and navigation algorithms. Its lesser performance in convergence rate is not considered a problem, because the measurements come in at a relatively high rate. Furthermore, tests have shown that the EKF is still able to converge within a small fraction of the total simulated flight time.

8.3 Extended Kalman Filter

For the EKF, $\hat{\mathbf{x}}$ is defined as the estimated state of the spacecraft:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{r}}^T & \hat{\mathbf{v}}^T & \hat{\mathbf{q}}^T \end{bmatrix}^T \in \mathbb{R}^{10} \tag{8.1}$$

The EKF propagates older state estimates over time, and updates these estimates with new measurements from the navigational sensors. Simon (2006) gives the following algorithm for the propagation of the estimated state from time k to $k+1$, where $\mathbf{f}$ is the state propagation function:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{f}(\hat{\mathbf{x}}_k) \tag{8.2}$$

$$\mathbf{F} = \frac{\partial \mathbf{f}(\hat{\mathbf{x}}_k)}{\hat{\mathbf{x}}_k} \tag{8.3}$$

$$\mathbf{P}_{k+1} = \mathbf{F}\mathbf{P}_k\mathbf{F}^T + \mathbf{Q} \tag{8.4}$$

For these equations, $\mathbf{P} \in \mathbb{R}^{10\times10}$ is the covariance matrix of the estimated state and $\mathbf{Q} \in \mathbb{R}^{10\times10}$ is the covariance matrix of the state propagation.

When a new state measurement ($\mathbf{y}$) is provided, Simon (2006) updates the estimated state and the covariance matrix from $\hat{\mathbf{x}}_m$ to $\hat{\mathbf{x}}_{m+1}$ and $\mathbf{P}_m$ to $\mathbf{P}_{m+1}$, respectively, as follows, where

$\mathbf{h}$ is the state observation function (by the sensors):

$$\mathbf{H} = \frac{\partial \mathbf{h}(\hat{\mathbf{x}}_m)}{\hat{\mathbf{x}}_m} \tag{8.5}$$

$$\mathbf{K} = \mathbf{P}\mathbf{H}^T \left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{8.6}$$

$$\hat{\mathbf{x}}_{m+1} = \hat{\mathbf{x}}_m + \mathbf{K}\left\{\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_m)\right\} \tag{8.7}$$

$$\mathbf{P}_{m+1} = \left(\mathbf{I}_{6\times 6} - \mathbf{K}\mathbf{H}\right)\mathbf{P}_m\left(\mathbf{I}_{6\times 6} - \mathbf{K}\mathbf{H}\right)^T + \mathbf{K}\mathbf{R}\mathbf{K}^T \tag{8.8}$$

Here, $\mathbf{R} \in \mathbb{R}^{9\times 9}$ is the covariance matrix of the measurements and $\mathbf{K} \in \mathbb{R}^{10\times 9}$ is the gain matrix. After both the state propagation and the measurement update, the attitude quaternion is normalised by force, by dividing it by its own magnitude.

This section just provided a quick overview of the EKF algorithm. The following sections elaborate on the state propagation and measurement update, respectively.

State Propagation

As Equation 8.2 showed, the estimated state is propagated by function $\mathbf{f}$:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) = \mathbf{x}_k + \dot{\mathbf{x}}_k h \tag{8.9}$$

Where the dynamic equations are given by:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{R}_{IB}\mathbf{M}_a^{-1}(\mathbf{a}_m - \mathbf{b}_a) - \frac{\mu}{\|\mathbf{r}\|^3}\mathbf{r} \\ \frac{1}{2}\boldsymbol{\Omega}\mathbf{M}_g^{-1}(\boldsymbol{\omega}_m - \mathbf{b}_g) \end{bmatrix} \tag{8.10}$$

With:

$$\mathbf{\Omega} = \begin{bmatrix} q_4\mathbf{I}_{3\times 3} + [\mathbf{q}\times] \\ -\mathbf{q}^T \end{bmatrix} = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{8.11}$$

These are the true dynamics of the spacecraft, where $\mathbf{R}_{IB}$ is the rotation matrix from the body frame to the inertial frame, $\mathbf{a}_m$ is the measured acceleration, $\mathbf{b}_a$ are the biases of the accelerometer, $\mathbf{M}_a$ are the scaling errors and misalignments of the accelerometer, $\boldsymbol{\omega}_m$ is the measured angular rate, $\mathbf{b}_g$ are the biases of the gyroscopes and $\mathbf{M}_g$ are the misalignments and scaling errors of the gyroscopes:

$$\mathbf{M}_a = \mathbf{I}_{3\times 3} + \begin{bmatrix} s_x & m_{xy} & m_{xz} \\ m_{yx} & s_y & m_{yz} \\ m_{zx} & m_{zy} & s_z \end{bmatrix} \tag{8.12}$$

$$\mathbf{M}_g = \mathbf{I}_{3\times 3} + \begin{bmatrix} s_p & m_{pq} & m_{pr} \\ m_{qp} & s_q & m_{qr} \\ m_{rp} & m_{rq} & s_r \end{bmatrix} \tag{8.13}$$

However, because the EKF does not estimated the biases, scaling errors and misalignments, Equation 8.10 simplifies to:

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} \dot{\hat{\mathbf{r}}} \\ \dot{\hat{\mathbf{v}}} \\ \dot{\hat{\mathbf{q}}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}} \\ \hat{\mathbf{R}}_{IB}\mathbf{a}_m - \frac{\mu}{\|\hat{\mathbf{r}}\|^3}\hat{\mathbf{r}} \\ \frac{1}{2}\mathbf{\Omega}\boldsymbol{\omega}_m \end{bmatrix} \tag{8.14}$$

Where $\hat{\mathbf{R}}_{IB}$ is constructed from the estimated attitude quaternion $\hat{\mathbf{q}}$. Substitution of this relation into Equation 8.9, yields:

$$\begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{q}} \end{bmatrix}_{k+1} = \mathbf{f}(\hat{\mathbf{x}}_k) = \begin{bmatrix} \hat{\mathbf{r}} \\ \hat{\mathbf{v}} \\ \hat{\mathbf{q}} \end{bmatrix}_k + \begin{bmatrix} \hat{\mathbf{v}} \\ \hat{\mathbf{R}}_{IB}\mathbf{a}_m - \frac{\mu}{\|\hat{\mathbf{r}}\|^3}\hat{\mathbf{r}} \\ \frac{1}{2}\mathbf{\Omega}\boldsymbol{\omega}_m \end{bmatrix}_k h \tag{8.15}$$

The partial derivative of propagation function $\mathbf{f}$ is required to propagate covariance matrix $\mathbf{P}$, as shown by Equation 8.4. The partial derivative of $\mathbf{f}(\mathbf{x})$ with respect to state $\mathbf{x}$ is obtained

from Equation 8.15 as follows:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{I}_{3\times3} & \frac{\partial \mathbf{r}}{\partial \mathbf{v}} & \mathbf{0}_{3\times4} \\ \frac{\partial \mathbf{v}}{\partial \mathbf{r}} & \mathbf{I}_{3\times3} & \frac{\partial \mathbf{v}}{\partial \bar{\mathbf{q}}} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \frac{\partial \bar{\mathbf{q}}}{\partial \bar{\mathbf{q}}} \end{bmatrix} \tag{8.16}$$

This matrix is filled with the following partial derivatives.

$$\frac{\partial \mathbf{r}}{\partial \mathbf{v}} = h\mathbf{I}_{3\times3} \tag{8.17}$$

$$\frac{\partial \mathbf{v}}{\partial \mathbf{r}} = \frac{\mu h}{\|\mathbf{r}\|^5} \begin{bmatrix} 2x^2 - y^2 - z^2 & 3xy & 3xz \\ 3xy & -x^2 + 2y^2 - z^2 & 3yz \\ 3xz & 3yz & -x^2 - y^2 + 2z^2 \end{bmatrix} \tag{8.18}$$

$$\frac{\partial \mathbf{v}}{\partial \bar{\mathbf{q}}} = 2h \left[\left\{ \mathbf{q}^T \mathbf{a}_m \mathbf{I}_{3\times3} + \mathbf{q}\mathbf{a}_m^T - \mathbf{a}_m \mathbf{q}^T + q_4 [\mathbf{a}_m \times] \right\} \quad \left\{ q_4 \mathbf{a}_m + \mathbf{a}_m \times \mathbf{q} \right\} \right] \tag{8.19}$$

The term in between the left curly brackets in Equation 8.19 is a 3 by 3 matrix, while the term in between the right curly brackets is a 3 by 1 matrix. Combination of these sub-matrices constructs the overall 3 by 4 matrix of $\frac{\partial \mathbf{v}}{\partial \bar{\mathbf{q}}}$. Furthermore, $[\mathbf{a}_B \times]$ is the cross-product matrix so that $[\mathbf{a}_m \times]\mathbf{q}$ would equal $\mathbf{a}_B \times \mathbf{q}$:

$$[\mathbf{a}_m \times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{8.20}$$

The partial derivative of the propagated attitude quaternion with respect to the initial attitude quaternion is given as follows:

$$\frac{\partial \bar{\mathbf{q}}}{\partial \bar{\mathbf{q}}} = \mathbf{I}_{4\times4} + \frac{h}{2} \begin{bmatrix} -[\boldsymbol{\omega} \times] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{8.21}$$

Where again, $[\boldsymbol{\omega} \times]$ is the cross-product matrix, so that $[\boldsymbol{\omega} \times]\mathbf{q}$ would equal $\boldsymbol{\omega} \times \mathbf{q}$.

Measurement Update

The observation of the spacecraft's state is given by the following relation, where $\mathbf{j}$ is the unit vector in the y-direction and $\mathbf{k}$ is the unit vector in the z-direction:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \mathbf{h}(\mathbf{x}) = \begin{bmatrix} \mathbf{r} \\ \mathbf{R}_{IB}\mathbf{j} \\ \mathbf{R}_{IB}\mathbf{k} \end{bmatrix} \tag{8.22}$$

In this equation, the first observation $\mathbf{y}_1$ is the position vector as measured by the visual navigation system. The second and third observations, $\mathbf{y}_2$ and $\mathbf{y}_3$, are star tracker direction measurements with respect to two stars. Because $\mathbf{j}$ and $\mathbf{k}$ remain constant, while $\mathbf{R}_{IB}$ changes with the spacecraft's attitude quaternion, this allows the attitude of the spacecraft to be determined.

For the measurement update in Equation 8.7, $\mathbf{y}$ are the true measurements as provided by the sensors, while $\mathbf{h}(\hat{\mathbf{x}}_m)$ are the expected measurements based on the estimated state. Equation 8.5 requires the partial derivative of the observation equation with respect to the state, which is obtained as follows:

$$\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{r}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \frac{\partial \mathbf{y}_2}{\partial \bar{\mathbf{q}}} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \frac{\partial \mathbf{y}_3}{\partial \bar{\mathbf{q}}} \end{bmatrix} \tag{8.23}$$

Where the sub partial derivatives are given by:

$$\frac{\partial \mathbf{y}_1}{\partial \mathbf{r}} = \mathbf{I}_{3\times3} \tag{8.24}$$

$$\frac{\partial \mathbf{y}_2}{\partial \bar{\mathbf{q}}} = 2\left[\left\{\mathbf{q}^T\mathbf{j}\mathbf{I}_{3\times3} + \mathbf{q}\mathbf{j}^T - \mathbf{j}\mathbf{q}^T + q_4[\mathbf{j}\times]\right\} \quad \{q_4\mathbf{j} + \mathbf{j}\times\mathbf{q}\}\right] \tag{8.25}$$

$$\frac{\partial \mathbf{y}_3}{\partial \bar{\mathbf{q}}} = 2\left[\left\{\mathbf{q}^T\mathbf{k}\mathbf{I}_{3\times3} + \mathbf{q}\mathbf{k}^T - \mathbf{k}\mathbf{q}^T + q_4[\mathbf{k}\times]\right\} \quad \{q_4\mathbf{k} + \mathbf{k}\times\mathbf{q}\}\right] \tag{8.26}$$

8.4 Results

To determine the landing performance of the spacecraft with navigational errors, a batch of simulations are run where no perturbations are introduced other than the noise of the navigational sensors, which the EKF attempts to filter. Table 8.3 shows the 1σ values of the normally distributed noise of the sensors.

Table 8.3: Noise levels on navigational sensors.

	1σ	Comment
Accelerometer	$0.001\ \mathrm{m/s^2}$	
Position	$100\ \mathrm{m}$	*main braking phase*
Position	$0.4\ \mathrm{m}$	*approach phase*
Gyroscope	$2 \cdot 10^{-6}\ \mathrm{rad/s}$	
Attitude	0.001°	*all axes*

The standard trajectory is flown as provided by Section 2.4. The EKF propagation algorithm, the accelerometers and the gyroscopes run at 50 Hz. The EKF estimation update algorithm and the position and attitude measurements run at 10 Hz. As a reminder, the guidance algorithm optimised the trajectory in steps of five seconds for the MBP and steps of one second for the AP. The trajectory tracker runs at a frequency of 10 Hz, while the attitude controller operates at 30 Hz.

Furthermore, a low pass filter, running at 10 Hz with a break frequency of 2 rad/s, is used to smooth the noisy estimated state from the EKF, as input for the trajectory tracker. If the noisy signal is not filtered, the noise is passed on from the navigational algorithm to the trajectory tracker and eventually to the attitude controller.

At the start of the flight (at PDI), the covariance matrix $\mathbf{P}$ is initialised based on the variance of the sensor noise, as given by Equation 8.27. Similarly, the $\mathbf{Q}$ and $\mathbf{R}$ matrices are tuned as shown in Equations 8.28 and 8.29:

$$\mathbf{P} = \mathrm{diag}\left\{10^4 \quad 10^4 \quad 10^4 \quad 1 \quad 1 \quad 1 \quad 4 \cdot 10^{-11} \quad 4 \cdot 10^{-11} \quad 4 \cdot 10^{-11} \quad 4 \cdot 10^{-11}\right\} \tag{8.27}$$

$$\mathbf{Q} = \mathrm{diag}\left\{10^{-1} \quad 10^{-1} \quad 10^{-1} \quad 10^{-5} \quad 10^{-5} \quad 10^{-5} \quad 4 \cdot 10^{-12} \quad 4 \cdot 10^{-12} \quad 4 \cdot 10^{-12} \quad 4 \cdot 10^{-12}\right\} \tag{8.28}$$

$$\mathbf{R} = \text{diag}\left\{10^4 \quad 10^4 \quad 10^4 \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11}\right\} \tag{8.29}$$

When the spacecraft transitions from the MBP to the AP, the $\mathbf{Q}$ and $\mathbf{R}$ matrices change according to the new sensor noise variances:

$$\mathbf{Q} = \text{diag}\left\{10^{-5} \quad 10^{-5} \quad 10^{-5} \quad 10^{-6} \quad 10^{-6} \quad 10^{-6} \quad 4\cdot10^{-12} \quad 4\cdot10^{-12} \quad 4\cdot10^{-12} \quad 4\cdot10^{-12}\right\} \tag{8.30}$$

$$\mathbf{R} = \text{diag}\left\{10^{-1} \quad 10^{-1} \quad 10^{-1} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11} \quad 4\cdot10^{-11}\right\} \tag{8.31}$$

Because the navigation algorithm is not able to estimate the spacecraft's position to within a metre certainty during the MBP, the trajectory tracker is relaxed. Because the noise level of the position sensor is 100 metres, the trajectory tracking position margin is also set to 100 metres. The velocity margin is increased from 1 m/s to 10 m/s. As later results show, 100 metre uncertainty at the AG hardly influences the propellant use. Because the navigation algorithm can determine the state of the spacecraft much more accurate during the AP, the trajectory tracker is set more strict for the AP. For the AP, the trajectory tracker is set to its original settings, as defined in Section 7.2.1.

The attitude controller is directly used with the gains from Section 7.1.2.

Figure 8.2 shows that the EKF converges rapidly for both the MBP and the AP. For the MBP the algorithm converges to the order of tens of metres, where the measurement errors are in the order of hundreds of metres. For the AP, the 3σ value decreased from 1.2 metres for the position measurement, to 0.3 metres for the position estimation. Note that the 3σ values from the EKF $\mathbf{P}$-matrix in Figure 8.2 have a considerable margin to the actual estimation errors. This margin is used to assure convergence when perturbations are introduced in the sensitivity analysis.

The attitude observation noise is 0.003° (3σ) on all three axes of rotation. This is difficult to compare with the EKF performance, because the attitude is estimated in the form of a quaternion. However, after converging, the Euler angle between the estimate attitude quaternion and the true attitude quaternion is 0.0005° on average.

Figure 8.3 shows the targeting performance of 1,000 simulated flights, with the navigational

performance mentioned above. These landings all meet the requirements of Section 2.5. As a result of the lower navigational performance during the MBP, the figure clearly shows a much larger standard deviation from the target at the AG than for the TG. However, because the trajectory is re-optimised at the AG and more accurate navigation is available during the AP, the spacecraft is able to reach the TG within centimetres, even though it can be off by 10 metres at the AG.

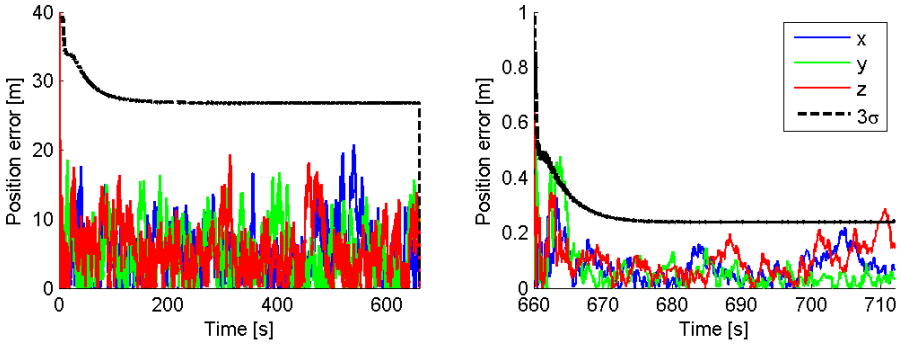


Figure 8.2: State estimation errors, where the black line is the 3σ-value from the $\mathbf{P}$-matrix. **Left**: Position errors during the MBP. **Right**: Position errors during the AP.

Table 8.4 shows the performance at TG in more detail. Note that the average error at TG corresponds with the error of the control system when no navigational errors are introduced, as discussed in Section 7.2.2. This shows that the navigation uncertainty has practically been superimposed on the performance of the control algorithms.

Table 8.4: Average errors at TG.

	Average	3σ
x	0.001 m	± 0.187 m
y	-0.099 m	± 0.196 m
z	-0.012 m	± 0.182 m
u	0 mm/s	± 41 mm/s
v	2 mm/s	± 41 mm/s
w	1 mm/s	± 41 mm/s
Propellant	808.45 kg	± 0.05 kg

Overall, these results show that the predominant factor in the landing performance is the sensor accuracy. Therefore, when less accurate sensors are available, the landing precision deteriorates accordingly, as can be seen for the approach gate in Figure 8.3. Note however,

that this study investigates the influence of convex guidance on the navigational performance, not the absolute performance of the system. Nonetheless, it can be concluded that convex guidance can facilitate pin-point landing, when high-accuracy navigational sensors are available.

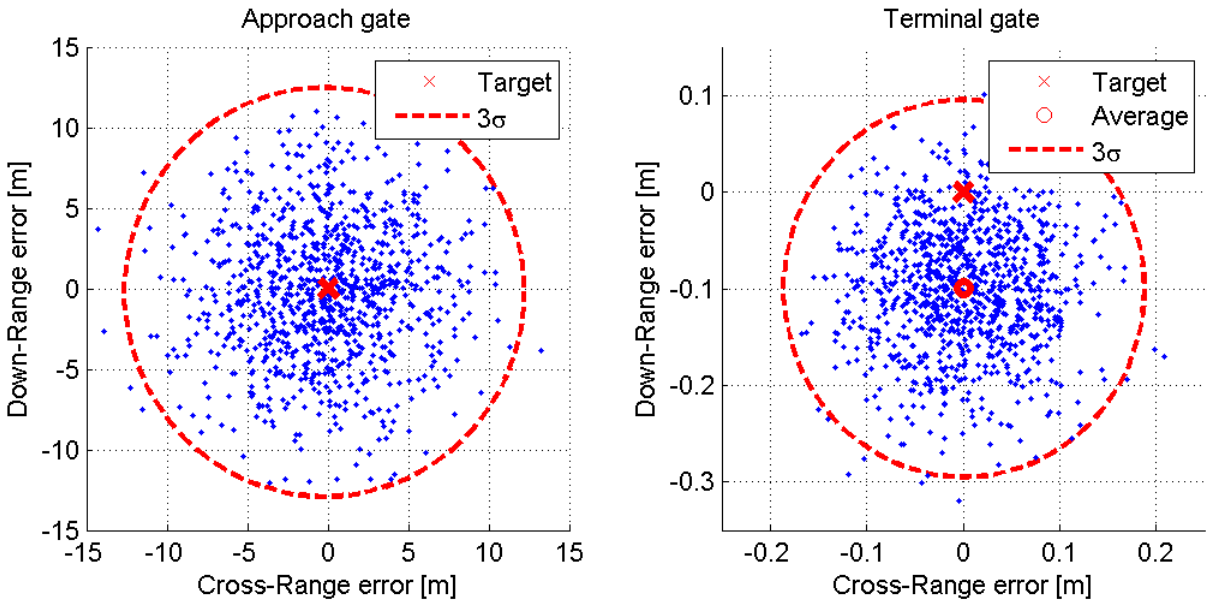


Figure 8.3: Landing performance for 1,000 flights with navigation errors.

8.5 Sensitivity

To determine the robustness of the GNC-loop, the impact of a set of perturbations on the landing performance is investigated. For the simulations, the baseline GNC-loop from Section 8.4 is used, on top of which, the perturbations are introduced. To determine the impact of these perturbations, 400 simulations are run for each type of perturbation. Testing with a different number of flights, at 400 runs the statistical parameters proved to converge.

Table 8.5 gives an overview of the average and 3σ standard deviation of the position and velocity errors at TG, as a consequence of different perturbations. Furthermore, the table also shows the expended propellant mass. A more detailed description of the conditions and results are discussed in the following paragraphs.

Table 8.5 shows that sensor misalignment and errors, thrust misalignment, mass errors and irregularities in the gravity field introduce the largest errors in position at TG. For the propellant mass, errors in the initial velocity and the mass of the spacecraft have the largest impact. Overall, the table shows that the GNC-loop is robust to these perturbations, as landing performances meet the requirements of Section 2.5.

Table 8.5: Average and 3σ standard deviation of final-boundary errors at TG and propellant mass.

	Position	**Velocity**	**Propellant**
	m	*mm/s*	*kg*
Reference	0.137 ± 0.166	22 ± 29	808.446 ± 0.053
Sensor bias	0.139 ± 0.170	22 ± 31	808.448 ± 0.054
Misalignment and scaling errors	0.289 ± 0.316	32 ± 46	808.446 ± 0.170
Initial position	0.134 ± 0.154	22 ± 28	808.456 ± 0.064
Initial velocity	0.135 ± 0.164	23 ± 28	808.938 ± 2.517
Lunar rotation	0.134 ± 0.169	22 ± 30	806.813 ± 0.053
3rd body perturbations	0.135 ± 0.153	21 ± 28	808.442 ± 0.053
Mass error	0.531 ± 0.835	37 ± 50	808.431 ± 3.572
RCS thrust error	0.133 ± 0.173	22 ± 28	808.461 ± 0.055
Irregular gravity field	0.261 ± 0.438	22 ± 28	808.425 ± 0.320
Thrust misalignment	0.371 ± 1.539	35 ± 105	808.451 ± 0.058

Sensor Bias Biases in the accelerometer and gyroscope have not been included thus far. Biases are introduced, as given by Section 8.1. At initialisation of a flight, the accelerometer bias set randomly in all three axes, normal distributed with $\sigma = 10^{-4}\,\mathrm{m/s^2}$. In the same way, the gyroscope bias is also set randomly in all three axes, normal distributed with $\sigma = 5 \cdot 10^{-9}\,\mathrm{rad/s}$. The simulations show no significant difference in either the final conditions or the propellant use.

Sensor Misalignment and Scaling Errors Accelerometer and gyroscope misalignment and scaling errors are introduced as discussed in Sections 8.1 and 8.3. The misalignments and scaling errors are randomly determined at the initialisation of each flight, with a normal distribution. For the accelerometer, both the misalignments and scaling errors are set with $\sigma = 10^{-3}$. For the gyroscope, the misalignments are set with $\sigma = 10^{-3}$, while the scaling errors are set with $\sigma = 10^{-4}$.

With these perturbations, the averages don't change much. However, the spread does. The x, y and z standard deviations change from $\pm 0.187\,\mathrm{m}$, $\pm 0.196\,\mathrm{m}$ and $\pm 0.182\,\mathrm{m}$, to $\pm 0.552\,\mathrm{m}$, $\pm 0.583\,\mathrm{m}$ and $\pm 0.275\,\mathrm{m}$ respectively. Similarly, the velocity standard deviation changes from $\pm 41\,\mathrm{mm/s}$ on all axes, to $\pm 61\,\mathrm{mm/s}$, $\pm 75\,\mathrm{mm/s}$ and $\pm 46\,\mathrm{mm/s}$ for u, v and w, respectively. Furthermore, the standard deviation in propellant changes from $\pm 0.05\,\mathrm{kg}$ to $\pm 0.17\,\mathrm{kg}$. Overall, these figure still meet the requirements of Section 2.5.

Initial Position The initial position at PDI is normally varied in all three axes by 100 metres (1σ). The simulations show no deterioration of the targeting performance at the TG. The propellant use changed from 808.45 ± 0.05 kg, to 808.46 ± 0.06 kg.

Initial Velocity The initial velocity is also varied at PDI. The perturbation applies to all three axes, where the velocity is varied by 10 m/s (1σ). No deterioration was found in the targeting performance at TG. However, the propellant used increased from 808.45 ± 0.05 kg, to 808.94 ± 2.52 kg. The variation in initial velocity directly influences the ΔV required, and thereby, also the propellant mass.

Lunar Rotation Landing the spacecraft on the equator of a rotating Moon, does not change the landing performance significantly. The only true difference is that 1.5 kg less propellant is needed on average, because the spacecraft's initial velocity with respect to the Lunar surface is lower.

3rd Body Perturbations Simulations are run that include the Earth and the Sun as third body perturbations. The spacecraft is landed at 0° longitude and 0° latitude so that the Earth is at its closest approach. The epochs defined in Section 6 are used. The simulations show no significant change in position errors, velocity errors or propellant mass.

Mass Error The mass offset of the spacecraft is uniformly varied from -4 kg to +4 kg. This offset is added to the true mass of the spacecraft, while the GNC-loop assumes the nominal mass for each flight. The position distribution at TG is $x = \pm 0.21$ m, $y = -0.13 \pm 0.49$ m and $z = 0.03 \pm 1.67$ m. Even though, $3\sigma = 1.67$ m for the z-position, only 1.75% of the flights exceed the 1 metre requirement set by Section 2.5.

On average the velocity is zero, with ± 44 mm/s in u and v and ± 103 mm/s for w. The propellant use is 808.43 ± 3.57 kg.

RCS Thrust Error Varying the truly applied torque from 90% to 110% of the commanded torque has no significant effect on either the position, velocity or the propellant mass. As Section 7.3 showed before, only the control effort increases by a few percent.

Irregular Gravity Field To determine the influence of irregularities in the gravity field, a batch of simulations are run for each of the landing sites defined in Section 6. The combined result show no change in the velocity errors at AG. However, the x performance changed from

$\pm 0.19\ \mathrm{m}$ to $\pm 0.28\ \mathrm{m}$, while the y performance changed from $-0.10 \pm 0.20\ \mathrm{m}$ to $-0.09 \pm 0.32\ \mathrm{m}$ and z changed from $-0.01 \pm 0.18\ \mathrm{m}$ to $\pm 0.75\ \mathrm{m}$. The propellant mass changed from $808.45 \pm 0.05\ \mathrm{kg}$ to $808.43 \pm 0.32\ \mathrm{kg}$. This is all within the set requirements.

Thrust Misalignment The thrust is misaligned at 0.4° in a random direction. This misalignment creates a 22 Nm torque when the main engine is at full throttle, which is almost 17% of the total torque the spacecraft can generate with its RCS. The simulations show an average position error of $0.01 \pm 1.30\ \mathrm{m}$, $-0.26 \pm 1.12\ \mathrm{m}$ and $-0.03 \pm 0.24\ \mathrm{m}$ in x, y and z, respectively. Although this shows a large increase with respect to the reference scenario, it still meets all requirements. The propellant mass remains practically unchanged at $808.45 \pm 0.06\ \mathrm{kg}$.

9 Conclusions and Recommendations

This chapter provides a short summary of all the conclusions found in this study. Subsequently, the chapter continues with listing recommendations for the practical use of convex guidance and future studies.

9.1 Conclusions

The main goal of this thesis is to answer the question:

Is convex guidance robust enough for use in a guidance, navigation and control loop, to the purpose of a pin-point Lunar landing?

In analysing convex guidance, this study identified that the thrust constraint can be approximated by a first-order function instead of a second-order function, simplifying the expression and easing the later transcription to conical form. Furthermore, the mass constraint was found to be redundant, after which it was removed from convex guidance, improving its computational speed. Ultimately, the expression of convex guidance was reformulated in a conical form to remove the need for intermediary modelling languages, further improving solving speed. Regarding the errors introduced by simplifying assumptions, it was found that neglecting Lunar rotation and assuming a spherical Lunar gravity field introduces the largest landing errors (kilometres and hundreds of meters, respectively). Where the Lunar rotation can easily be introduced to convex guidance (Section 6), modelling higher-order terms of the gravity field is computationally very expensive, and is therefore not recommended. In fact, a trajectory tracker is very capable of correcting for gravitational anomalies.

Another important aspect that is neglected by convex guidance are the attitude dynamics of the spacecraft. As a result, the commanded attitude of the spacecraft can change instantaneously over large angles, for example, in a landing site redirect for hazard avoidance. In this form of convex guidance, it is meaningless to impose initial- or final-boundary conditions on the attitude, because the thrust vector will just comply at the boundaries, regardless of the attitude at

any other node. The attitude at one node, is in no way connected with the attitude at another node. Attitude controllers and trajectory trackers can be tuned to remain stable with these large discontinuities and return the spacecraft to the optimal trajectory. However, an excursion from the nominal path could not be prevented during an attitude discontinuity. Therefore, it is advised to introduce an angular rate limiter to convex guidance, to allow for the meaningful use of initial- and final-boundary attitude conditions, preventing large discontinuities.

A further complication regarding the spacecraft attitude, is the assumption of a zero-order hold function for the control vector (thrust) in combination with a large discretisation step (five seconds for the MBP and one second for the AP). This forces the spacecraft to hold the attitude for a short time, before instantaneously jumping to the next attitude. As a result the spacecraft has to build and stop its angular momentum repeatedly, wasting propellant. The discretisation step cannot be made smaller, because that will greatly increase the problem size and exponentially increase the computational effort, especially for the MBP. In turn, this greatly reduces the practicality of employing convex guidance as an on-board guidance algorithm for quick trajectory optimisation on the fly.

For a solution, the discontinuous control vector can be linearly interpolated. This proved to save 98% of the attitude control effort, because the spacecraft has a much more constant angular rate. However, this creates a mismatch with the optimal trajectory, which is an integration of a discontinuous attitude. The trajectory tracker picks up on these deviations and sends corrective commands to the attitude controller, trying to re-introduce the zero-order hold behaviour. However, because the trajectory tracker allows for a margin with respect to the optimal trajectory, this effect is less severe than directly using the zero-order hold commanded attitude. Nonetheless, it is advised to implement an first-order control function in convex guidance, so that it returns a smooth attitude command, which also corresponds with the optimal trajectory.

Convex guidance was not found to have a significant effect on the navigational performance of the EKF. The performance ratings of the sensors are by far the most dominant factor in the navigation accuracy. And in the end, the landing cannot be more precise than the uncertainty in navigation. Therefore, the navigation errors are likely to overshadow the guidance and control errors for most missions.

Overall, it is concluded that convex guidance can be used as a guidance algorithm to unify the MBP and the AP for a pin-point Lunar landing. The GNC loop managed to deliver the spacecraft at TG with position errors in the order of decimetres, reliably. However, it is recommended to improve the robustness of the algorithm by implementing an angular rate limiter to allow for the meaningful use of initial- and final-boundary attitude conditions.

9.2 Recommendations

- Implement convex guidance with a first-order, or higher, function for optimisation of the control vector, to smoothen the optimal control. This can greatly reduces the control effort required to track the attitude.

- Study the expansion of convex guidance with an angular rate limiter, reducing discontinuities in the attitude. This would greatly improve convex guidance, as it would allow for to meaningful application of initial and final attitude constraints, giving a much better guarantee that the optimal trajectory can physically be flown.

- Although the GNC loop in this study remains stable under large position or attitude errors, landing redirects have not been studied. Therefore, it is recommended to study the performance and robustness of convex guidance for diverts. It is expected that the addition of an angular rate limiter will be most important here.

- This study only investigated the performance of convex guidance for landing from a LLO. Therefore, it is advised to also study the use of convex guidance for different landing profiles, like directly landing from a hyperbolic orbit or introducing a hover period for hazard detection.

- This study considered a landing at the Moon, a body without atmosphere. Aerodynamics were neglected when Açikmeşe and Ploen (2007) developed convex guidance, even though they had a Mars landing in mind. Although Mars' atmosphere is tenuous, it is significant. Therefore, it is interesting to study the effect of neglecting the Martian atmosphere during a powered descent (after the main braking phase with a parachute) and to see if these errors can also be corrected by a trajectory tracker, instead of expanding the convex guidance model.

- A comparison with other guidance algorithms is recommended, because only the performance and robustness of convex guidance was studied without putting it in the perspective of other algorithms. It is especially interesting to see how the unified convex guidance algorithm compares to landing techniques that equip different guidance algorithms for the different phases of the landing.

Bibliography

Açikmeşe, B. and Ploen, S. (2005). "A Powered Descent Guidance Algorithm for Mars Pinpoint Landing". AIAA 2005 GNC Conference and Exhibit. DOI: 10.2514/6.2005-6288.

Açikmeşe, B. and Ploen, S. (2007). "Convex Programming Approach to Powered Descent Guidance for Mars Landing". Journal of Guidance Control and Dynamics, vol. 30, no. 5, pp. 1353–1366. DOI: 10.2514/1.27553.

Allione, M., Blackford, A., Mendez, J., and Whittouck, M. (1968). *Guidance, Flight Mechanics and Trajectory Optimization: Volume VI - The N-Body Problem and Special Perturbation Techniques*. NASA CR-1005.

Amzajerdian, F., Pierrottet, D., Petway, L., Hines, G., and Roback, V. (2011). "Lidar Systems for Precision Navigation and Safe Landing on Planetary Bodies". International Symposium on Photoelectronic Detection and Imaging 2011. DOI: 10.1117/12.904062.

Andrews, T. (2012). "Computation Time Comparison Between Matlab and C++ using Launch Windows". American Institute of Aeronautics and Astronautics. `http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1080&context=aerosp`.

Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics. ISBN-10: 0-89871-491-5. ISBN-13: 978-0-89871-491-3.

Bennett, F. (1970). "Lunar Descent and Ascent Trajectories". AIAA 8th Aerospace Sciences Meeting. DOI: 10.2514/6.1970-25.

Berman, A., Zarchan, P., and Lewis, B. (2014). "Comparisons Between the Extended Kalman Filter and the State-Dependent Riccati Estimator". Journal of Guidance, Controla and Dynamics, vol. 37, no. 5, pp. 1556-1567. DOI: 10.2514/1.G000332.

Bhagat, M. (2016). "Convex Guidance for Envisat Rendezvous". Master Thesis, Delft Univer-

sity of Technology.

Bryson, A. and Ho, Y. (1975). *Applied Optimal Control*. John Wiley & Sons.

Burden, R. and Faires, J. (2011). *Numerical Analysis*. Brooks and Cole, Cengage Learning, 9th edition. ISBN-10: 0-538-73564-3. ISBN-13: 978-0-538-73564-3.

Crassidis, J. and Markley, F. (2003). "Unscented Filtering for Spacecraft Attitude Estimation". Journal of Guidance, Control and Dynamics, vol. 26, no. 4, pp. 536-542. DOI: 10.2514/2.5102.

Domahidi, A., Chu, E., and Boyd, S. (2013). ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076.

Ely, T., Heyne, M., and Riedel, J. (2012). "Altair Navigation Performance During Translunar Cruise, Lunar Orbit, Descent and Landing". Journal of Spacecraft and Rockets, vol. 49, no. 2, pp. 295-317. DOI: 10.2514/1.52233.

Fisackerly, R., Pradier, A., Gardini, B., Houdou, B., Philippe, C., de Rosa, D., and Carpenter, J. (2011). "The ESA Lunar Lander Mission". AIAA 2011 Space Conference. AIAA 2011-7217, DOI: 10.2514/6.2011-7217.

Gerth, I. (2014). "Convex Optimization for Constrained and Unified Landing Guidance". Master Thesis, Delft University of Technology.

Gopal, M. (1993). *Modern Control System Theory*. John Wiley & Sons. ISBN-10: 047022157-7. ISBN-13: 978-047022157-0.

Gross, J., Gu, Y., Gururajan, S., Seanor, B., and Napolitano, M. (2010). "A Comparison of Extended Kalman Filter, Sigma-Point Kalman Filter, and Particle Filter in GPS/INS Sensor Fusion". AIAA 2010 GNC Conference. DOI: 10.2514/6.2010-8332.

Groves, P. (2013). *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2nd edition. ISBN-10: 1-60807-005-0. ISBN-13: 978-1-60807-005-3.

Holtkamp, G. (2014). "Descent, Touchdown and Repositioning of a hopping planetary lander on Enceladus". Master Thesis, Delft University of Technology.

Hull, T., Enright, W., Fellen, B., and Sedgwick, A. (1972). "Comparing Numerical Methods for Ordinary Differential Equations". SIAM Journal on Numerical Analysis, vol. 9, no. 4, pp. 603-637. http://www.jstor.org/stable/2156215?seq=1#page_scan_tab_contents.

Johnson, A. and Montgomery, J. (2008). "Overview of Terrain Relative Navigation Approaches for Precise Lunar Landing". Aerospace Conference, IEEE. DOI: 10.1109/AERO.2008.4526302.

Kerr, M., Hagenfeldt, M., Ospina, J., Ramón, J., Peñín, L., Mammarella, M., Bidaux, A., and Colmenarejo, P. (2013). "ESA Lunar Lander: Approach Phase Concept and G&C Performance". AIAA 2013 GNC Conference. AIAA 2013-5018, DOI: 10.2514/6.2013-5018.

Klees, R. and Dwight, R. (2013). *Applied Numerical Analysis*. Delft University of Technology. Course AE2220-I. Article: 06917710035.

Konopliv, A., Park, R., Yuan, D., Asmar, S., Watkins, M., Williams, J., Fahnestock, E., Kruizinga, G., Paik, M., Strekalov, D., Harvey, N., Smith, D., and Zuber, M. (2013). "The JPL Lunar Gravity Field to Spherical Harmonic Degree 660 from the GRAIL Primary Mission". Journal of Geophysical Research: Planets, vol. 118. DOI: 10.1002/jgre.20097.

Lay, D. (2012). *Linear Algebra and Its Applications*. Pearson, 4th edition. ISBN-10: 0-321-62335-5. ISBN-13: 978-0-321-62335-5.

Lee, A., Ely, T., Sostaric, R., Strahan, A., Riedel, J., Ingham, M., Wincentsen, J., and Sarani, S. (2010). "Preliminary Design of the Guidance, Navigation and Control System of the Altair Lunar Lander". AIAA 2010 GNC Conference. AIAA 2010-7717, DOI: 10.2514/6.2010-7717.

Lissauer, J. and de Pater, I. (2013). *Fundamental Planetary Science: Physics, Chemistry and Habitability*. Cambridge University Press. ISBN-10: 0-521-61855-X. ISBN-13: 978-0-521-61855-7.

Liu, X. and Lu, P. (2014). "Solving Non-Convex Optimal Control Problems by Convex Optimization". Journal of Guidance, Control, and Dynamics, vol. 37, no. 3, pp. 750-765. DOI: 10.2514/1.62110.

Meriam, J. and Kraige, L. (2008). *Engineering Mechanics: Dynamics*. John Wiley & Sons, Inc, 6th edition. ISBN-10: 0-471-78703-5. ISBN-13: 978-0-471-78703-7.

Nise, N. (2008). *Control Systems Engineering*. John Wiley & Sons, Inc, 5th edition. ISBN-10: 0-470-16997-4. ISBN-13: 978-0-470-16997-1.

Parker, P. (1974). *The Apollo On-board Computers*. NASA, `http://history.nasa.gov/afj/compessay.htm`.

Renzetti, N. (1969). "Tracking and Data System Support for Surveyor Missions 1 and 2".

Technical Memorandum 33-301, NASA.

Rhudy, M., Gu, Y., and Napolitano, M. (2013). "Does the Unscented Kalman Filter Converge Faster than the Extended Kalman Filter? A Counter Example". AIAA 2013 GNC Conference. DOI: 10.2514/6.2013-5198.

Riehle, M., Diedrich, T., Perigo, D., Kraus, S., and Bühner, M. (2012). "Propulsion System for the European Lunar Lander - Development Status and Breadboarding Activities". 7th ESA Space Propulsion Conference. http://www.ecosimpro.com/wp-content/uploads/2015/02/SpacePropulsion2012_2357503.pdf.

Rogers, G., Schwinger, M., Kaidy, J., and Strikwerda, T. (2006). "Autonomous Star Tracker Performance". 57th International Astronautical Congress. DOI: 10.2514/6.IAC-06-D1.2.01.

Schulte, G. (2004). "400 N Engine Qualification Test Report". EADS Space Transportation. Tech. rep. 400N-RILAN-TRP-0006. In Gerth (2014).

Shuster, M. (1993). "A Survey of Attitude Representations". The Journal of the Astronautical Sciences, vol. 41, no. 4, pp. 439-517.

Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 1st edition. ISBN-10: 0-471-70858-5. ISBN-13: 978-0-471-70858-2.

St-Pierre, M. and Gingras, D. (2004). "Comparison between the Unscented Kalman Filter and the Extended Kalman Filter for the Position Estimation Module of an Integrated Navigation Information System". IEEE 2004 Intelligent Vehicles Symposium. DOI: 10.1109/IVS.2004.1336492.

Sun, Z., Jia, Y., and Zhang, H. (2013). "Technological Advancements and promotion roles of Chang'e-3 Lunar Probe Mission". Science China: Technological Sciences. DOI: 10.1007/s11431-013-5377-0.

Turner, M. (2004). *Rocket and Spacecraft Propulsion*. Springer, 2nd edition. ISBN-10: 3-540-22190-5. ISBN-13: 978-3-540-22190-6.

VanDyke, M., Schwartz, J., and Hall, C. (2004). "Unscented Kalman Filtering for Spacecraft Attitude State and Parameter Estimation". American Astronomical Society. AAS-04-115. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.5877&rep=rep1&type=pdf.

Wakker, K. (2015). *Fundamentals of Astrodynamics*. Delft University of Technology, 1st edition. Course AE4874. ISBN-10: 94-6186-419-1. ISBN-13: 978-94-6186-419-2.

Wertz, J. (2001). *Orbit & Constellation Design & Management*. Microcosm Press, Springer, renewed edition. ISBN-10: 1-881883-07-8. ISBN-13: 978-1-881883-07-4.

Wie, B. (1998). *Space Vehicle Dynamics and Control*. American Institute of Aeronautics and Astronautics, 1st edition. ISBN-10: 1-56347-261-9. ISBN-13: 978-1-56347-261-9.

Wie, B., Weiss, H., and Arapostathis, A. (1988). "A Quaternion Feedback Regulator for Spacecraft Eigenaxis Rotations". AIAA 1988 GNC Conference. DOI: 10.2514/6.1988-4128.

Appendices

A Simulator

A.1 Architecture

For running Monte Carlo simulations, there is a *Monte Carlo Manager* that calls the *Flight Manager* with a set of different conditions, as shown in Figure A.1. The *Flight Manager* exports the results to text files, which are analysed by a separate *Data Analyser* script. The *Flight Manager* block in Figure A.1 contains the entire architecture of Figure A.2 to simulate individual flights.

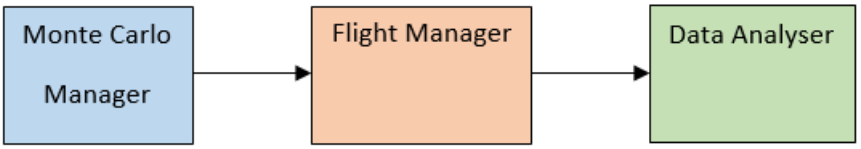


Figure A.1: Flow of Monte Carlo simulations.

Figure A.2 shows a detailed architecture of the simulation of a flight. A simulation starts at the grey *Spacecraft* block, where the state of the spacecraft is set to its initial conditions. Subsequently, the *Sensor* block takes measurements of the spacecraft's state (adding errors to the true state), which are passed on to the *Navigation* block. From the measurements, the navigation algorithm attempts to estimate the state, effectively reducing the measurement errors. From the estimated state, the *Guidance* block determines where the spacecraft is located with respect to the optimal trajectory (which has been determined at initialisation, based on the initial conditions), and what corrections need to be made to remain or return to the optimal trajectory. Subsequently, the *Control* block compares the desired attitude from the guidance algorithm to the estimated attitude from the navigation algorithm. Based on the attitude error, the attitude control algorithm determines the required torque to rotate the spacecraft (and thereby the thrust vector), so that the spacecraft follows the optimal trajectory given by the guidance algorithm. The *Actuators* block applies the commanded torque and throttle setting, but also introduces delays and errors, to model actuator performance.

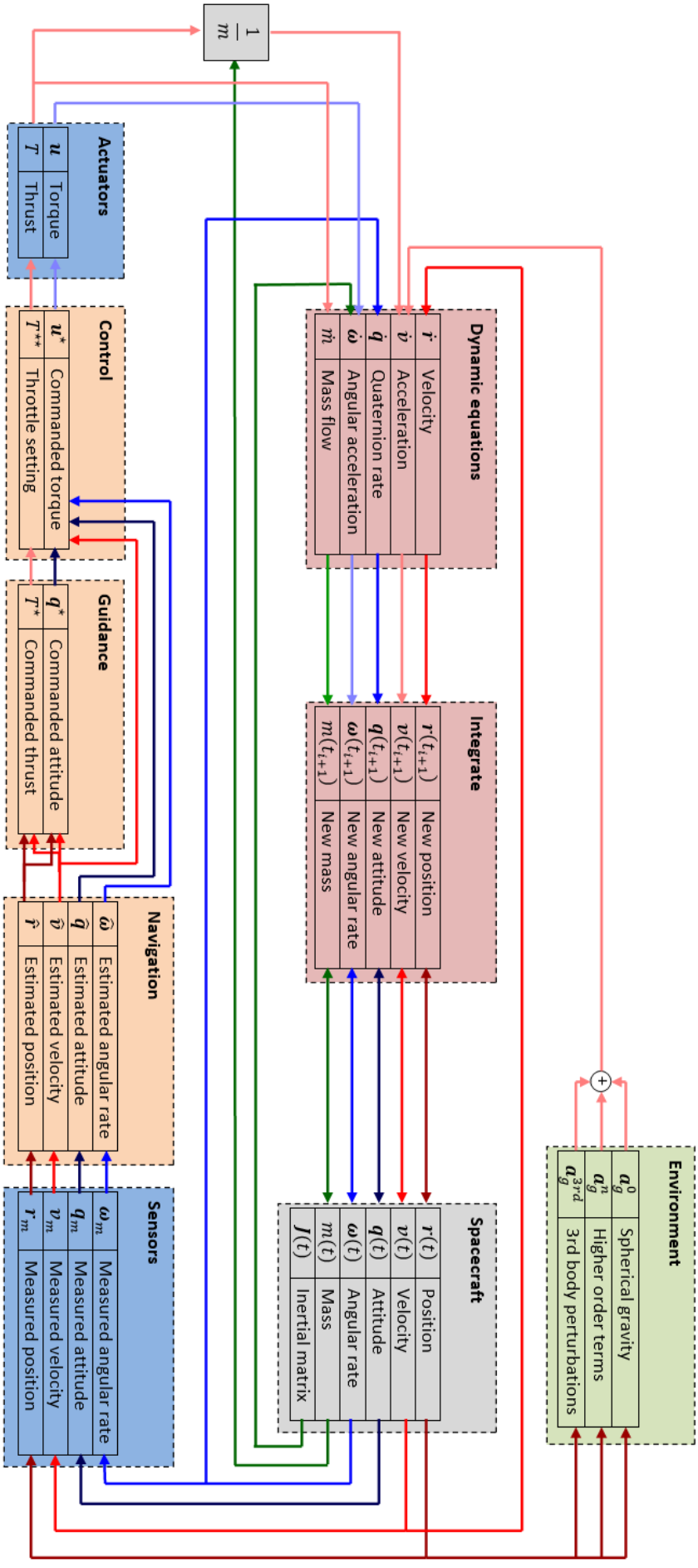


Figure A.2: Simulator architecture.

At the same time as the state passes through the GNC blocks at the bottom of Figure A.2, the state also passes through the *Environment* block. Based on the position, the *Environment* block determines the acceleration caused by the Lunar gravity field and the gravity field of perturbing bodies. The gravitational accelerations from the *Environment* and the torque and thrust from the *Actuators* are used to calculate the time derivative of the spacecraft's state in the *Dynamic equations* block. Using the dynamic equations, the *Integrate* block propagates the state of the spacecraft to the next time step in the simulation. All blocks in Figure A.2 are initialised by the *Flight Manager*, which also oversees the simulation and logs the parameters of all subcomponents during the flight.

A.2 Programming Language

The used programming language for the simulator mainly affects the computational performance and the ease of development. The two most dominant languages used for scientific computing are MATLAB and C++. Therefore, this section performs a trade-off between these two languages to be used for this research project.

MATLAB is a great prototyping software package that allows for quick programming and data visualization. Therefore, MATLAB is very well suited for evaluating and optimizing the performance of algorithms. Furthermore, Gerth (2014) has developed the convex guidance algorithm discussed in Section 4 in MATLAB. Therefore, using MATLAB would ease the implementation of this guidance algorithm in the overall simulator. Although MATLAB is highly optimized for matrix operations, it is far less efficient while using loops. Therefore, a common way to improve the performance of a MATLAB script is to vectorize most operations. However, not all algorithms can be vectorized, like iterative processes.

C++ is much more bare-bone than MATLAB. This makes it harder to program and debug applications, but this freedom allows for a much greater flexibility in return. Furthermore, because C++ allows the user to be closer to the hardware, it greatly outperforms MATLAB in computational speed, especially when considering loops. Andrews (2012) shows in a comparison of computational time that C++ can be as much as 500 times faster than MATLAB, when solving Lambert's problem iteratively for an interplanetary flight from the Earth to Mars. Furthermore, C++ allows for multithreading, which can be of great benefit when running Monte Carlo simulations. MathWorks provides the Parallel Computing Toolbox[1] for MATLAB, but from experience this again provides less flexibility than C++. Additionally, where MathWork's Parallel Computing Toolbox is paid for, multithreading is natively supported by C++ since the C++11[2] standard, meaning it is practically freely available.

[1] `http://www.mathworks.com/products/parallel-computing/`

[2] 2011 standard for C++ by the International Organization for Standardization (ISO)

Because many simulations are run for this research, the simulator is developed in C++. This is purely out of the consideration for computational speed. C++ is faster than MATLAB and the multithreading support allows for several landings to be simulated in parallel. All data is exported from the C++ programme in a CSV format. This allows MATLAB to be used for analysis of the results and data visualisation.

A.3 Numerical Integration

Many different numerical integrators exist to solve initial value problems. Considering multi-step and single-step methods, multi-step methods reach high orders of accuracy with relatively little computational effort. However, multi-step methods need to be initialized using multiple epochs. Klees and Dwight (2013) add that when using time-step control, multi-step methods need to be re-initialized for every change in time-step. Furthermore, as pointed out by Açikmeşe and Ploen (2007) and as discussed in Chapter 4, convex guidance returns a bang-bang control solution which is not compatible with multi-step solvers according to Allione et al. (1968). Therefore, only single-step methods are considered.

Hull et al. (1972) performed an extensive study into different numerical methods[3] for solving initial value problems. For very high accuracy integration they suggest using the Bulirsch-Stoer method, which is based on extrapolation. However, for problems where only moderate accuracy is required, Hull et al. (1972) suggest Runge-Kutta methods, because of their relatively light computational load. For this study only a moderate accuracy is required because a Moon landing only has a short integration time and the expected errors - purely introduced by the guidance algorithm alone - are in the order of centimetres according to Gerth (2014). Therefore, a Runge-Kutta method is selected for this study.

For a streamlined simulator design, it is desired to use an integrator step size that directly steps to the following data point. That is, if the guidance system has a discretisation step of 5 seconds, than the integrator should also be targeted to use a step size of 5 seconds. This assures that the integration stays synchronised with the guidance system. The integration step might be smaller to obtain a higher accuracy, but it may never exceed the 5 second step size, or it will skip control commands from the guidance algorithm.

To determine the order of the Runge-Kutta integrator, a full revolution of the Lunar descent orbit (10x100 km orbit as given by Section 2.4) is simulated in a spherical gravity field. Because no perturbations are assumed, the spacecraft should return to its exact initial conditions after one orbit. Furthermore, because during the Main Braking Phase (MBP) the guidance al-

[3] A selection of Adams schemes, extrapolating schemes (amongst others, Bulirsch-Stoer) and different order Runge-Kutta schemes.

gorithm uses a discretisation step of 5 seconds (see Section 6), the integration step cannot exceed 5 seconds. With the very popular fourth-order Runge-Kutta (RK4) method, the error after an orbit is in the order of 0.1 millimetres. Note however, that a full orbit takes over 1 hour and 53 minutes, while the entire landing from the Powered Descent Initiation (PDI) to the Terminal Gate (TG) takes just over 10 minutes. Therefore, this integration method is expected to yield even higher accuracies than the stated 0.1 millimetres. Because the guidance algorithm is expected to produce errors in the order of metres for the MBP (see Section 6), an RK4 integrator with a step size of 5 seconds provides sufficient accuracy.

Section 6 establishes that a 100×100 degree and order spherical harmonics model is used to represent gravitational irregularities. For the Moon, this corresponds with 109 km features. Furthermore, Section 2.4 shows that the initial velocity of the spacecraft is 1.7 km/s, and decreases rapidly. Therefore, the relatively large integration step poses no problem with respect to the spatial resolution of the Lunar gravity field for these simulations.

Because the guidance algorithm discretizes the Approach Phase (AP) with steps of 1 second (see Section 6), the integrator cannot exceed this step size during the AP simulation. Running the same full orbit simulation with a step size of 1 second, yields an error in the order of 0.1 micrometres for the RK4 method. This is well below the expected guidance error, which is in the order of centimetres for the AP (see Section 6). These analyses show that the RK4 method provides sufficient accuracy for the entire landing phase, even when using the maximum allowable step sizes of 5 seconds and 1 second for the MBP and AP, respectively.

Burden and Faires (2011) give the RK4 method as follows, where $\mathbf{x}$ is the state vector, h is the step size and $\mathbf{f}$ are the dynamic equations returning $\dot{\mathbf{x}}$.

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \tag{A.1a}$$

$$\mathbf{k}_1 = \mathbf{f}(t, \mathbf{x})h \tag{A.1b}$$

$$\mathbf{k}_2 = \mathbf{f}(t + h/2, \mathbf{x} + \mathbf{k}_1/2)h \tag{A.1c}$$

$$\mathbf{k}_3 = \mathbf{f}(t + h/2, \mathbf{x} + \mathbf{k}_2/2)h \tag{A.1d}$$

$$\mathbf{k}_4 = \mathbf{f}(t + h, \mathbf{x} + \mathbf{k}_3)h \tag{A.1e}$$

Numerical Considerations

The RK4 method assumes a continuous system. However, the optimal control given by the convex guidance algorithm is a discontinuous signal because it assumes zero-order hold, as discussed in Section 4.1.3. For an accurate integration this difference needs to be taken into account. Figure A.3 shows the integration of the control command by the RK4 method. It shows that both times t and $t+h/2$ are on the initial control command. However, $t+h$ coincides with the next command. Consequently, the $\mathbf{k}_1$, $\mathbf{k}_2$ and $\mathbf{k}_3$ vectors from Equation A.1 will 'see' the $\mathbf{u}_k$ command while $\mathbf{k}_4$ 'sees' the $\mathbf{u}_{k+1}$ command. Because the RK4 method assumes the signal to be continuous, it interpolates the control command so that it no longer represents a zero-order hold function. To get an accurate integration of the control command, the RK4 method is forced to use the same value of $\mathbf{u}$ for $t+h$ as it uses for t. This ensures a zero-order hold integration of the control command, while still integrating the further spacecraft dynamics using the RK4 method.

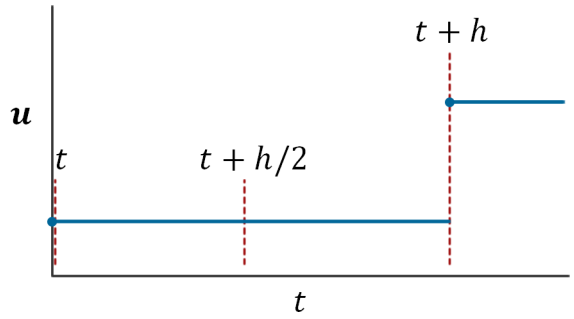


Figure A.3: RK4 integrator step size h on the discontinuous control signal.

Using this method of forcing zero-order hold requires use of proper synchronisation of the integration with the control commands. Otherwise, numerical errors in the representation of the in-simulation time t may build up to significant proportions. Previous section showed that the discretisation step of the guidance algorithm can directly be used as the step size of the RK4 integrator. However, when an additional system (like navigation) is implemented in the simulation that runs at a much higher frequency than the guidance algorithm, the integration step size needs to be lowered accordingly to prevent skipping any of the frames of the new system. For example, when using a step size of 0.01 seconds, this number is represented by a floating-point format. When using double-precision, this time-step is not exactly 0.01 seconds, but actually:
0.010 000 000 000 000 000 21 seconds

As a test, this number is summed 10 000 times in C++ to get to the hypothetical simulation time of 100 seconds. The true value obtained by the double-precision floating-point format is actually:

100.000 000 000 014 253 49 seconds.

Going over the targeted time by an error this small does not influence the simulation results. However, when the step size is added another 50 000 times to get to a simulation time of 600 seconds, the simulation time starts to fall short of the targeted time:
599.999 999 999 599 367 58 seconds. It is not fully understood why the simulator goes from overestimating the time to underestimating it, but it is expected to do something with the change of the exponent, from 1 to 2. No matter the reason, this time underestimation significantly influences the simulation results.

Even when rounding the underestimated time of 600 seconds, it is still off by $4 \cdot 10^{-10}$ seconds. When t falls short $4 \cdot 10^{-10}$ seconds of $\mathbf{u}_k$, the integrator enforces a zero-order hold on command $\mathbf{u}_{k-1}$ for the duration of step size h, because t is still in the region of $\mathbf{u}_{k-1}$ instead of $\mathbf{u}_k$, as shown by Figure A.4. When running simulations for the MBP as described in Section 2.4, the guidance algorithm proves to be off by just over 6 metres at AG. However, when code is added to check and correct small time offsets, it was found to make time corrections of 10^{-14} and 10^{-13} seconds throughout the MBP. With actively synchronising the simulation time with the control command - to prevent situations as shown in Figure A.4 - the guidance algorithm proved to be off by almost 12 metres at AG. Almost twice as much.

To verify this result, the simulation was also run with a time-step of 5 seconds (this time-step matches the discretisation step of the guidance algorithm for the MBP, and Section A.3 proved that a 5 seconds integration step yields a high enough accuracy, sub-millimetre). The simulation with a time-step of 5 seconds yields the exact same results as the 0.01 seconds time-step simulation with the synchronisation activated (just under 12 metres). Furthermore, no corrections were enforced by the synchronisation algorithm when the time-step of 5 seconds was used, indicating that any numerical errors stayed sufficiently small.

These results suggest that when using a too small time-step for integration, numerical errors in time-keeping may cause the integration to desynchronise with the guidance trajectory, causing the integration of wrong/old commands. The resulting control errors happen to work in favour of the guidance algorithm, meaning that the deviation from the landing site reduces. Therefore, lowering the integration step seems to give a more accurate result, while in fact it does not.

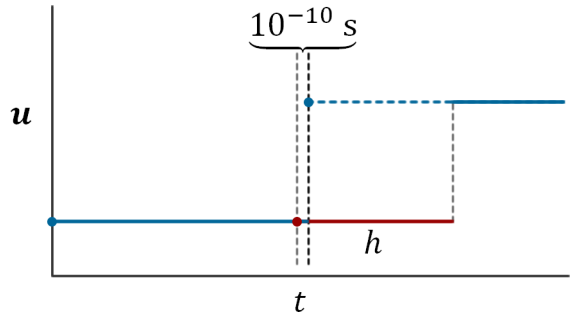


Figure A.4: Numerical error in simulation time t and the consequential zero-order hold of the control signal at fixed time-step h.

B Guidance Variables

This appendix gives an overview of all variables used in convex guidance in Chapter 4, as provided by Gerth (2014).

Table B.1: Input variables.

$\mathbf{r}$	Position	$[\mathrm{m}]$
$\mathbf{g}$	Gravitational acceleration	$[\mathrm{m/s^2}]$
m	Mass	$[\mathrm{kg}]$
$\mathbf{T}$	Thrust	$[\mathrm{N}]$
T_h	Upper limit thrust	$[\mathrm{N}]$
T_l	Lower limit thrust	$[\mathrm{N}]$
a	Thrust to mass flow ratio	$[\mathrm{s/m}]$
α	Glide slope	$[\mathrm{rad}]$
$\hat{\mathbf{n}}$	Attitude target	$[-]$
θ	Attitude margin	$[\mathrm{rad}]$
t_f	Time of flight	$[\mathrm{s}]$

Table B.2: Change of variables.

Γ	$\geq \|\mathbf{T}\|$
$\boldsymbol{\tau}$	$= \mathbf{T}/m$
σ	$= \Gamma/m$
z	$= \ln m$
$\dot{z}$	$= -a\sigma$
$z_1(t)$	$= \ln\left(m_0 - aT_l t\right)$
$z_2(t)$	$= \ln\left(m_0 - aT_h t\right)$
$\mu_1(t)$	$= T_l e^{-z_1(t)}$
$\mu_2(t)$	$= T_h e^{z_2(t)}$
N	$= \lceil t_f \rceil$
Δt	$= tf/N$

Table B.3: Discrete state propagation.

$$
\begin{aligned}
\mathbf{x} &= \begin{bmatrix}\mathbf{r}^T & \dot{\mathbf{r}}^T & z\end{bmatrix}^T \in \mathbb{R}^7 \\
\mathbf{u} &= \begin{bmatrix}\boldsymbol{\tau}^T & \sigma\end{bmatrix}^T \in \mathbb{R}^4 \\
\dot{\mathbf{x}} &= \mathbf{A}_c\mathbf{x} + \mathbf{B}_c\begin{bmatrix}\mathbf{g}^T & 0\end{bmatrix}^T + \mathbf{B}_c\mathbf{u} \\
\mathbf{A}_c &= \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0\end{bmatrix} \in \mathbb{R}^{7\times7} \\
\mathbf{B}_c &= \begin{bmatrix}\mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{I}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & -a\end{bmatrix} \in \mathbb{R}^{7\times4} \\
\mathbf{A}_d &= \mathbf{I}_{7\times7} + \mathbf{A}_c\Delta t + 0.5\mathbf{A}_c^2\Delta t^2 \\
\mathbf{B}_d &= \mathbf{B}_c\Delta t + 0.5\mathbf{A}_c\mathbf{B}_c\Delta t^2 \\
\mathbf{x}(k) &= \mathbf{A}_d^k\mathbf{x}_0 + \textstyle\sum_{j=1}^{k}\mathbf{A}_d^{k-j}\mathbf{B}_d\left\{\mathbf{u}(j) + \begin{bmatrix}\mathbf{g}^T(j) & 0\end{bmatrix}^T\right\}
\end{aligned}
$$

Table B.4: Stacking.

$$
\begin{aligned}
\mathbf{X} &= \begin{bmatrix}\mathbf{x}^T(1) & \cdots & \mathbf{x}^T(N)\end{bmatrix}^T \in \mathbb{R}^{7N} \\
\mathbf{X}_0 &= \begin{bmatrix}\mathbf{x}^T(0) & \cdots & \mathbf{x}^T(0)\end{bmatrix}^T \in \mathbb{R}^{7N} \\
\mathbf{U} &= \begin{bmatrix}\mathbf{u}^T(0) & \cdots & \mathbf{u}^T(N-1)\end{bmatrix}^T \in \mathbb{R}^{4N} \\
\mathbf{G} &= \begin{bmatrix}\begin{bmatrix}\mathbf{g}^T(0) & 0\end{bmatrix}^T & \cdots & \begin{bmatrix}\mathbf{g}^T(N-1) & 0\end{bmatrix}^T\end{bmatrix}^T \in \mathbb{R}^{4N} \\
\mathbf{S}_\mathbf{X} &= \begin{bmatrix}\mathbf{A}_d^1 & \mathbf{0}_{7\times7} & \mathbf{0}_{7\times7} \\ \mathbf{0}_{7\times7} & \ddots & \mathbf{0}_{7\times7} \\ \mathbf{0}_{7\times7} & \mathbf{0}_{7\times7} & \mathbf{A}_d^N\end{bmatrix} \in \mathbb{R}^{7N\times7N} \\
\mathbf{S}_\mathbf{U} &= \begin{bmatrix}\mathbf{B}_d & \mathbf{0}_{7\times4} & \cdots & \mathbf{0}_{7\times4} \\ \mathbf{A}_d\mathbf{B}_d & \mathbf{B}_d & \cdots & \mathbf{0}_{7\times4} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_d^{N-1}\mathbf{B}_d & \mathbf{A}_d^{N-2}\mathbf{B}_d & \cdots & \mathbf{B}_d\end{bmatrix} \in \mathbb{R}^{7N\times4N} \\
\mathbf{X} &= \mathbf{S}_\mathbf{X}\mathbf{X}_0 + \mathbf{S}_\mathbf{U}\mathbf{G} + \mathbf{S}_\mathbf{U}\mathbf{U} \\
\mathbf{X}_h &= \mathbf{S}_\mathbf{X}\mathbf{X}_0 + \mathbf{S}_\mathbf{U}\mathbf{G} \\
\mathbf{X} &= \mathbf{X}_h + \mathbf{S}_\mathbf{U}\mathbf{U}
\end{aligned}
$$

Table B.5: Extraction matrices and vectors, part 1.

$$\mathbf{E}_{\boldsymbol{\tau}} = \begin{bmatrix}\mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 1}\end{bmatrix} \in \mathbb{R}^{3\times 4}$$

$$\mathbf{e}_{\sigma} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T \in \mathbb{R}^4$$

$$\mathbf{e}_{N\sigma\Delta t} = \begin{bmatrix}\mathbf{e}_{\sigma}^T\Delta t & \cdots & \mathbf{e}_{\sigma}^T\Delta t\end{bmatrix}^T \in \mathbb{R}^{4N}$$

$$\mathbf{E}_{\sigma} = \begin{bmatrix}\mathbf{e}_{\sigma}^T & \mathbf{0}_{1\times 4} & \mathbf{0}_{1\times 4} \\ \mathbf{0}_{1\times 4} & \ddots & \mathbf{0}_{1\times 4} \\ \mathbf{0}_{1\times 4} & \mathbf{0}_{1\times 4} & \mathbf{e}_{\sigma}^T\end{bmatrix} \in \mathbb{R}^{N\times 4N}$$

$$\mathbf{E}_{\mathbf{xy}} = \begin{bmatrix}\mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 5}\end{bmatrix} \in \mathbb{R}^{2\times 7}$$

$$\mathbf{e}_{r_z} = \begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}^T \in \mathbb{R}^7$$

$$\mathbf{E}_{r_z} = \begin{bmatrix}\mathbf{e}_{r_z}^T & \mathbf{0}_{1\times 7} & \mathbf{0}_{1\times 7} \\ \mathbf{0}_{1\times 7} & \ddots & \mathbf{0}_{1\times 7} \\ \mathbf{0}_{1\times 7} & \mathbf{0}_{1\times 7} & \mathbf{e}_{r_z}^T\end{bmatrix} \in \mathbb{R}^{N\times 7N}$$

$$\mathbf{e}_{d} = \begin{bmatrix}0 & 0 & \tan\left(\pi/2-\alpha\right) & 0 & 0 & 0 & 0\end{bmatrix}^T \in \mathbb{R}^7$$

$$\mathbf{e}_{z} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}^T \in \mathbb{R}^7$$

$$\mathbf{E}_{z} = \begin{bmatrix}\mathbf{e}_{z}^T & \mathbf{0}_{1\times 7} & \mathbf{0}_{1\times 7} \\ \mathbf{0}_{1\times 7} & \ddots & \mathbf{0}_{1\times 7} \\ \mathbf{0}_{1\times 7} & \mathbf{0}_{1\times 7} & \mathbf{e}_{z}^T\end{bmatrix} \in \mathbb{R}^{N\times 7N}$$

$$\mathbf{E}_{k,\mathbf{U}} = \begin{bmatrix}\underbrace{\mathbf{0}_{4\times 4} \quad \cdots}_{\times(k-1)} & \mathbf{I}_{4\times 4} & \underbrace{\cdots \quad \mathbf{0}_{4\times 4}}_{\times(N-k)}\end{bmatrix} \in \mathbb{R}^{4\times 4N}$$

$$\mathbf{E}_{k,\mathbf{X}} = \begin{bmatrix}\underbrace{\mathbf{0}_{7\times 7} \quad \cdots}_{\times(k-1)} & \mathbf{I}_{7\times 7} & \underbrace{\cdots \quad \mathbf{0}_{7\times 7}}_{\times(N-k)}\end{bmatrix} \in \mathbb{R}^{7\times 7N}$$

$$\mathbf{E}_{\mathbf{x}(N)} = \begin{bmatrix}\underbrace{\mathbf{0}_{6\times 7} \quad \cdots}_{\times(N-1)} & \mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 1}\end{bmatrix} \in \mathbb{R}^{6\times 7N}$$

Table B.6: Extraction matrices and vectors, part 2.

$$\boldsymbol{\mu}_{1N} = \begin{bmatrix} \mu_1(1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mu_1(N) \end{bmatrix} \in \mathbb{R}^{N\times N}$$

$$\boldsymbol{\mu}_{2N} = \begin{bmatrix} \mu_2(1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mu_2(N) \end{bmatrix} \in \mathbb{R}^{N\times N}$$

$$\mathbf{z}_{1N} = \begin{bmatrix} \ln\left(m_0 - aT_l\Delta t \cdot 1\right) \\ \vdots \\ \ln\left(m_0 - aT_l\Delta t \cdot N\right) \end{bmatrix} \in \mathbb{R}^{N}$$

$$\mathbf{z}_{2N} = \begin{bmatrix} \ln\left(m_0 - aT_h\Delta t \cdot 1\right) \\ \vdots \\ \ln\left(m_0 - aT_h\Delta t \cdot N\right) \end{bmatrix} \in \mathbb{R}^{N}$$

$$\mathbf{1}_N = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^T \in \mathbb{R}^N$$

C Spherical Harmonics Degree and Order

The following figures show the change in landing position due to modelling a higher degree gravity field. The change is with respect to a perfect spherical gravity field (degree and order 0). Subfigures *a* show the contribution from the main braking phase. Subfigures *b* show the contribution from the approach phase. The landing sites refer to the landing sites defined in Section 6, Figure 6.6. These figures backup the conclusion drawn from Figure 6.7: the position for the MBP converges by degree and order 100, while the position for the AP only converges by degree and order 140.

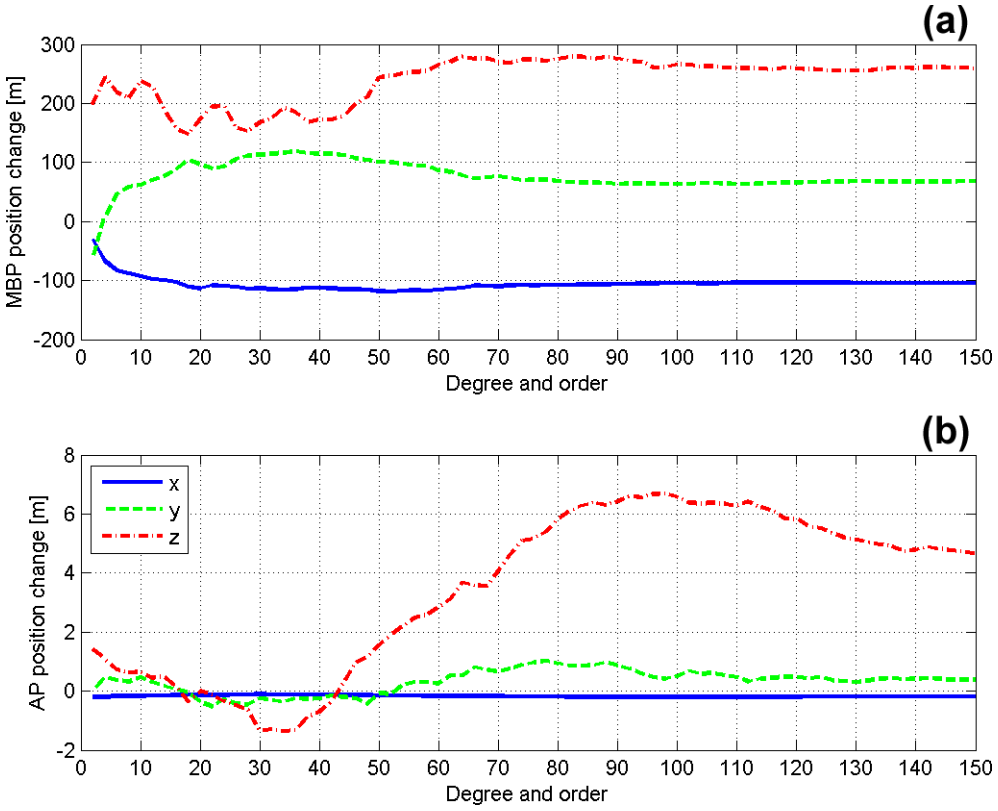


Figure C.1: Landing site 2 (88° West, 65° South).

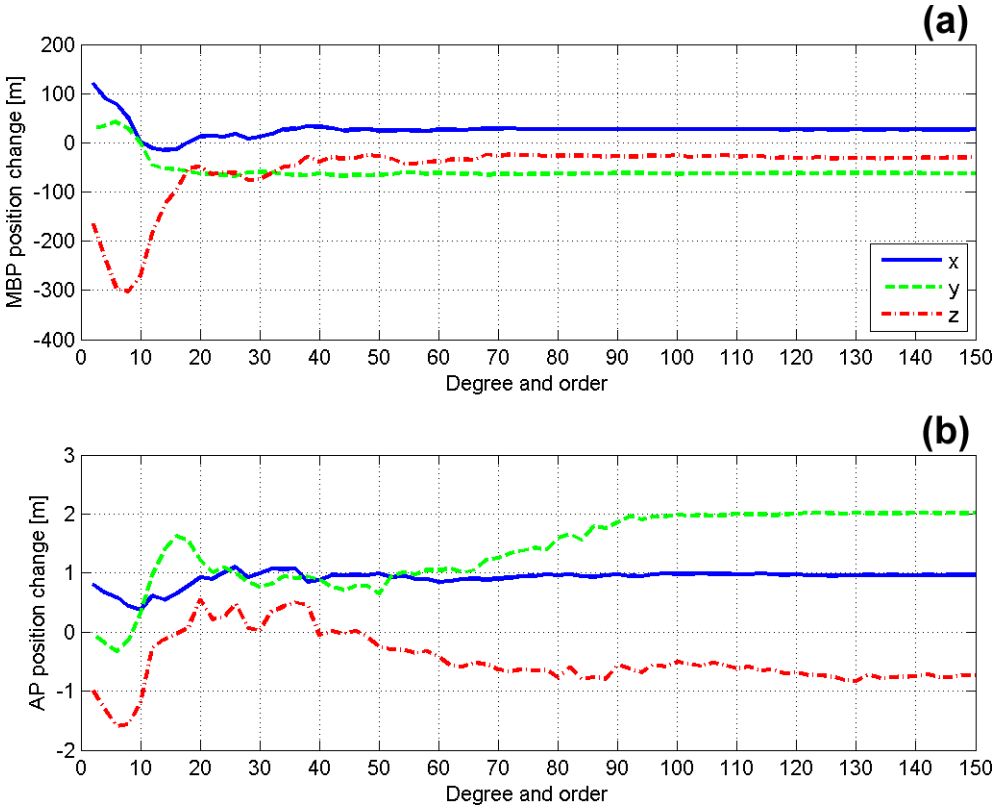


Figure C.2: Landing site 3 (3.63° East, 26.13° North).

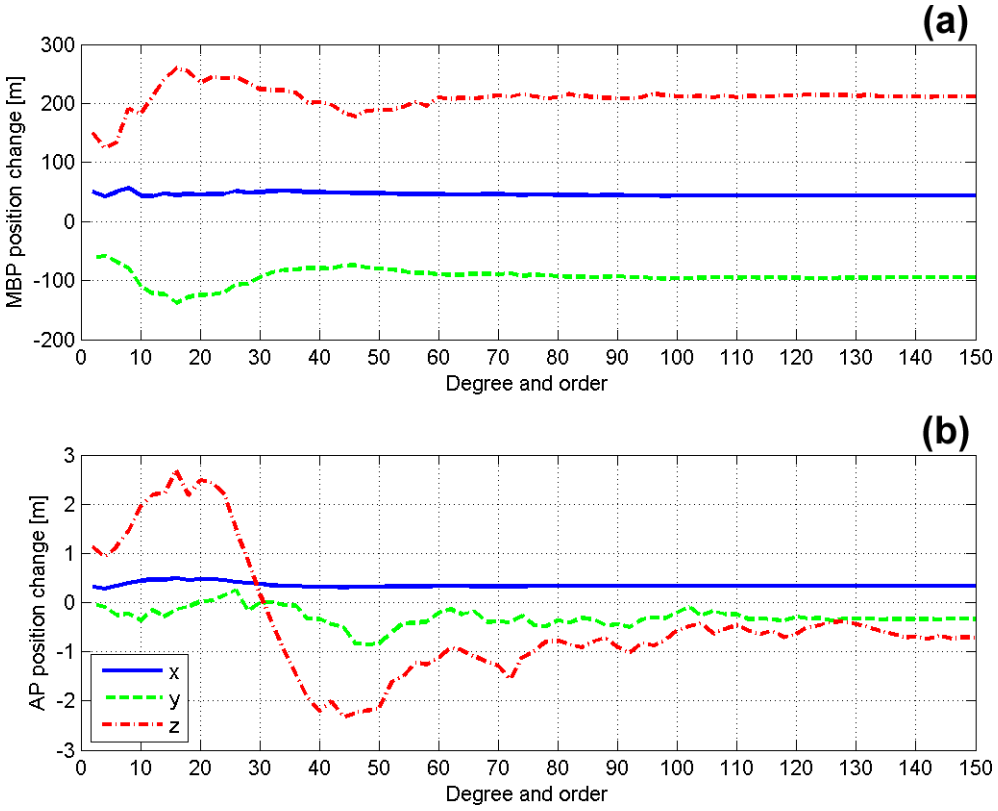


Figure C.3: Landing site 4 (83° East, 58° North).

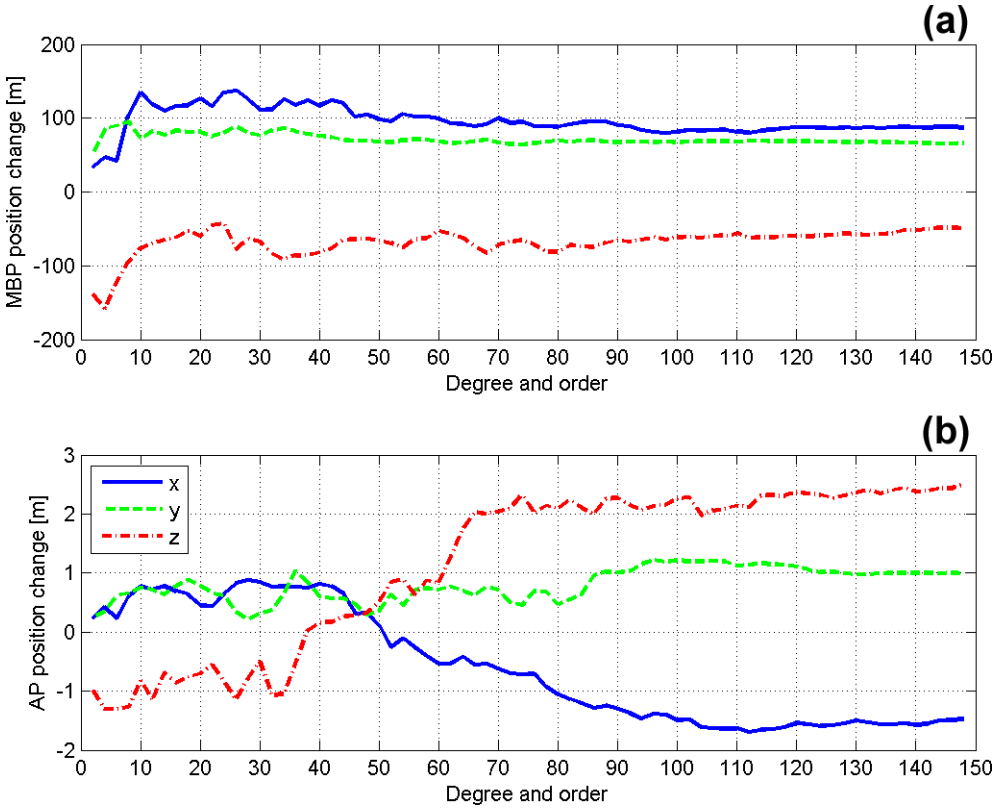


Figure C.4: Landing site 5 (120° East, 7° North).

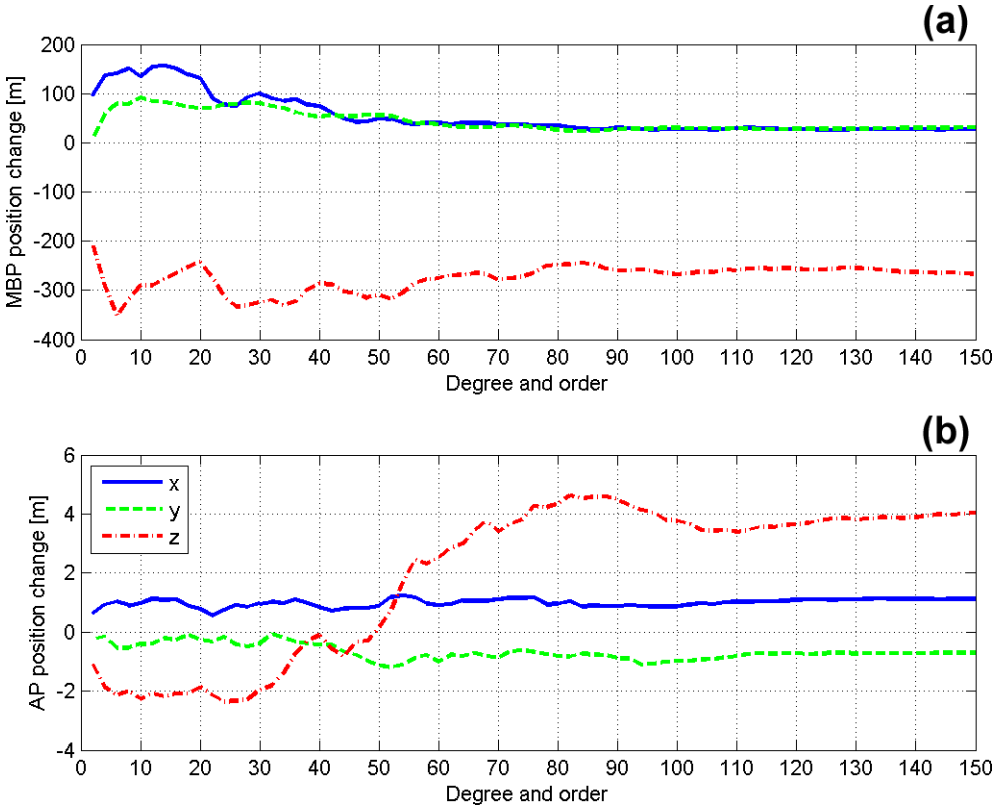


Figure C.5: Landing site 6 (149° West, 19° North).

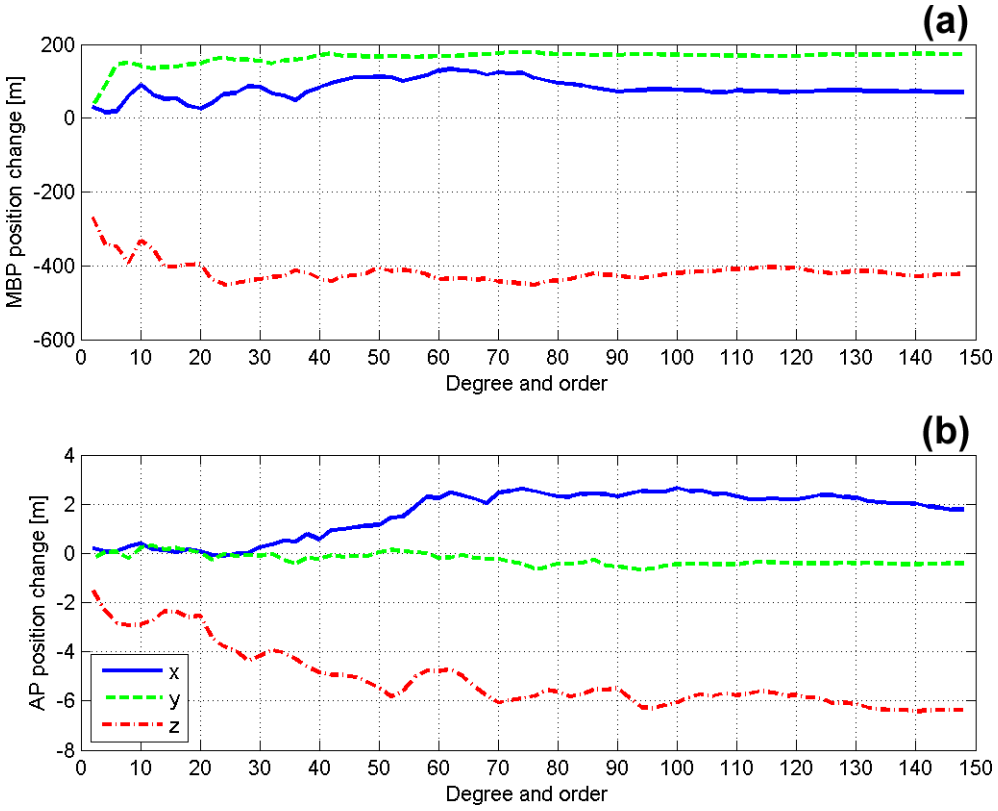


Figure C.6: Landing site 7 (157° West, 5° North).