Ingo Gerth

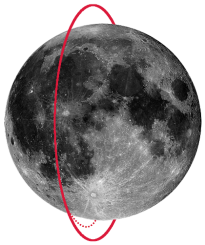
Convex Optimization

For Constrained and Unified Landing Guidance

TU Delft

CONVEX OPTIMIZATION FOR CONSTRAINED AND UNIFIED LANDING GUIDANCE

A Master Thesis



INGO GERTH, BSC
UNDER SUPERVISION OF
Erwin Mooij and Christian Philippe

August 15, 2014

Submitted in partial fulfillment of the requirements for the degree of Master of Science at Delft University of Technology.

This work was partially produced during a three-month internship at ESA ESTEC, Lunar Lander Office D/HSO-IL. The major part was carried out at the Section of Astrodynamics & Space Missions, Faculty of Aerospace Engineering, TU Delft.

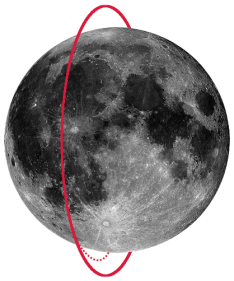
The author was supervised by Erwin Mooij, Assistant Professor/TU Delft, Thesis Supervisor, with additional support by Christian Philippe, Landing System Engineer/ESTEC.

WEBSITES:
http://www.as.lr.tudelft.nl/
http://www.esa.int/
http://www.ingogerth.de/

E-MAIL:
ingo@gerth-ac.de

STUDENT NUMBER:
1544160

TITLE PICTURE:
The ESA Lunar Lander after its successful landing on the surface of the Moon. Artist's conception. Courtesy of ESA.



ABSTRACT

Historically speaking, the safe-landing probability for missions to the Moon and Mars has been unsatisfyingly low. To mitigate the risks of planetary landing, vision-based hazard detection and avoidance and precision-landing systems are currently being developed at institutions worldwide. Not only can these key technologies improve the safe-landing probability overall, but they also bring new mission scenarios within reach. First, scientifically-interesting regions in *a priori* unsafe terrain become accessible. Second, the delivery of vehicles to previously-landed surface-assets will be possible.

An enabling technology for vision-based technologies is highly-constrained, adaptive guidance. Future guidance algorithms need to constrain a lander's flight path such that the field of view of sensors is pointed at the landing-site throughout the flight. They also have to do this in real-time, because new trajectories may need to be generated on-board for diverting to safer landing sites. At the same time, thrust-levels must be constrained to remain within feasible ranges, and sub-surface flight must also be avoided. Previous algorithms can be mostly traced to the Apollo era and do not possess these features.

Another limitation of heritage guidance-algorithms is their limited range of application. Typically, there is a specific algorithm for each flight-phase of a landing. This leads to complex guidance systems.

This thesis successfully tackles the challenge of highly-constrained, rapid trajectory-generation and unifies guidance into a single algorithmic framework. This is achieved by formulating the guidance problem as a convex program, more specifically a second-order cone-problem (SOCP). Convex optimization theory proofs that SOCPs can be solved to find the *global optimum* within polynomial time and are *guaranteed to converge* up to a prescribed level of accuracy. This makes such algorithms promising candidates for on-board implementations.

An algorithm is developed that can take all these constraints into account and that is applicable for both, the main-braking and approach phases. Viewing-angle constraints are enforced by successively converging to strict field of view (FOV) constraints. Spherical gravity is implemented using a linear time-varying system where the magnitude of the gravity vector is obtained through iteration. This is implemented using the CVX modelling language and an embedded conic optimization solver, that has been developed for similar on-board applications.

The developed algorithm is used to generate a Pareto front of optimal trajectories based on a propellant cost-function and a cost criterion for the viewing conditions during the approach. The vehicle baseline for this study is ESA Lunar Lander (LL), which has been studied at the agency up until 2012 and system requirements review maturity. As an exemplary scenario, the Pareto-optimal mid-point is studied in detail. A sensitivity analysis shows that a large range of initial conditions is feasible. This is also demonstrated by a 1000-run Monte Carlo analysis under LL-derived realistic dispersions. Successful trajectory generation can be ensured in 100 % of the cases. The globally optimal solution suffers from merely 188 m/s of gravity losses on a total Δv of 1891 m/s.

PREFACE

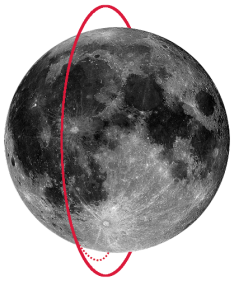
> My friends there were dancing here in the streets of Huntsville when our first satellite orbited the Earth. There was dancing again when the first Americans landed on the Moon. I'd like to ask you: don't hang up your dancing slippers.
>
> *Wernher von Braun*

The moon beckons! It will not take all too long until we will return to the surface of our celestial neighbour. And despite the lessons learned in the forty years since the last Lunar landing, it will be a challenge again. New and exciting technologies are on the horizon, that will enable us to build a new generation of Lunar landers. This thesis aims to contribute to this in the field of landing guidance. It presents an algorithm that has the potential of generating highly-constrained trajectories in real-time. This guidance law is based on the groundbreaking work by Açıkmeşe and Ploen [2007], who were the first to apply convex optimization to the guidance problem. Moreover, the complexity of the guidance system is successfully reduced by consolidating the algorithms of multiple phases into a single, unified framework. To the best of this author's knowledge, this thesis thereby presents the first constrained, consolidated, potentially on-board capable guidance algorithm in the publicly available literature.

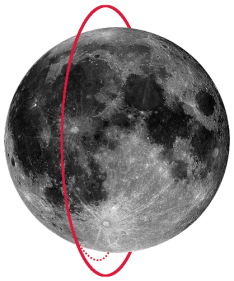
This thesis is submitted in partial fulfillment of the Master of Science degree program at the faculty of Aerospace Engineering, Delft University of Technology. Part of this research has been published at the AIAA GNC conference 2014 under the title "Guidance for Autonomous Landing on Atmosphereless Bodies" in January this year. An extensive literature study has been done in prepartion of this work as part of the course AE4020, that was partly carried out in scope of an internship in the Lunar Lander Office (D-HSO/IL) at ESTEC. This document is available upon request.

I am grateful that I could conduct this extensive 42 credit research project (AE5810) under supervision of my supervisor Dr. Erwin Mooij and with the support from Christian Philippe of ESA/ESTEC. With their experience, I have received all the guidance (pun intended) I needed. Many thanks go to Dr. P.N.A.M. (Pieter) Visser and Dr. H.G. (Dries) Visser, who agreed to take their precious time for joining my graduation committee. I would also like to acknowledge Georg Schulte of Airbus Defense & Space, who kindly provided data on a European apogee motor. Further thanks go to Svenja Woicke, with whom I had plenty of fruitful arguments on HDA systems and planetary landing in general. (Who – of course! – was always right.) Finally, I also want to extend my thanks to my parents, who have supported me throughout all the years in Delft. To all the others who feel left out in this list: thank you very much as well. You know who you are.

I would have had great pleasure in continuing this research and further pushing the boundaries of Lunar landing technology. Alas, it was not meant to be. Nevertheless, my passion for space exploration remains unwavered, and I plan to keep contributing in the future. In writing this thesis I have made an effort in presenting this material in the most readable and exciting way possible—inasmuch as that is possible for such a theoretical topic. I hope that you, the reader, will have enjoyed yourself upon finishing the read. Please, enjoy the tour!



CONTENTS



NOMENCLATURE

ACRONYMS

AG	approach gate
AIAA	American Institute of Aeronautics and Astronautics
AOCS	attitude and orbit control system
ATV	Automated Transfer Vehicle (ESA's re-supply vehicle for the space station)
BC	boundary condition
CFTOC	constrained finite-time optimal-control problem
CITOC	constrained infinite-time optimal-control problem
COM	center of mass
CPU	central processing unit
CSG	Centre Spatial Guyanese (ESA's launch site in Kourou, French Guyana)
D/L	descent and landing
DCM	direction cosine matrix
DEM	digital elevation model
DG	Director General (chief of ESA)
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DOF	degree of freedom
DOI	descent orbit insertion
DPS	descent propulsion system (or Lunar module descent engine, used on LEM and manufactured by TRW)
EAGLE	Entry and Guided Landing Environment (simulator suite)
EAM	European Apogee Motor
ECOS	Embedded Conic Optimization Solver
EDL	entry descent and landing
ENU	east-north-up
EOM	equation of motion
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre (ESA site in Noordwijk, Holland)
FOV	field of view
GNC	guidance, navigation, and control
GNSS	global navigation satellite system
HD	hazard detection
HDA	hazard detection and avoidance
HEO	highly eccentric orbit
HG	high gate
HM	hazard mapping
HSO	Human Spaceflight and Operations (ESA directorate)
IMU	inertial measurement unit

ISS	International Space Station
JPL	Jet Propulsion Laboratory (NASA-financed center of Caltech)
L/S	landing site
LCLF	lunar-centric, lunar-fixed
LEM	Lunar Excursion Module (Apollo Lunar lander)
LG	low gate
LIDAR	light detection and ranging (laser instrument)
LL	Lunar Lander (ESA mission)
LLO	low lunar orbit
LQCIP	longest quasi-continuous illumination-period
LRO	Lunar Reconnaissance Orbiter (NASA mission)
LTI	linear time-invariant system
LTV	linear time-varying system
LV	local-vertical
LVLH	local-vertical, local-horizontal
MC	Monte Carlo (probabilistic analysis)
MSC	Master of Science
MSL	Mars Science Laboratory (NASA mission)
MVM	mission vehicle management
NASA	National Aeronautics and Space Administration
NED	north-east-down
NLP	nonlinear programming
ODE	ordinary differential-equation
PDF	probability density function
PDI	powered descent initiation
PEG	powered explicit guidance
RAAN	right ascension of the ascending node
RCS	reaction control system
RHU	radioisotope heater unit
SOCP	second-order cone-problem
TBC	to be confirmed
TD	touchdown
TG	terminal gate
TPBVP	two-point boundary-value problem

SYMBOLS

▶ Greek letters

α	thrust-to-massflow conversion factor, s/m
β	viewing angle, rad
Γ	convexification slack-variable, N
γ	fight-path angle, rad; sensor angle, rad
δ	latitude, rad
ε	error; tolerance
ζ	constraint limit of viewing angle, rad
θ	pitch angle, rad; glide-slope angle, rad; attitude-cone angle, rad

μ gravitational parameter, m^3/s^2; lumped thrust-constraint
σ mass-normalized convexification slack variable, m/s^2
$\boldsymbol{\tau}$ mass-normalized thrust-command, m/s^2
τ longitude, rad; time-integration dummy-variable, s
Φ transition matrix
φ roll angle, rad; activation-function; golden ratio
ξ obliquity angle, rad
χ heading angle, rad
Ψ discretization matrix
ψ yaw angle, rad

▶ Latin letters

A system matrix
$\boldsymbol{a}$ acceleration, m/s^2
a golden-section left bracket
B input matrix
$\hat{\boldsymbol{b}}$ boresight vector
b golden-section right bracket
c golden-section midpoint
d downrange, m
E selection matrix
$\boldsymbol{e}$ selection vector
e eccentricity
$\boldsymbol{F}$ force vector, N
$\boldsymbol{G}$ stacked gravity vector, m/s^2
$\boldsymbol{g}$ gravity vector, m/s^2; inequality constraint vector
$\boldsymbol{h}$ equality constraint vector
h height (altitude), m
I_{sp} specific impulse, s
$\hat{\boldsymbol{\imath}}$ unit vector in x-direction
J performance index
$\hat{\boldsymbol{\jmath}}$ unit vector in y-direction
$\hat{\boldsymbol{k}}$ unit vector in z-direction
m mass, kg
N number of temporal nodes
$\hat{\boldsymbol{n}}$ nominal-direction vector
$\boldsymbol{p}$ combined control-vector ($\boldsymbol{u}$, $\boldsymbol{\sigma}$), m/s^2
R radius, m
$\boldsymbol{r}$ position vector, m
S stacked system/input matrix
$\boldsymbol{T}$ thrust, N
$\boldsymbol{t}$ frame translation vector, m
t time, s
$\boldsymbol{U}$ stacked control-vector
$\boldsymbol{u}$ control vector
$\boldsymbol{v}$ velocity vector, m/s

$\boldsymbol{X}$ stacked state-vector
x x component (*e. g.*, of position vector)
y y component (*e. g.*, of position vector)
z z component (*e. g.*, of position vector); convexified mass

▶ Calligraphic letters

$\mathcal{D}$ domain
$\mathcal{E}$ orbital energy
$\mathcal{O}$ origin, order
$\mathcal{C}$ convex set

▶ Operators

$\mathsf{I}_{n\times n}$ identity matrix of dimension n
R_x elementary rotation matrix about x axis
R_y^x rotation matrix of frame x w. r. t. frame y

INDICES

▶ Latin letters

$\square_c$ control; continuous time
$\square_d$ discrete time
$\square_e$ exhaust
$\square_f$ final conditions
$\square_g$ gravitational
$\square_{\text{go}}$ …to go (as in t_{go})
$\square_h$ upper constraint boundary
$\square_k$ at time-step k
$\square_l$ lower constraint boundary
$\square_{\max}$ maximum/upper bound
$\square_{\min}$ minimum/lower bound
$\square_r$ radial
$\square_u$ control matrix index
$\square_x$ system matrix index

▶ Frames

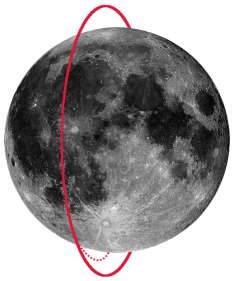
$\square_{\mathcal{A}}$ approach frame
$\square_{\mathcal{F}}$ arbitrary frame
$\square_{\mathcal{I}}$ Lunar-inertial frame
$\square_{\mathcal{L}}$ landing-site frame
$\square_{\mathcal{V}}$ local-vertical, local-horizontal frame

▶ Numbers

$\square_0$ initial conditions; at zero epoch

▶ Symbols

$\square_{\leftmoon}$ Moon



Chapter 1

INTRODUCTION

"Houston, tranquility base here. *The Eagle has landed.*" These were the first words that Edwin "Buzz" Aldrin radioed back to Houston, TX, after touching down as the first humans on the Moon on July 20, 1969. The reply from Charles Duke in mission control was: "Roger, Tranquility. We copy you on the ground. *You got a bunch of guys about to turn blue.* We're breathing again. Thanks a lot." But what had happened? Why did people nearly turn blue?

The answer lies a few hundred meters away from the landing site, on the surface of the Moon, at the crater "Big West" (see Figure 1.1). The automatic landing system of the Lunar Module, that was capable of landing the spacecraft autonomously on the Moon, was targeting the large crater and its surrounding boulder field for touchdown. When the pilot Neil Armstrong noticed this, he took over controls and steered the lander manually away from this very hazardous region, that could have compromised mission success and might have stranded the astronauts on the Moon. Although mission control was not aware of this, they realized that the landing took much longer than planned—and that Armstrong finally landed with merely 20 s of propellant remaining, a small margin. The early Lunar landings were clearly more risky undertakings than is commonly thought [Heppenheimer 1999].

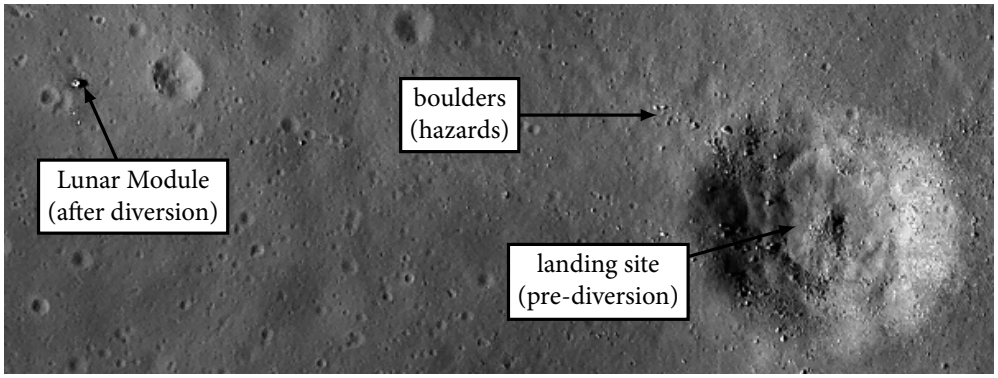


FIGURE 1.1 The Apollo 11 landing-site as seen by NASA's Lunar Reconnaissance Orbiter. Had Armstrong not manually steered the Lunar Module to a safe spot, it would have landed in a boulder-filled, hazardous crater. Courtesy: NASA/GSFC/ASU/GoneToPlaid

In hindsight, Apollo 11 was saved only for one reason: the pilot on board who could see the hazards and fly past them. Later robotic landers that attempted landings on the Moon, Mars, and other celestial bodies could not rely on any such input. They could also not be piloted remotely, due to the far distances between landers and the mission operators back on Earth. This very risky nature of planetary landings has been proven over and over again ever since, and reflects in the high loss rates of such vehicles: by the beginning of the last decade, the historical probability of mission failure had been more than 20 % (although not all losses can be traced to hazards) [Strandmoe et al. 1999].

For future missions such a high probability of failure is deemed unacceptable. Fortunately, there have recently been significant technological advances in making planetary landing safer. So-called hazard detection and avoidance (HDA) and precision-landing systems harness developments in vision-based technologies and faster on-board computers. Such systems are able to detect hazards and command diversions to safe sites, just as Armstrong had done in Apollo 11. The applications of HDA and precision landing do not stop there, however. They also open up the possibilities for new missions. For example, a reliable HDA system might have been able to safely land in the generally dangerous "Big West" crater, which would have been much more interesting geologically speaking than the flat landing site where the Eagle eventually landed. Moreover, precision landing can enable scenarios where it is required to land a spacecraft precisely in the vicinity of a previously landed assets. An example for such a mission would be a re-supply vehicle for a Moon base.

One of the key technologies that is required to successfully realize such systems is robust and optimal *guidance*. Guidance takes up the task of generating the lander's landing trajectory on board of the spacecraft, much in the same way as Armstrong steered the lander safely to the surface but, in an automated fashion. The new vision-based systems impose new constraints on guidance: first and foremost, the landing site has to remain in the field of view (FOV) of the sensors for a major part of the flight. Guidance must thus generate trajectories that satisfy this condition. Moreover, it must do so in a propellant-optimal manner. The example of Apollo 11 underlines how critical propellant is to a landing. Finally, diversions to safer landing sites require guidance to re-generate new trajectories in real-time. The computational efficiency of algorithms is therefore of utmost importance.

In the past, many missions and studies have relied on the guidance algorithms developed in the Apollo era. These are comparably simple closed-form expressions. While they have been proven to work, it is also known that they are sub-optimal in terms of propellant and cannot take into account any constraints [Rea 2009]. They are thus less suitable for next-generation landers. A further issue is that there are typically a number of different guidance algorithms for different phases. It would be desirable to reduce the number of algorithms for decreasing guidance complexity.

A very promising guidance algorithm has recently been proposed by Açıkmeşe and Ploen [2007]. This is based on a special field within optimization, that deals with finding the optima of *convex* problems. Specifically, the guidance problem is formulated as a second-order cone-problem (SOCP). As it turns out, it can be proven that convex optimization-problems are *guaranteed* to find the *global* optimum of a problem up to specified precision and extremely quickly in polynomial time. These properties make SOCP-based algorithms very promising for real-time on-board applications. As a matter of fact, the National Aeronautics and Space Administration (NASA) is currently testing the work by Açıkmeşe and Ploen [2007] on terrestrial rocket-powered landing-demonstrators [Williams 2013]. The special features make convex optimization stand out among other numerical methods, such as nonlinear programming (NLP), as the latter is inefficient and cannot guarantee global optima and convergence.

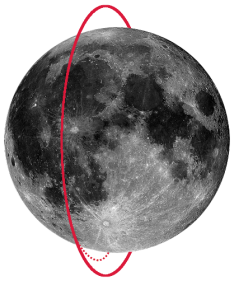
Up until now, the application of "convex guidance" has been restricted to Mars-landing scenarios. It is the goal of this thesis to derive several major extensions to the algorithm by Açıkmeşe and Ploen [2007], that will make it applicable to Lunar missions.

First, a viewing-angle constraint is added that tailors trajectory generation for optimized landing-site visibility. Second, the very basic dynamical models will be improved by extending the algorithm to take spherical gravity into account. Finally, this allows to apply the developed guidance algorithm for the entire descent and landing (D/L) maneuver of lunar landing, from main braking up to terminal gate (TG). This consolidation is a major innovation compared to previous guidance systems, which applied up to three or more different algorithms throughout the flight. As a result, the overall system complexity and risk can be reduced. This thesis focuses on the theoretical development of the algorithm and presents an exemplary application inspired by the the European Space Agency (ESA) Lunar Lander (LL) scenario, a mission that has been under study at the agency up until its cancellation in 2012. The thesis does not intend to present a full mission-analysis design-cycle for the landing scenario. The research question may be formulated as follows:

> *Can convex optimization theory be used to derive a guidance algorithm that is globally propellant-optimal, can constrain the viewing conditions in the approach phase, and unify main braking and approach-phase guidance into a single algorithm?*

To the best of this author's knowledge, this thesis presents the first detailed treatment of a highly-constrained convex-programming-based guidance-algorithm with appropriate gravity models for Lunar flight.

The report is organized as follows. Chapter 2 introduces the reference mission and gives some insight into Lunar landing in general by studying important heritage missions. To establish the necessary mathematical background to describe motion, elementary concepts in flight mechanics are introduced in Chapter 3. Building on these physical concepts, guidance and its place within the guidance, navigation, and control (GNC) systems of planetary landers is explained in Chapter 4. This is followed by Chapter 5, which discusses the theoretical background in optimization theory needed to understand convex guidance. The fundamental ideas of the latter are the subject of Chapter 6. Extensions to convex guidance, the main contributions of this thesis, are proposed in Chapter 7. The implementation of convex guidance in a simulation environment is then treated in Chapter 8. This software tool is applied in Chapter 9 for the design of an optimal scenario. The thesis is concluded with Chapter 10, which also gives some recommendations for future work.



Chapter 2

REFERENCE MISSION AND HERITAGE

> "I am sure humans will return to the Moon; but they will go to the Moon together and not in the context of two competing countries."
>
> *Jean-Jacques Dordain (ESA's Director General, 2009)*

> "The problem with the Lunar Lander is that we have not found enough money to do it."
>
> *Johann-Dietrich Wörner (Chairman of DLR, 2012)*

Although the Moon has been frequently visited by spacecraft in the past, there are still many reasons to return. Amongst all other major space agencies, ESA had the goal of sending a Lunar lander to the South Pole in 2019, see Figure 2.1. Unfortunately, these efforts were cancelled due to lack of funding in the optional program at the council on ministerial level in 2012. Even though this plan has been shelved for now, the ambition to return to the Moon in the future is alive within ESA and other agencies, and the development of landing systems is now continuing in cooperation with Роскосмос (the Russian space agency Roscosmos).

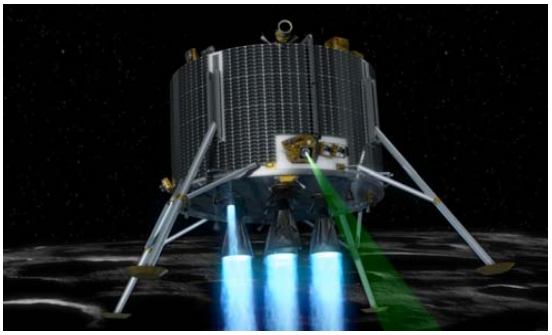
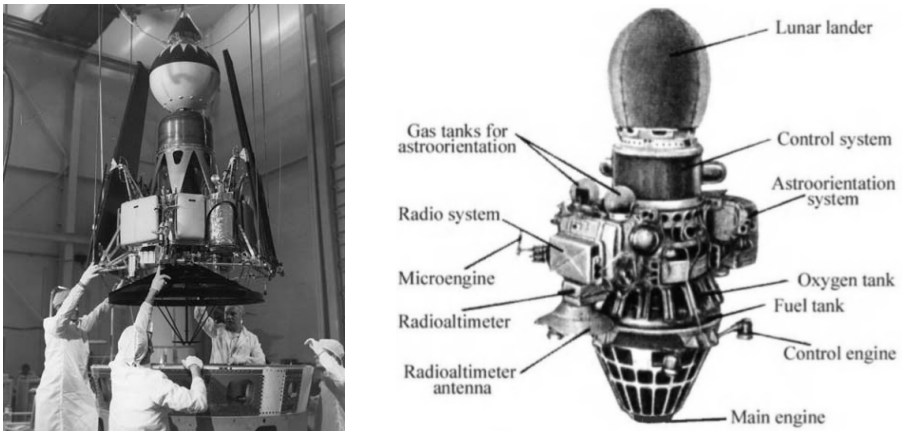


FIGURE 2.1 The Lunar Lander during its final descent phase toward the South Pole, using instruments for hazard detection and avoidance. Source: [Fisackerly et al. 2011, p. 2]

Having been developed up to an equivalent of Phase-B1 (*i. e.*, ending with a Pre System Requirements Review), the ESA LL has reached a high level of maturity and will serve as the reference vehicle for this study. This chapter will set the scene for future discussions on guidance, by explaining the general context. The following chapters will build on this, and systematically derive a new concept for landing guidance.

An extensive study of all previously Lunar landing missions, the scientific rationale for returning to the Moon, and a detailed description of the ESA LL mission has been previously [Gerth 2013]. The scope of this thesis does not suit presenting this material again, which is why this chapter is an abridged version of the former. As such, the content is laid out as follows. Section 2.1 gives an overview of previous Lunar landers. The ESA LL mission is described in Section 2.2.



(a) Ranger Block II **(b)** Luna 9

FIGURE 2.2 First-generation Lunar landers with rough-landing capsules. Sources — Left: NASA; Right: [Harvey 2006, p. 121]

2.1 MISSION HERITAGE

The twentieth century has seen a staggering number of Lunar landing attempts, many of which were successful. In the heat of the space race from 1962 to 1976, a total of 32 landers were sent to the Moon, out of which 18 have been successful [Ball et al. 2007]. It is worthwhile to have a look at what has been done in the past to understand what is needed in the future. For this reason, a brief overview of heritage missions is given in this section.

2.1.1 An overview of Lunar landers

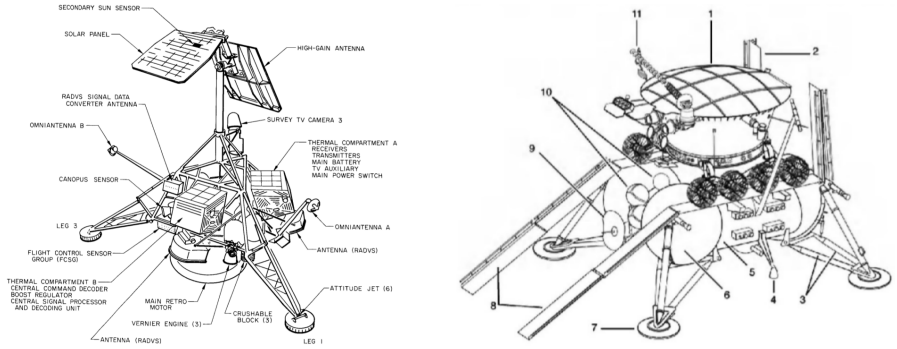
The early development of Lunar landers has seen a rapid evolution. Within the short time-span of just seven years, from the first landing attempt in 1962 up to the first human landing in 1969, three phases can be identified:

2.1.1.1 *Phase 1 — Early rough landings*

The earliest missions did not have any major goals, save for one: *to be the first to land on the Moon*. These landers were consequently rushed designs and primitive compared to later efforts. The first Lunar landing attempts were undertaken by NASA's Ranger Block II Seismo capsules (Ranger 3-5), which were balsa-wood capsules that were dropped onto the Moon from 300 m altitude, see Figure 2.2(a) [Hall 1977]. All landing attempts failed.

After six tries with a design that used airbags to protect the spacecraft at impact, the Soviets were the first to successfully land on another celestial body with Luna 9 on February 3, 1966 [Harvey 2006]. This spacecraft is shown in Figure 2.2(b).

Both families of missions differ severely from the designs of today's landers. Being very primitively guided landings, they will thus not be discussed any further.



(a) Surveyor **(b)** Ye-8 with rover

FIGURE 2.3 Second-generation Lunar landers with legged platforms. Sources — Left: NASA; Right: [Huntress and Marov 2011, p. 190]

2.1.1.2 *Phase 2 — Advent of legged landers*

Success rates of Lunar landings surged when legged landers were developed. The first successful American landing took place on May 30, 1966 with Surveyor 1 [NASA 1966, p. 23]. The Soviets also upgraded their landing platform and deployed a legged platform on the Moon with Luna 15 (Ye-8 series) on July 13, 1969 (just three days before the first human landing) [Huntress and Marov 2011, p. 190]. Both spacecraft are shown in Figure 2.3. The rover was successfully deployed.

This second generation of landers was highly influential to future missions and also applied novel GNC concepts. The Surveyor landers were the first spacecraft to fly gravity-turn landing-maneuvers, for instance (see Section 4.4) [Cheng 1964]. However, the following class of landers is even more important, and a discussion of the Surveyor missions is thus spared.

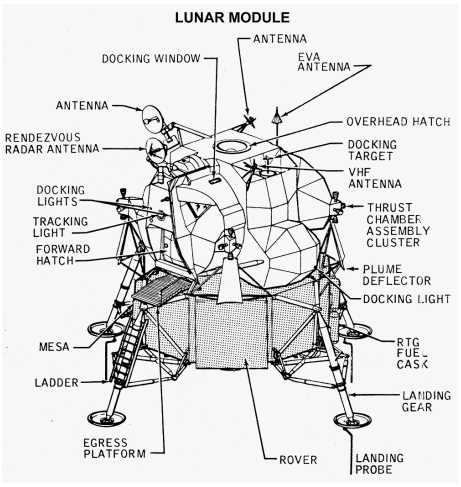
2.1.1.3 *Phase 3 — Human missions*

By far the most important landing mission of all times is Apollo 11—not only from a historical point-of-view, but also under technological considerations. The mission touched down on July 16, 1969. Because it is the most important reference, it will be discussed in more detail in the following section.

On a side note, the Soviets also had a Moon program for manned flights. In scope of this, the engineers managed to send animals on a free-return trajectory around the Moon and return them safely, and even performed initial tests on the intended landing platform in low Earth orbit. However, the program was eventually doomed by a string of failures of the intended Moon rocket, the N-1 [Ball et al. 2007; Johnson 1995].

2.1.2 Apollo – Crewed Lunar landings

Being the most important reference of heritage vehicles, the technical features of the Apollo Lunar Excursion Module (LEM) will be detailed here. It is shown in Figure 2.4. The system's characteristics are listed in Table 2.1.



(a) Technical features of the LEM.

(b) The LEM, Lunar rover, and astronaut James Irwin of the Apollo 15 mission.

FIGURE 2.4 The Apollo Lunar Excursion Module. Courtesy: NASA

TABLE 2.1 Overview of Apollo 11 technical data. Only listing landing-GNC relevant systems. Sources: [Orloff 2001], [Grumman 1971] and those listed in the table.

DELIVERY	Lunar transfer to 122.4 km × 101 km LLO
LANDING	• Start from LLO to 108 km × 14.5 km transfer orbit; • Then PDI at 1.7 km/s and 482 km downrange for main braking at maximum thrust; • Guidance using explicit law and variable thrust (see below) enabled during last two minutes of main braking to correct for errors; Subsequent approach phase starting at 8.3 km downrange using same guidance, but landing site as target, enabling astronaut pilot vision of landing site; • Final landing guidance phase starts at 150 m altitude (low gate) with same law; • At 10 s from target horizontal velocity nulling law is engaged; • Finally, a vertical descent law is called and executed until touchdown; • Manual pilot override possible during entire approach phase; • Very detailed data in: [Orloff 2001] and [Bennett 1970]
GNC HARDWARE	Guidance computer, inertial measurement unit (IMU) (with accelerometers and gyros); Window; Sextant and telescope for eye-ball pilot navigation; Landing radar; Rendezvous radar; Four RCS blocks with four 445 N Marquardt R-4D thrusters at I_{sp} = 312 s each
MAIN ENGINE	TRW DPS; Nominal thrust: 43 903 N (92.5 %); Full thrust: 45 040 N; Throttleable between 4671 N to 30 248 N (10 % to 65 %); cannot operate between 65 % to 92.5 % as this will lead to engine failure; I_{sp} = 311 s; restartable; gimballed with ±6° motion capability
LANDER MASS	For LEM at separation from command and service module: 15 278.6 kg; at DOI: 15 272.3 kg; at end of DOI: 15 150.7 kg; at landing: 7327 kg
GUIDANCE	Explicit guidance (see Section 4.4, [Klumpp 1971])
CONTROL	Time-optimal controller [Widnall n.d.]

It is safe to say that Apollo has been the most influential program for planetary landers to date. Much of the mission baseline of the ESA LL and other landers is based on the Apollo concept. Starting with the overall landing sequence, Apollo introduced the parking orbit around the Moon to the mission plan. Previous landers would directly descent from the Earth-Moon transfer-trajectory, which was a less robust approach. The ESA LL will also first stay in such a low lunar orbit (LLO).

The landing sequence from LLO was divided into several phases. The lander would first perform a descent orbit insertion (DOI) burn, that would place it in a Hohmann transfer-orbit with a perilune close to the landing-site. From there, the lander would first eliminate most of its velocity in a main braking phase, starting at powered descent initiation (PDI). When most of the speed was cancelled, an approach phase that served the purpose of precisely targeting a landing site was initiated at approach gate (AG). It is exactly this sequence that is used by most Lunar missions today, for reasons of optimality and the scenario's robustness. For details, see Table 2.1.

In scope of the development of this scenario, a number of guidance laws have been developed as well. The most important one has come to be known simply as "Apollo guidance" [Cherry 1964; Klumpp 1971]. Up to this day, this guidance law is being used and continues to be influential. Just as recently as 2012, the Mars Science Laboratory (MSL) mission has applied this guidance law in its terminal-descent phase [Wong et al. 2006].

Some other distinctive features of Apollo deserve a brief discussion. Most obviously, the LEM was a crewed vehicle. Despite this, the GNC system was designed for *autonomous* landings, where only little data would need to be relayed from Earth or entered by the astronauts. This makes it an applicable reference even for today's robotic landers. In practice, however, no entirely autonomous landing sequence has been executed throughout the Apollo program. In all landings, pilot intervention occurred during the approach phase, especially during the final landing phase [NASA 1969]. This was done to correct for targeting errors of the landing system: with the 1960s hardware, the navigation capability was very limited and led to large navigation inaccuracies in the order of kilometers. The automatic guidance system would have steered the lander to wrong locations. The pilots corrected these errors with "vision-based navigation" *(aka: looking out of the window)*. This could be done either by entering new landing coordinates into the automatic guidance computer, or steering the spacecraft directly and manually with a joystick [Klumpp 1971]. In Apollo 11 the navigation errors were unexpectedly large. Therefore, in later missions the spacecraft was tracked in orbit up until the braking phase leading up to touchdown. The astronauts would enter the final state estimate uplinked from the ground about a minute before the braking maneuver. This approach is similar to the absolute vision-based navigation employed by the ESA LL, see Section 4.5.1.

Sometimes outshined by its success, the Apollo landings were actually perilous and errors bugged all missions. For example, in Apollo 11 the guidance computer crashed due to a data overflow, because the rendezvous radar was not shut down before the approach [NASA 1969, p. 4-9]. In fact, the list of LEM anomalies from the latter document counts 14 items. The spacecraft was also approaching an area scattered by boulders, which could have damaged the LEM (see Chapter 1). More anomalies occurred in later missions. Amongst others, a faulty landing-abort error occurred that would have aborted the automatic landing sequence had the pilot not overruled it (Apollo

14, [NASA 1971, p. 14-28]), and the LEM temporarily lost its attitude-control capabilities (Apollo 16, [NASA 1972, p. 14-60]). Severe landing hazards were also found in the terrain near all Apollo landing sites post-touchdown. For example, Apollo 15 nearly tipped over as it landed with one leg in a small crater [Paschall et al. 2009].

This shall not depreciate the success of the Moon landings. It is a feat that the complex vehicle could be handled to overcome these problems, and a testament to the designers. Mission success was reached, after all. But although it is speculative, it has to be mentioned that these problems could have caused mission losses if the pilots would not have intervened. For today's missions, and in plans and requirements for the ESA LL, a higher reliability must be achieved. Landings must be carried out entirely autonomously, without any human or ground intervention. It is a lesson learned that every contingency that can be thought of has to be taken into account—which also applies to the guidance and control-system design. For these reasons, the ESA LL was supposed to carry an HDA system, and developments of such systems continue even in spite of the lander's cancellation. It is thus one of the requirements for the guidance algorithm developed in this thesis to be compatible with HDA.

As touched upon above, the pilots could steer the spacecraft and correct the flight-path manually. Using this kind of visual navigation, the astronauts could achieve impressively precise landings. The need for precision landing is of great importance again today, and was one of the main features to be demonstrated by the ESA LL in fact. Having set an appropriate context, the LL's mission is now detailed in the following section.

2.2 REFERENCE SCENARIO

The mission statement for the ESA Lunar Lander was formulated as follows [Fisackerly et al. 2011, p. 1]:

> *The European Lunar Lander mission, targeted for launch in 2018 and a landing near the Moon's South Pole, shall demonstrate critical technologies associated with planetary landing and shall prove Europe's ability to land safely and precisely.*

Phase B1 of the mission was managed by the Lunar Lander Office within ESA's Human Spaceflight and Operations (HSO) directorate (ESA – D-HSO/IL) and was developed by EADS ASTRIUM SPACE TRANSPORTATION GMBH of Bremen as prime contractor.[1] The small payload of the lander was to be used especially to prepare future human exploration [Carpenter et al. 2010]. This would have included, for example, characterizing the surface dust and the radiation environment. This is also the reason why the lander was developed under D-HSO—it was originally envisioned to support human surface-operations by acting as a cargo lander, much like the Automated Transfer Vehicle (ATV) is serving the International Space Station (ISS) today.

The lander was planned to be launched by a Soyuz 2-1b launch vehicle from Centre Spatial Guyanese (CSG) in Korou. After insertion into a highly eccentric orbit (HEO), the apogee of this orbit would have been raised in intermediate steps until the lander would have been inserted into a circular 100 km LLO, see Figure 2.5.

1 Now AIRBUS DEFENSE & SPACE.

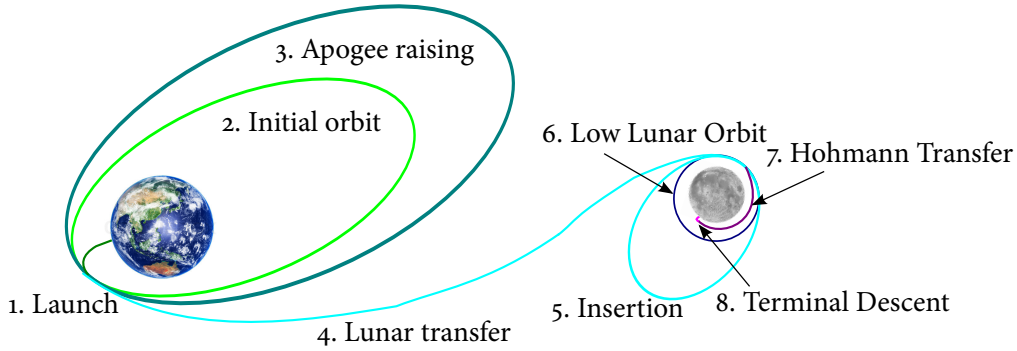


FIGURE 2.5 Tentative notional Earth-Moon transfer. Not to scale.

TABLE 2.2 Some design features of the Lunar Lander. Sources: [Diedrich 2011; Schoenemaekers 2011]

FEATURE	PROPERTY
Launch mass	2202 kg
Wet mass at HG	854 kg
Dry mass	752 kg
Propulsion	Five 500 N EAMS ($I_{sp} \geq 323$ s), six 220 N ATVS-derived thrusters ($I_{sp} = 280$ s)
AOCS	Three-axis stabilized, BBQ-mode during cruise
Power	Solar cells
Thermal control	Passive, no radioisotope heater units (RHUS)
Landing system	Four landing legs, LIDAR for HDA and camera for vision-based navigation.

The right constellation of orbit and landing site was to be achieved by waiting until the Moon had rotated appropriately below the inertial orbit. Once the LLO was aligned properly with the landing site, the landing sequence would have started with a de-orbit burn, which lowers the perilune with a Hohmann transfer down to about 10 km. Reaching this point marks PDI, in which the lander eliminates most of its velocity by firing at full thrust. The end of this maneuver leaves the lander remaining with about 100 m/s velocity at an altitude of roughly 2 km (high gate). From there, the approach phase would have started.

The spacecraft design had several key features, which are summarized in Table 2.2. A render of the lander with dimensions is shown in Figure 2.6. Solar power is used, because of the lack of nuclear power and heat sources in Europe. Because no throttleable engine is available in Europe either, the lander would have been equipped with five 500 N engines and using an additional set of six 220 N engines as support and limited thrust modulation. The propulsion system is also the reason for the large size of the lander: most of the vehicle's interior is occupied by propellant tanks. A suite of sensors for hazard detection (HD) and navigation would have been on board, including a laser imaging, detection, and ranging (LIDAR) system, an altimeter (distance-to-ground sensor), a camera, and an IMU. For more detailed information, see [Fisackerly et al. 2011].

For all intents and purposes, the ESA LL serves as a good baseline vehicle for this study. It has been studied up to significant maturity and its scenario is thus a good reference for this work. However, one major modification is required to allow for the

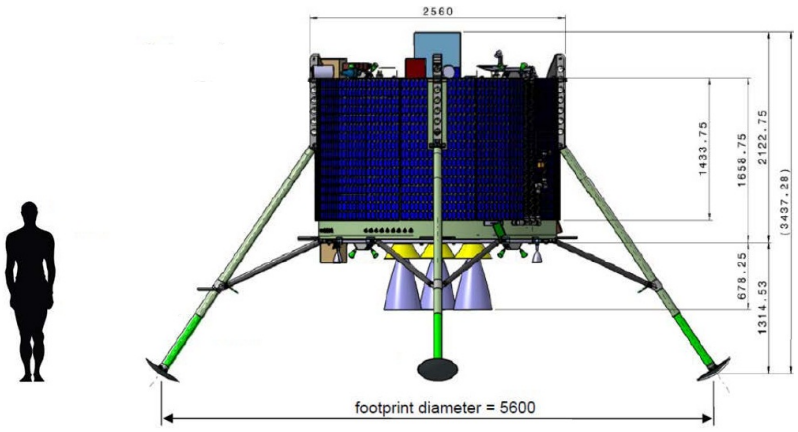


FIGURE 2.6 The baseline spacecraft design of the Lunar Lander. Source: [Carpenter et al. 2012, courtesy of Astrium]

use of convex guidance. In the sequel it will become clear that convex guidance yields bang-bang control solutions in terms of the thrust. This is incompatible with the LL's sequential engine-shutdown strategy using fixed-thrust and pulseable engines. This demonstrates how closely vehicle and guidance/trajectory design are coupled.

In an effort to keep the study as realistic as possible, the main engine's manufacturer AIRBUS DEFENSE & SPACE has been contacted. It turns out that the predecessor of the (as of yet unflown) 500 N European Apogee Motor, the 400 N apogee motor, has been qualified up to a pulse capability emulating a minimum thrust of 83.3 N on the test stand [Schulte 2004]. This has not been taken into account in the ESA LL design, which only performed breadboarding tests with pulseable 220 N engines and constant-thrust apogee motors. Therefore, in this thesis the following thrust levels will be assumed:

$$T_{\min} = 5 \cdot 83.3\,\mathrm{N} + 6 \cdot 0\,\mathrm{N} = 416.5\,\mathrm{N} \quad [2.1]$$

$$T_{\max} = 5 \cdot 500\,\mathrm{N} + 6 \cdot 220\,\mathrm{N} = 3820\,\mathrm{N} \quad [2.2]$$

which assumes hot-restart ability of the 220 N engines. Note that to leave a margin, the following upper and lower limits will be assumed in the guidance algorithm:

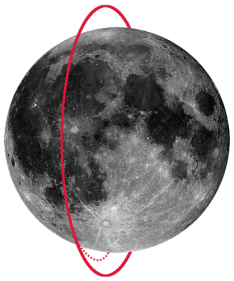
$$T_l = 1.05 T_{\min} = 437.3\,\mathrm{N} \quad [2.3]$$

$$T_h = 0.95 T_{\max} = 3629\,\mathrm{N} \quad [2.4]$$

Such an assumption has also been made in the ESA LL study to account for possible dispersions and leave margins for trajectory-tracking controllers [Soppa 2011].

Chapter 3

ELEMENTARY FLIGHT MECHANICS



> "Everything in space obeys the laws of physics. If you know these laws, and obey them, space will treat you kindly."
>
> *Wernher von Braun*

The landing problem can be traced to one of the most fundamental problems in physics: the motion of a body under influence of external forces. In this case, the body is the LL, and the external forces acting on it are the thrust produced by its engine, the gravity of the Moon, and other "disturbing" forces.

Landing guidance is directly related to the motion of the lander. In fact, it is the purpose of guidance to harness the dynamic capabilities of the system to steer it correctly to the landing site. Most algorithms are based directly on the landing system's dynamic equations. The analysis of the lander's motion is also particularly relevant to this report because the equations of motion (EOMs) are required for simulating the vehicle's flight path. Thus, a proper understanding of flight mechanics is key.

To this end, the chapter is organized as follows. The necessary reference frames to describe motion are introduced in Section 3.1. State representations that can express this motion are treated in Section 3.2. This is followed by Section 3.3, that gives the necessary relations to convert between the different frames. The EOMs needed to model the motion are derived in Section 3.4. Finally, the forces that cause the lander to move are described in Section 3.5.

Note that this chapter is an abridged version of a very extensive literature study conducted in preparation for this thesis. Only material directly relevant to the research is presented in this chapter. The literature study presents some of these topics in more detail and elaborates on many subjects beyond what is needed here. It can be provided upon request [Gerth 2013].

3.1 FRAMES OF REFERENCE

Motion and position can only be described if one has a reference with respect to which the object under study is moving. This chapter therefore introduces what reference frames are exactly, and which frames are needed for this research.

3.1.1 Definition

A reference frame can be defined as follows:

Reference frame A set of axes that serve as reference for a coordinate system, which are aligned to a defined origin and orientation.

A reference frame is thus a set of axes that is attached in a certain orientation to a certain origin. An example of a reference frame is given in Figure 3.1.

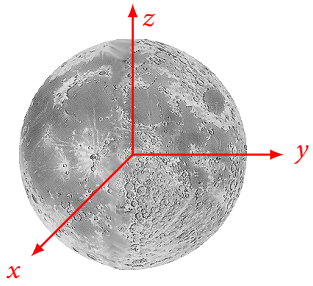


FIGURE 3.1 Example of a reference frame definition: the coordinate axes are attached to the center of the Moon and its directions are defined by the z and x vectors, which point to the north pole and a reference surface feature respectively. y completes the right-handed triad.

In this report, all coordinate systems will be right-handed, and defined by three mutually orthogonal unit-vectors. These are written as follows:

$\hat{\boldsymbol{\imath}}_{\mathcal{F}}$: defines the x-axis
$\hat{\boldsymbol{\jmath}}_{\mathcal{F}}$: defines the y-axis
$\hat{\boldsymbol{k}}_{\mathcal{F}}$: defines the z-axis

$$\hat{\boldsymbol{k}} \quad \perp \quad \perp \quad \hat{\boldsymbol{\imath}} \quad \perp \quad \hat{\boldsymbol{\jmath}}$$

where $\mathcal{F}$ denotes the corresponding frame. Note that two vectors are sufficient for a definition, as the remaining one can be obtained with the cross product of the other two.

There are different frames for different purposes. *Intertial* frames are those that are either at rest or moving at constant velocity, and do so in a time-independent manner. Newton's laws of motion can be directly applied in inertial frames.[1] *Body-fixed* frames can be used to describe attitudes of objects with respect to each other. Finally, *rotating* frames primarily suit the purpose of expressing motion with respect to a central body that is moving in inertial space.

3.1.2 Assumptions

An important assumption in this thesis is that *all motion is treated inertially.* In other words, even though the Moon is moving around the Earth and is rotating, this is neglected. This is a common assumption in the practice of designing guidance algorithms [Açıkmeşe and Ploen 2007]. Only in later stages, when more advanced simulations are necessary, is the motion of celestial bodies considered. This relieves the programming and analysis overhead and allows to focus more on guidance-algorithm development. The error made is negligible in this development phase.[2] Consequently, no rotating frames are needed in this thesis work.[3]

1 In reality, inertial frames do not precisely exist (to the best of our knowledge). That is why such frames ought to be called "quasi-" or "pseudo-inertial" in fact, but this bulky phrase is omitted here for convenience. For virtually all purposes frames that are nearly inertial serve very well and little error is made by using them [Mooij 1998].

2 In practice, there are two methods of accounting for non-inertial motion: i) using a trajectory-tracking controller to compensate; and ii) including the EOMs in the rotating frame directly in the guidance algorithm. The latter is easily possible for the convex-guidance law developed in this report and proposed as a future improvement.

3 Note that because the attitude is determined by the thrust vector in this report, strictly speaking an instantaneously-rotating body-fixed frame is used. However, it does not appear as such explicitly in the text.

Assumption 1: resting Moon The Moon is assumed to be at rest, so that any frame attached to its center of mass (COM) is inertial.

A second assumption is that only translational motion is considered, and thus the three rotational degrees of freedom (DOFS) are ignored. This is also a common assumption in the development of guidance laws, and, in fact, in all of trajectory optimization. The main reason for this is that the rotational dynamics are much faster, and may be considered to be decoupled from the translational dynamics [Açıkmeşe and Ploen 2007]. This assumption translates to the notion of "perfect control", because it is assumed that the state commanded by guidance is realized instantaneously. As a result, no body-fixed frames appear in this report.

Assumption 2: perfect control Rotational DOFS are ignored.

In conclusion, only inertial frames to describe translational motion will be applied. The following sections present all reference frames used in this work. These frames have been self-defined, but the references by Mooij [1998]; Mulder et al. [2011] have been very helpful in the process.

3.1.3 Lunar inertial frame, index $\mathcal{I}$

The Lunar inertial frame, indexed by $\mathcal{I}$ for inertial, is the primary frame used in this work. It is non-rotating and has its origin $\mathcal{O}(\mathcal{I})$ at the Moon's COM. Because the Moon is considered to be at rest, the $\mathcal{I}$ frame coincides with the lunar-centric, lunar-fixed (LCLF) at any time. This makes it very convenient to specify landing sites, as these will not move through inertial space.

The axes are aligned as follows:

$\hat{\boldsymbol{\imath}}_{\mathcal{I}}$: Through the Moon's prime meridian; in the equatorial plane.
$\hat{\boldsymbol{\jmath}}_{\mathcal{I}}$: Completes the right-handed triad.
$\hat{\boldsymbol{k}}_{\mathcal{I}}$: Points through the geometric north pole.

3.1.4 Landing site frame, index $\mathcal{L}$

The $\mathcal{L}$-frame originates from the currently targeted landing site, and is thus located on the surface of the central body. The frame is chosen according to geodetic convention as an east-north-up (ENU) frame, sometimes also referred to as local-vertical (LV) frame, and is commonly used in global navigation satellite system (GNSS) applications [Hoffmann-Wellenhof et al. 2007]. It is illustrated with respect to the LCLF/$\mathcal{I}$ frame in Figure 3.2. The axes are given as:

$\hat{\boldsymbol{\imath}}_{\mathcal{L}}$: Pointing east; parallel to equator.
$\hat{\boldsymbol{\jmath}}_{\mathcal{L}}$: Pointing north; along local meridian.
$\hat{\boldsymbol{k}}_{\mathcal{L}}$: Pointing up with respect to the local horizontal plane. In other words, the vector is collinear with the vector that defines the location of the landing site, $\hat{\boldsymbol{k}}_{\mathcal{L}} \parallel \boldsymbol{r}_{\mathcal{L}}$, which is defined w. r. t. the $\mathcal{I}$-frame.

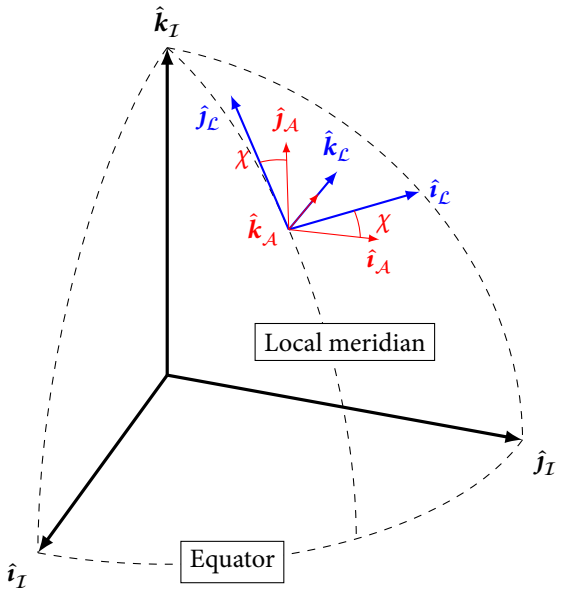


FIGURE 3.2 Relation of the landing-site frame to the LCLF frame.

3.1.5 Approach frame, index $\mathcal{A}$

The $\mathcal{A}$-frame originates from the currently targeted landing site, and is thus located on the surface of the central body. This frame is derived from the $\mathcal{L}$-frame, and deviates from it only by a rotation about the z-axis of the spacecraft's heading angle χ, as shown in Figure 3.2.

$\hat{\boldsymbol{i}}_\mathcal{A}$: Aligned with the spacecraft's velocity vector.
$\hat{\boldsymbol{j}}_\mathcal{A}$: Completes the triad.
$\hat{\boldsymbol{k}}_\mathcal{A}$: Pointing up with respect to the local horizontal.

3.1.6 Local-vertical, local-horizontal frame, index $\mathcal{V}$

This frame is the same as the $\mathcal{L}$-frame, save for its origin. Rather than the landing site, the $\mathcal{V}$ frame originates from the spacecraft's COM. Given its ENU orientation, it is thus also called local-vertical, local-horizontal (LVLH) frame.

$\hat{\boldsymbol{i}}_\mathcal{L}$: Pointing east, parallel to equator.
$\hat{\boldsymbol{j}}_\mathcal{L}$: Pointing north, along local meridian.
$\hat{\boldsymbol{k}}_\mathcal{L}$: Pointing up with respect to the local horizontal plane. In other words, the vector is collinear with the spacecraft position-vector, $\hat{\boldsymbol{k}}_\mathcal{L} \parallel \boldsymbol{r}(t)$, which is defined w. r. t. the $\mathcal{I}$-frame.

3.2 STATE REPRESENTATIONS

Kinematics is the study of describing a body's geometrical state with respect to a reference, irrespective of its cause. It gathers methods to do so and can be loosely divided into two classes: *point* or *translational kinematics* (the three-DOF position and motion of point masses), and *rigid body or rotational kinematics* (the three-DOF rotation and motion of non-infinitesimally sized bodies). The collected kinematic information of a point mass or rigid body is termed a *state*. As mentioned in Section 3.1.2, only translational motion is modelled in this thesis.

The state is mathematically described by a *state vector*. For example, if position $\boldsymbol{r}$ and velocity $\dot{\boldsymbol{r}}$ were described by Cartesian components:

$$\boldsymbol{r} = (x \quad y \quad z)^\mathsf{T} \quad \dot{\boldsymbol{r}} = (\dot{x} \quad \dot{y} \quad \dot{z})^\mathsf{T} \tag{3.1}$$

then *a* possible state vector could be chosen as:

$$\boldsymbol{x} = (\boldsymbol{r}^\mathsf{T} \quad \dot{\boldsymbol{r}}^\mathsf{T})^\mathsf{T} \tag{3.2}$$

State *representations* that allow for the formulation of such vectors to conveniently analyze problems are explained in this section. Great sources for all of the following discussions are [Schaub and Junkins 2009] and [Mooij 1998].

3.2.1 Coordinate systems

Coordinates provide means to describe vectors with respect to some set of axes defined by a reference frame. The term coordinate system can be defined as follows:

Coordinate system A mathematical description of vectors with respect to a given axis system.

Two basic types of coordinate systems are introduced in the following.

3.2.1.1 *Cartesian coordinates*

The most-used system is the *Cartesian* coordinate system. It describes vectors by forming scaled linear-combinations along the unit vectors of the reference frame. In simpler terms, one multiplies each unit vector of the coordinate axes with a scalar, and then adds the resulting vectors to obtain the desired vector. This principle is shown in Figure 3.3 for the definition of a vector $\boldsymbol{r}$.

Mathematically, vectors in the Cartesian coordinate system are thus defined as follows:

$$\boldsymbol{r} = \boldsymbol{r}^{\mathcal{F}} = x\,\hat{\boldsymbol{\imath}}_{\mathcal{F}} + y\,\hat{\boldsymbol{\jmath}}_{\mathcal{F}} + z\,\hat{\boldsymbol{k}}_{\mathcal{F}} \tag{3.3}$$

with $\mathcal{F}$ indicating an arbitrary frame.

3.2.1.2 *Spherical coordinates*

Although Cartesian coordinates can be easily visualised, many problems can be analyzed far easier if coordinates based on angles and a radial component are chosen.

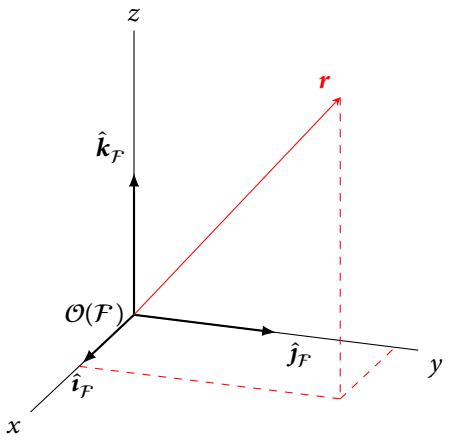


FIGURE 3.3 The general Cartesian coordinate system for defining vectors.

The definition for *spherical* coordinate systems adopted in this report is shown in Figure 3.4. Coordinates in this system are specified by three parameters: a longitudinal component τ, a latitudinal component δ, and the radial distance r (which is the modulus of the vector $\boldsymbol{r}$). Sometimes, the co-latitude $\tilde{\delta}$ is used rather than the latitude. The longitudinal component τ is measured from $-180°$ to $180°$, positive in eastern direction.[4] The latitude ranges between $-90° \leq \delta \leq 90°$, with the negative direction pointing south, whereas the co-latitude ranges from $0° \leq \tilde{\delta} \leq 180°$.

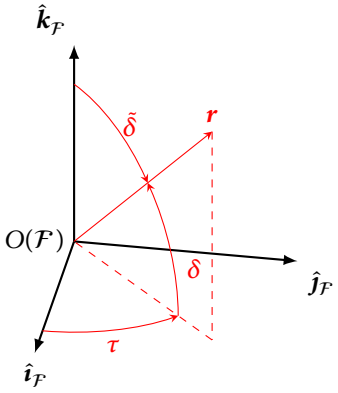


FIGURE 3.4 The general spherical coordinate system for defining vectors. Longitude and latitude are drawn positively here.

3.2.2 Translational state representations

To describe a translational state one needs to express the position, velocity, and acceleration with respect to some reference. Apart from position vectors, different ways for expressing velocity exist. These will be introduced in this section. Two commonly used options for describing translational states are described in the following: representations based on basic coordinate systems and orbital elements.

4 Note that the geographical terms "east", "west", "north", and "south" will also be used for problems not involving any kind of body such as a planet. This is for convenience and the intuitive description of directions in this fashion. Northern direction, for example, generally means "in the direction pointing to the positive z-axis", with the other directions termed accordingly.

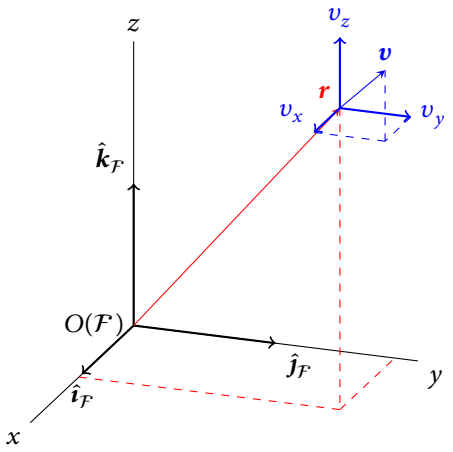


FIGURE 3.5 The Cartesian coordinate system for representing translational states.

3.2.2.1 *Cartesian components*

If a state is expressed using Cartesian components, the position vector $\boldsymbol{r}$ is simply formed as described before. However, the velocity also needs to be expressed somehow to complete the state. Although several ways exist for this, only the simple Cartesian components will be needed. These are shown in Figure 3.5. The velocity vector is decomposed along the coordinate axes in this representation.

The convention for denoting vectors in this system is:

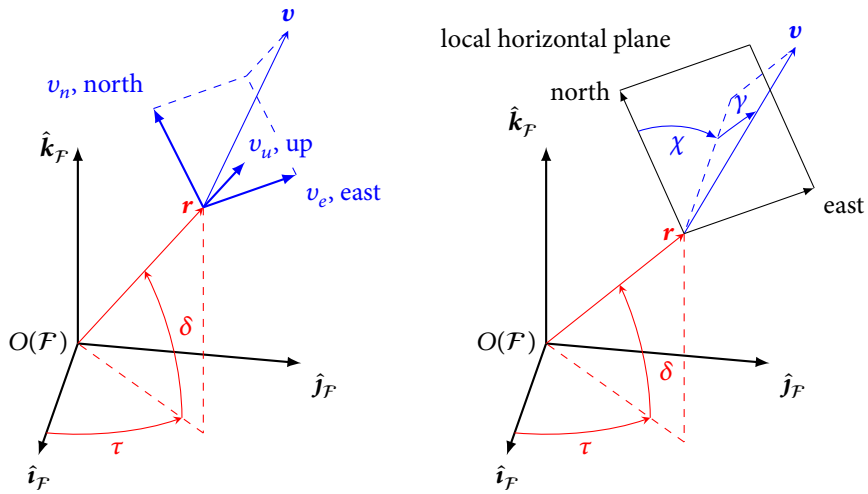
$$\boldsymbol{r} = (x \quad y \quad z)^{\mathsf{T}}, \quad \dot{\boldsymbol{r}} = (\dot{x} \quad \dot{y} \quad \dot{z})^{\mathsf{T}}$$

Accelerations are simply the time derivatives of the individual velocity-vector components, and are thus defined in the same manner.

3.2.2.2 *Spherical components*

There are different ways of describing velocities with respect to the spherically defined position vector $\boldsymbol{r}$. Two convenient definitions are shown in Figure 3.6(a) and Figure 3.6(b). For the Cartesian definition in (a), the velocity components are defined along the north-east-down (NED) directions: a latitudinal (north), a longitudinal (east), and a radial component (down). The first two components may thus be considered "tangential" to the velocity vector. This has the advantage that the downward velocity component will be positive, as will usually be the case in landing manoeuvres.

In the spherical convention (Figure 3.6(b)), the velocity is defined with the heading χ, the flight-path angle γ (positive below the local horizontal), and the modulus of the velocity vector $\boldsymbol{v}$. The heading is measured in the local horizontal-plane from north to the ground speed, which is the projection of the velocity vector $\dot{\boldsymbol{r}}$ onto the local horizontal. The heading is defined between $-180° \leq \chi < 180°$, and consequently $\chi = 90°$ indicates flight into easterly direction, parallel to the equator. The flight-path angle is defined between $-90° \leq \gamma < 90°$, and is measured between the local horizontal and the velocity vector.



(a) Cartesian velocity. **(b)** Spherical parameters.

FIGURE 3.6 The spherical coordinate system. All parameters shown here are drawn positively.

3.2.2.3 *Orbital elements*

In the previous two sections, different parametrizations of position and velocity to describe a state have been introduced. These are not necessarily the most convenient descriptions, however. In exoatmospheric flight, orbital elements are commonly used to express states of motion around central bodies. Orbital elements are needed in this study to calculate the lander's state at PDI.

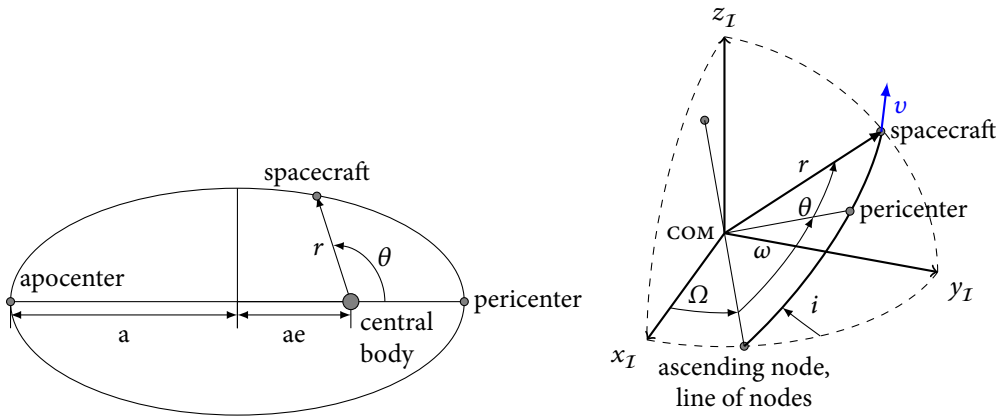
The Keplerian set of orbital elements is shown in Figure 3.7. The spacecraft's motion is of the shape of a conic section, in this case an ellipse. It revolves around the central body, which is at one of the foci of the ellipse. Six parameters describe the state:

- e: eccentricity ($0 \leq e < 1$)
- a: semi-major axis ($a > R$, the radius of the central body)
- i: inclination ($0° \leq i \leq 180°$)
- ω: argument of pericenter ($0° \leq \omega < 360°$)
- Ω: right ascension of the ascending node (RAAN) ($0° \leq \Omega < 360°$)
- θ: true anomaly ($0° \leq \theta < 360°$)

Although position and velocity are not obviously readable from these elements, they are indirectly contained in the set and can be calculated.

The orbit's shape is defined by two parameters: the semi-major axis a, which determines the size, and the eccentricity e, which determines how much the ellipse deviates from a circle. For $e = 0$ the ellipse degenerates to a circle and $e = 1$ represents a parabola. Every value in between describes ellipses.

The orientation of the orbit with respect to inertial space is defined by three parameters: the inclination i, the RAAN, and the argument of pericenter ω. The out-of-plane relation i is measured between the x_I-y_I plane and the orbital plane. The point where these two planes cross, and which the satellite crosses from below as seen from the x_I-y_I plane, is called the *ascending node*. The location of this point is defined by Ω, the RAAN, and is taken from the x_I axis. Finally, the argument of pericenter ω is the angle



(a) Planar ellipse. **(b)** 3D representation.

FIGURE 3.7 State representation using classical orbital (Keplerian) elements.

between the line of nodes (the line connecting the ascending node and the descending node) and the pericenter.

As shown in Figure 3.7(a), the location of a spacecraft at a given epoch can be described by the radius r and the true anomaly θ. These parameters can be determined for given times by considering the mean anomaly M, an auxiliary orbital parameter relating position to time. A discussion of this is beyond the scope, however, and may simply be looked up in many sources if necessary (such as [Vallado 2007]).

3.2.2.4 *Coordinate system conversions*

Although a standard procedure, the conversion between the two types of coordinate systems introduced in this section shall also be given for completeness' sake. Obtaining orbital elements from a Cartesian position and velocity, or vice versa, is rather involved, and may be looked up in [Vallado 2007] if needed. It will not be laid out here.

To convert from Cartesian components to spherical coordinates, one can apply the following formulae:

$$r = \sqrt{x^2 + y^2 + z^2} \tag{3.4}$$

$$\tau = \arctan \frac{y}{x} \tag{3.5}$$

$$\delta = \arcsin \frac{z}{r} \tag{3.6}$$

Inversely, one can obtain Cartesian coordinates from any spherical set using:

$$x = r \cos\delta \cos\tau \tag{3.7}$$

$$y = r \cos\delta \sin\tau \tag{3.8}$$

$$z = r \sin\delta \tag{3.9}$$

The relation between latitude and co-latitude is:

$$\delta = 90° - \tilde{\delta} \tag{3.10}$$

$$\tilde{\delta} = 90° - \delta \tag{3.11}$$

This concludes the discussion on translational three-DOF states. The following section will elaborate in detail on how vectors can be converted between reference frames.

3.3 FRAME TRANSFORMATIONS

To make use of the beneficial properties of the different reference frames one has to be able to convert between them. This requires knowledge of two features: the proper definition of the frames, and attitude parameters that express the relative angular orientation of the frames. Because rotational motion is ignored the discussion of attitude parameters is kept at a minimum. The necessary transformations are explained and derived in this section for right-handed Cartesian systems.

The translation of frames will be denoted by the translation vector $\boldsymbol{t}_{\mathcal{B}}^{\mathcal{A}}$, where $\mathcal{A}$ and $\mathcal{B}$ are the appropriate indices of the frames. Vectors that can be represented in different frames will be marked as $\boldsymbol{r}^{\mathcal{F}}$, where $\mathcal{F}$ indicates in which frame it is expressed. For the rotation of two frames with respect to each other, unit-axis rotations using Euler angles (and consequently direction cosine matrices (DCMs)) are used. The case of frames that are also translated with respect to each other is discussed after that. A brief primer on DCMs is given next. For an extensive discussion on this topic, see [Shuster 1993].

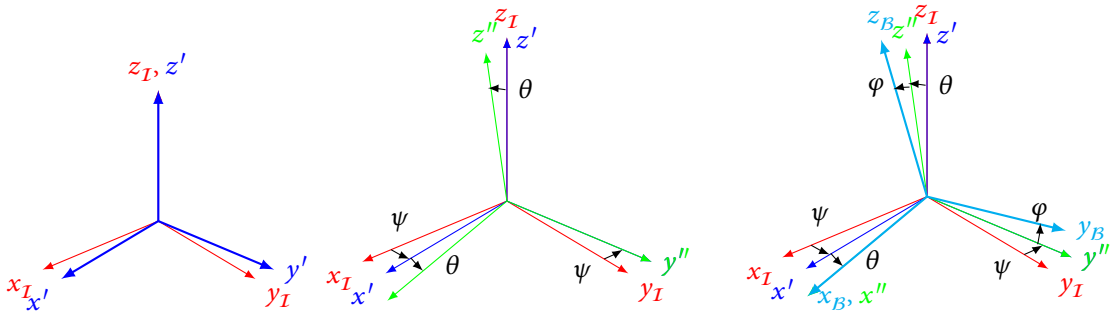
3.3.1 Euler-angle derived DCMs for rotational frame transformations

Euler angles are one of many possible *attitude representations* that describe the orientation of one frame with respect to another. They are based on the fact that any orientation of two frames in $\mathbb{R}^3$ can be expressed by at most three successive rotations around coordinate axes. This is illustrated in Figure 3.8, which shows how the attitude of $\mathcal{B}$ is expressed with respect to $\mathcal{I}$ using the three angles φ, θ, and ψ. The individual rotations about the (intermediate) x-, y-, and z-axes respectively can be expressed using *rotation matrices*, that read:

$$\mathsf{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{3.12}$$

$$\mathsf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.13}$$

$$\mathsf{R}_z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$



(a) Rotation about z. **(b)** Rotation about intermediate axis y'. **(c)** Rotation about intermediate axis x", final orientation.

FIGURE 3.8 Successive unit-axis rotations yield a certain orientation. Here it was rotated by 10° (positive) about all axes.

These three matrices are DCMs that do not only describe the orientation of one frame to another, but can also be used to transform vectors between frames. The angular orientation of the frames $\mathcal{B}$ and $\mathcal{I}$ is expressed by such a rotation matrix as:

$$\mathsf{R}_{\mathcal{I}}^{\mathcal{B}} = \mathsf{R}_x(\varphi)\mathsf{R}_y(\theta)\mathsf{R}_z(\psi) \tag{3.15}$$

The process of transforming a vector can be explained as follows. Consider a vector $\boldsymbol{r}$, and suppose that this vector has some *inertial* orientation. Then this vector can be expressed by its components in different frames, which is denoted by a superscript: $\boldsymbol{r}^{\mathcal{I}}$ if it is expressed in the $\mathcal{I}$ frame, and $\boldsymbol{r}^{\mathcal{B}}$ for the other. If the orientation of the two frames is known through the matrix $\mathsf{R}_{\mathcal{I}}^{\mathcal{B}}$, and the components of the vector are known in the $\mathcal{I}$ frame, then it can be projected onto the $\mathcal{B}$ frame as follows: [5]

$$\boldsymbol{r}^{\mathcal{B}} = \mathsf{R}_{\mathcal{I}}^{\mathcal{B}}\boldsymbol{r}^{\mathcal{I}} \tag{3.16}$$

This operation can also be inverted if the vector is given in the $\mathcal{B}$ frame, but it is to be known in the $\mathcal{I}$ frame:

$$\boldsymbol{r}^{\mathcal{I}} = \mathsf{R}_{\mathcal{B}}^{\mathcal{I}}\boldsymbol{r}^{\mathcal{B}} = \mathsf{R}_{\mathcal{I}}^{\mathcal{B}\mathsf{T}}\boldsymbol{r}^{\mathcal{B}} \tag{3.17}$$

3.3.2 Transformations involving translated frames

Apart from frames that are merely related to each other by a rotation, it also occurs that they are translated with respect to each other. For instance, this is the case for the landing-site frame $\mathcal{L}$, that has its origin at the landing site rather than the Moon's COM, where the $\mathcal{I}$ frame originates. This case is illustrated in Figure 3.9. Translations of

5 At this point it should be noted that the terminology "rotation matrix" is to be read with caution, because it could be misleadingly understood as if a physical rotation was to be performed on the vector. This is *not* the case. As a matter of fact, no vector is ever actively rotated in any frame conversion! The transformation can be better pictured if one imagines the vector to be converted as "inertial", and one considers the transformation as a projection of the vector from one frame to another.

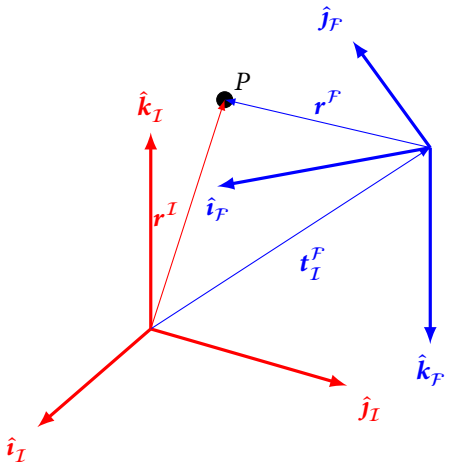


FIGURE 3.9 The case of a translated and rotated frame with respect to the inertial one.

frames are indicated by the translation vector $\boldsymbol{t}_{\mathcal{I}}^{\mathcal{F}}$, which is to be read as the translation of $\mathcal{F}$ with respect to $\mathcal{I}$ in accordance to rotation matrices.

To explain the process of transforming position vectors from one frame to another, consider point P in Figure 3.9. This point can be expressed in the two different frames as either $\boldsymbol{r}^{\mathcal{I}}$ or $\boldsymbol{r}^{\mathcal{F}}$. Firstly, consider the case that the vector is known in the $\mathcal{I}$ frame, the translation is known, and $\boldsymbol{r}^{\mathcal{F}}$ is to be found. Then it is most intuitive to first express the vector $\boldsymbol{r}^{\mathcal{F}}$ in the $\mathcal{I}$ frame, and then project it from the $\mathcal{I}$ frame onto the $\mathcal{F}$ frame using the corresponding transformation matrix:

$$\boldsymbol{r}^{\mathcal{F}} = \mathsf{R}_{\mathcal{I}}^{\mathcal{F}} \left(\boldsymbol{r}^{\mathcal{I}} - \boldsymbol{t}_{\mathcal{I}}^{\mathcal{F}}\right). \tag{3.18}$$

1. Vector from $\mathcal{O}(\mathcal{F})$ to P, expressed in $\mathcal{I}$.
2. Projection onto $\mathcal{F}$.

The opposite can be performed as well, if the vector $\boldsymbol{r}^{\mathcal{I}}$ is to be found given $\boldsymbol{r}^{\mathcal{F}}$ and $\boldsymbol{t}_{\mathcal{I}}^{\mathcal{F}}$. One first expresses $\boldsymbol{r}^{\mathcal{I}}$ in the $\mathcal{F}$ frame, and then projects it onto $\mathcal{I}$. To do that, the translation vector must be known in the $\mathcal{F}$ frame:

$$\boldsymbol{t}_{\mathcal{F}}^{\mathcal{I}} = \mathsf{R}_{\mathcal{I}}^{\mathcal{F}} \boldsymbol{t}_{\mathcal{I}}^{\mathcal{F}} \tag{3.19}$$

Then the projection can be carried out:

$$\boldsymbol{r}^{\mathcal{I}} = \mathsf{R}_{\mathcal{F}}^{\mathcal{I}} \left(\boldsymbol{t}_{\mathcal{F}}^{\mathcal{I}} + \boldsymbol{r}^{\mathcal{F}}\right). \tag{3.20}$$

1. Vector from $\mathcal{O}(\mathcal{I})$ to P, expresssed in $\mathcal{F}$.
2. Projection onto $\mathcal{R}$.

Substituting Equation [3.19] into Equation [3.20], this equation can be further simplified:

$$\boldsymbol{r}^{\mathcal{I}} = \mathsf{R}_{\mathcal{F}}^{\mathcal{I}} \left(\mathsf{R}_{\mathcal{I}}^{\mathcal{F}} \boldsymbol{t}_{\mathcal{I}}^{\mathcal{F}} + \boldsymbol{r}^{\mathcal{F}}\right) \tag{3.21}$$

$$= \boldsymbol{t}_{\mathcal{I}}^{\mathcal{F}} + \mathsf{R}_{\mathcal{F}}^{\mathcal{I}} \boldsymbol{r}^{\mathcal{F}} \tag{3.22}$$

where it has been made use of the identity $\mathsf{R}_{\mathcal{I}}^{\mathcal{F}}\mathsf{R}_{\mathcal{F}}^{\mathcal{I}} = \mathsf{I}$, which holds for all DCMs. Note that as such, $\mathsf{R}_{\mathcal{I}}^{\mathcal{F}\mathsf{T}} = \mathsf{R}_{\mathcal{F}}^{\mathcal{I}}$.

3.3.3 Frame-transformation relationships

The different frames discussed in Section 3.1 are related to one another in a hierarchical way. This structure is shown in Figure 3.10. The transformations that relate the individual frames are explained in the remainder of this section.

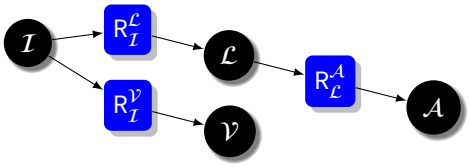


FIGURE 3.10 Frank diagram for the hierarchy of frame transformations.

3.3.3.1 *From inertial to landing-site frame,* $\mathsf{R}_{\mathcal{I}}^{\mathcal{L}}$

Because the landing-site frame $\mathcal{L}$ is translated w. r. t. to the main reference frame $\mathcal{I}$, the relations of Section 3.3.2 need to be applied. Consider Figure 3.11. If one is to convert from $\mathcal{I}$ to $\mathcal{L}$, only the vector pointing to the landing-site, $\boldsymbol{t}_{\mathcal{I}}^{\mathcal{L}}$, and the position vector of the spacecraft in the $\mathcal{I}$-frame, $\boldsymbol{r}^{\mathcal{I}}$, are known.

To be found is the position vector of the spacecraft in the $\mathcal{L}$-frame. The rotation matrix can be derived remembering that the $\mathcal{L}$-frame is defined as ENU.

Related variables:

- $\boldsymbol{t}_{\mathcal{I}}^{\mathcal{L}}$: Vector from $\mathcal{O}(\mathcal{I})$ to the landing site.
- $\boldsymbol{r}^{\mathcal{I}}$: Vector from $\mathcal{O}(\mathcal{I})$ to the spacecraft's COM.
- $\tau_{\mathcal{L}}$: Longitude of the landing site.
- $\delta_{\mathcal{L}}$: Latitude of the landing site.

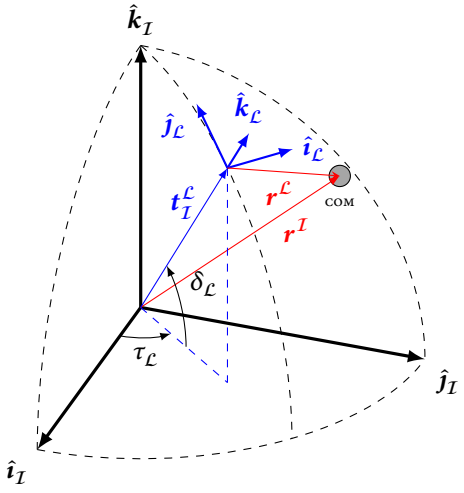


FIGURE 3.11 Relation between the LCLF frame and the landing-site frame, and the expression of the spacecraft's position vector in both frames.

Transformation matrix:

$$\mathsf{R}_{\mathcal{I}}^{\mathcal{L}} = \mathsf{R}_x\left(\frac{\pi}{2} - \delta\right)\mathsf{R}_z\left(\frac{\pi}{2} + \tau\right) = \begin{bmatrix} -\mathrm{s}\,\tau_{\mathcal{L}} & \mathrm{c}\,\tau_{\mathcal{L}} & 0 \\ -\mathrm{c}\,\tau_{\mathcal{L}}\mathrm{s}\,\delta_{\mathcal{L}} & -\mathrm{s}\,\tau_{\mathcal{L}}\mathrm{s}\,\delta_{\mathcal{L}} & \mathrm{c}\,\delta_{\mathcal{L}} \\ \mathrm{c}\,\tau_{\mathcal{L}}\mathrm{c}\,\delta_{\mathcal{L}} & \mathrm{s}\,\tau_{\mathcal{L}}\mathrm{c}\,\delta_{\mathcal{L}} & \mathrm{s}\,\delta_{\mathcal{L}} \end{bmatrix} \quad [3.23]$$

Full transformation:

$$\boldsymbol{r}^{\mathcal{L}} = \mathsf{R}_{\mathcal{I}}^{\mathcal{L}}\left(\boldsymbol{r}^{\mathcal{I}} - \boldsymbol{t}_{\mathcal{I}}^{\mathcal{L}}\right) \quad [3.24]$$

The inverse transform can be obtained as outlined in Section 3.3.2, by taking the transpose of the rotation matrix:

$$\boldsymbol{r}^{\mathcal{I}} = \boldsymbol{t}_{\mathcal{I}}^{\mathcal{L}} + \mathsf{R}_{\mathcal{L}}^{\mathcal{I}}\boldsymbol{r}^{\mathcal{L}} \quad [3.25]$$

3.3.3.2 *From landing-site frame to approach frame,* $\mathsf{R}_{\mathcal{L}}^{\mathcal{A}}$

The approach frame $\mathcal{A}$ is related to the landing-site frame by a rotation of the heading angle about the z-axis.

Related variables:

χ: Spacecraft heading angle. This is computed in the $\mathcal{L}$-frame as: $\chi = \mathrm{atan2}(v_x/v_y)$. See Figure 3.6(b), Page 20.

Transformation matrix:

$$\mathsf{R}_{\mathcal{L}}^{\mathcal{A}} = \mathsf{R}_z\left(\frac{\pi}{2} - \chi\right) = \begin{bmatrix} \mathrm{s}\,\chi & \mathrm{c}\,\chi & 0 \\ -\mathrm{c}\,\chi & \mathrm{s}\,\chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [3.26]$$

3.3.3.3 *From inertial to local-vertical, local-horizontal frame,* $\mathsf{R}_{\mathcal{I}}^{\mathcal{V}}$

This transformation uses the same transformation matrix as for the landing-site frame, however, now applied to the spacecraft's current position rather than the landing-site coordinates.
Related variables:

τ_r: Longitude of the sub-spacecraft point.
δ_r: Latitude of the sub-spacecraft point.

Transformation matrix:

$$\mathsf{R}_{\mathcal{I}}^{\mathcal{V}} = \mathsf{R}_x\left(\frac{\pi}{2} - \delta\right)\mathsf{R}_z\left(\frac{\pi}{2} + \tau\right) = \begin{bmatrix} -\mathrm{s}\,\tau_r & \mathrm{c}\,\tau_{\mathcal{L}} & 0 \\ -\mathrm{c}\,\tau_r\mathrm{s}\,\delta_r & -\mathrm{s}\,\tau_r\mathrm{s}\,\delta_r & \mathrm{c}\,\delta_r \\ \mathrm{c}\,\tau_r\mathrm{c}\,\delta_r & \mathrm{s}\,\tau_r\mathrm{c}\,\delta_r & \mathrm{s}\,\delta_r \end{bmatrix} \quad [3.27]$$

3.4 EQUATIONS OF MOTION

After discussing translational *kinematics* throughout the past sections, the *dynamics* of point masses can now be introduced. Dynamics describes the effect of forces and

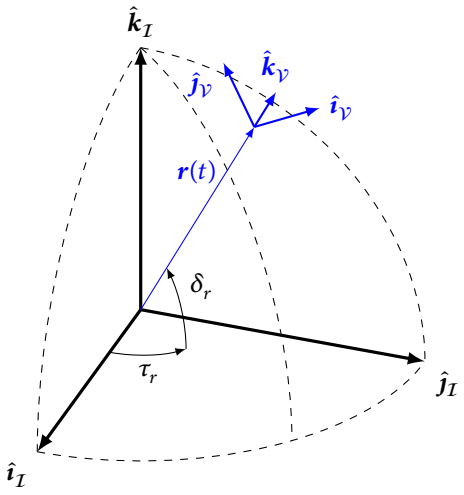


FIGURE 3.12 Relation between the LCLF and the LVLH.

moments on motion, while kinematics concerns itself with the change of position due to a velocity (irrespective of the cause of the velocity change).

Translational EOMS can be derived using different state representations, which can suit a variety of applications. In this thesis all developments are made with Cartesian components, which is why the EOMS are not presented for spherical components (although this is also often used in guidance applications).

The EOMS are derived using classical mechanics and rest on the foundations of *Galileo's principle of relativity, the laws of conservation of mass, linear momentum, and energy,* and three basic postulates: *Newton's Laws of Motion* [Cornelisse et al. 1979, p. 25]. These foundations are documented in detail in [Schaub and Junkins 2009, pp. 31-33], and summarized in [Gerth 2013].

An additional assumption is made for the derivation:

Assumption 3: coriolis force ignored Effects due to variable mass and rotational rates are ignored.

The Coriolis force due to variable mass is in fact commonly neglected but at the same time widely unknown, which is why it deserves a brief highlight here. In any system subject to variable mass, such as a rocket expelling propellant, a force appears if the system turns at finite rotational speed. The Coriolis force is necessary because of the principle of solidification, the details of which are described in [Cornelisse et al. 1979]. It reads [Cornelisse et al. 1979, p. 71]:

$$\boldsymbol{F}_c = -2\boldsymbol{\omega} \times \dot{m}\boldsymbol{r}_e \qquad [3.28]$$

where $\boldsymbol{\omega}$ is the rotational velocity vector of the system, $\dot{m}$ the mass flow of the propulsive plant, and $\boldsymbol{r}_e$ the vector pointing from the vehicle's COM to the point of application of the thrust force (the engine's nozzle).

A simple calculation suffices to show that this force can be ignored. The maximum mass flow of the reference vehicle is 1.27 kg/s (Section 2.2). The engine is located about 1.5 m away from the COM (Figure 2.6). Finally, the peak attitude rate is about 0.56 °/s, or

0.01 rad/s during approach (Figure 9.26(d), Page 199). The maximum expected Coriolis force is thus:

$$F_c = 2 \cdot 0.01 \cdot 1.27 \cdot 1.5 \approx 0.02\,\mathrm{N} \quad [3.29]$$

which is readily negligible. Mooij [1997, p. 43] confirms this assumption. In fact, not a single publication studied in the scope of this survey models the Coriolis force, so simply for the sake of comparability it should be ignored.

Having settled the assumptions, the EOMs can now be formulated. As introduced in Section 3.2.2.1, the translational state is written in Cartesian components as:

$$\boldsymbol{r} = (x \quad y \quad z)^\mathsf{T} \quad [3.30]$$

$$\dot{\boldsymbol{r}} = \boldsymbol{v} = (\dot{x} \quad \dot{y} \quad \dot{z})^\mathsf{T} \quad [3.31]$$

The EOM and the kinematic equation in the inertial frame are thus [Cornelisse et al. 1979]:

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}t^2} = \ddot{\boldsymbol{r}} = \boldsymbol{a} = (\ddot{x} \quad \ddot{y} \quad \ddot{z})^\mathsf{T} = \frac{\boldsymbol{F}_e}{m} \quad [3.32]$$

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \dot{\boldsymbol{r}} = \boldsymbol{v} = (\dot{x} \quad \dot{y} \quad \dot{z})^\mathsf{T} \quad [3.33]$$

where the external forces $\boldsymbol{F}_e$ will be defined in the next section.

3.5 FORCE MODELS

The previous section introduced EOMs that model the motion of spacecraft. This motion is governed by *forces* acting on the vehicle. The forces were simply treated as a black box in these equations, making an appearance through the term $\boldsymbol{F}_e$—without any further specification. It is the purpose of this section to give models that allow for computing the force vectors (magnitude and direction) acting on the spacecraft.

Gerth [2013] gives a complete overview of forces that would need to be considered in a high-fidelity simulation. However, only a small subset of those discussed in that report are needed for the analysis in this document. These will be discussed below.

3.5.1 Gravity-field model

Gravity is arguably the most important important force in spaceflight. Apart from thrust forces, it completely dominates the motion of space vehicles—the next strongest forces are more than *five* orders of magnitude lower [Montenbruck and Gill 2000]. Owing to this importance, many different gravity-field models exist, varying from very simple to extremely sophisticated.

The most important model is Newtonian gravity. Plainly termed the *inverse-square law*, this model for the gravitational acceleration is illustrated in Figure 3.13 and reads [Montenbruck and Gill 2000, p. 56]:

$$\boldsymbol{a}_g = \ddot{\boldsymbol{r}} = -\frac{Gm}{r^2}\frac{\boldsymbol{r}}{r} = -\frac{Gm}{r^2}\hat{\boldsymbol{r}} \quad [3.34]$$

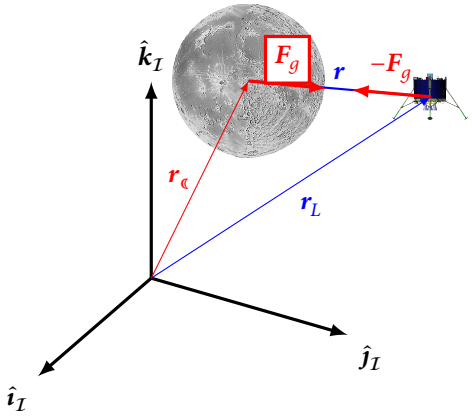


FIGURE 3.13 The Newtonian gravity-field model.

where G is the universal gravitational constant, and m the mass of the central body.

The term Gm is a recurring quantity in any equation describing motions in gravitational fields. It is thus customary to write it compactly in the following manner, indicating the respective central body (here the Moon, ☾):

$$\mu_☾ = Gm_☾ = 4.902\,801\,076 \cdot 10^{12}\ \mathrm{m}^3/\mathrm{s}^2 \qquad [3.35]$$

Another reason for using μ is that the product of mass and gravitational constant, Gm, can be determined up to a much higher degree of accuracy than any of the individual quantities on its own.

With Newton's second law, $\boldsymbol{F} = m\boldsymbol{a}$, the force can hence be written in either of the following ways:

$$\boldsymbol{F}_g = m\boldsymbol{a}_g = m\nabla U = -m\frac{\mu_☾}{r^2}\hat{\boldsymbol{r}} \qquad [3.36]$$

The simplest model assumes a flat Moon. In this model, the gravity is expressed as a constant vector in vertical direction:

$$\boldsymbol{a}_g = -g\hat{\boldsymbol{k}} = -\frac{\mu_☾}{R_☾^2}\hat{\boldsymbol{k}} \qquad [3.37]$$

Note that its magnitude is computed as the local gravitational acceleration on the surface of the Moon, where $R_☾$ is the mean radius.

3.5.2 Thrust forces

After gravity, thrust is the most important force in astrodynamics. The thrust is used to alter the trajectory governed by gravity, and steer the spacecraft where it is desired to go. It is thus essential to model the thrust properly.

In its simplest form, the thrust is modelled to the following formula [Sutton and Biblarz 2010, p. 35]:

$$\boldsymbol{T} = -\dot{m}\boldsymbol{v}_e + p_e A_e \hat{\boldsymbol{r}}_e \qquad [3.38]$$

Instead of the exhaust velocity $\boldsymbol{v}_e$, this equation is often written with the specific impulse I_{sp}—a measure for the efficiency of an engine. The I_{sp} is defined as:

$$I_{\text{sp}} = \frac{T}{\dot{m} g_0} = \frac{-\dot{m} v_e + p_e A_e}{\dot{m} g_0} \tag{3.39}$$

where $g_0 = 9.806\,65\ \text{m/s}^2$ is the standard gravitational constant. Note that this is just a scaling factor, and thus one must apply g_0 *irrespective* of the environment. Thus, one does *not* need to use a different value for g_0 when flying around the Moon or Mars, for example. A second observation from the equation is that, because of its dependency on the pressure, the specific impulsive is dependent on the environment in which it is measured. It will be different at sea-level and in vacuum. This must be taken into account when obtaining values for the I_{sp}.

As per the definition above, the I_{sp} bundles the pressure and the momentum thrust, and one can thus use the following simplified form in all implementations:

$$\boldsymbol{T} = \dot{m} I_{\text{sp}} g_0 \hat{\boldsymbol{T}} \tag{3.40}$$

where $\hat{\boldsymbol{T}}$ indicates the direction of the thrust. For some figures of merit corresponding to the LL, see Table 2.2 on Page 11. Possible refinements of this model that take into account rise- and settling-time, as well as pulsing, are explained in [Wertz 1979].

It is essential to remark that the I_{sp} appearing in the equations above is actually not a constant quantity if multiple engines are used. Rather, if there are multiple thrust sources, it must be computed as:

$$I_{\text{sp}} = \frac{\sum_{i=0}^{N} T_i}{\sum_{i=0}^{N} \frac{T_i}{I_{\text{sp}_i}}} \tag{3.41}$$

For the ESA LL the I_{sp} at maximum thrust is (see Table 2.2, Page 11):

$$I_{\text{sp}} = \frac{5 \cdot 500\ \text{N} + 6 \cdot 200\ \text{N}}{\frac{5 \cdot 500\ \text{N}}{323\ \text{s}} + \frac{6 \cdot 200\ \text{N}}{280\ \text{s}}} = 306.7\ \text{s} \tag{3.42}$$

As the thrust of the LL is realized by a variable number engines, the I_{sp} would thus have to be recomputed based on the output of the propulsion controller, that chooses a suitable setting based on the guidance command. Moreover, the I_{sp} is not perfectly constant for different thrust levels (in pulsed modes), which does not appear in the equation above. For simplicity and due to the lack of a propulsion controller this is ignored and the following assumption is made:

Assumption 4: constant I_{sp} Variations in I_{sp} due to engine clustering and throttle settings are ignored.

The baseline value used in this thesis is $I_{\text{sp}} = 306.7\ \text{s}$.

3.5.3 Perturbing forces

Apart from the major forces introduced in the previous section, *perturbations* of much smaller magnitude need to be taken into account in the mission planning. Especially if integration times are long (such as during main braking), small accelerations can cause a noticeable error. When determining the achievable landing-precision of a system through high-fidelity simulations, it is therefore essential to model such forces.

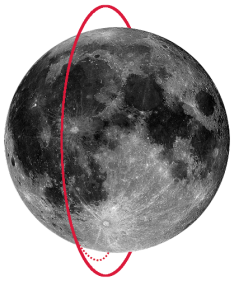
As this thesis focuses on theoretical guidance-algorithm development and neither mission design nor a full GNC system study, perturbations will be not be treated for now:

Assumption 5: no perturbing forces Perturbations of any kind, such as third-body, irregular gravity, or solar radiation-pressure forces are ignored.

This decision can be well motivated by the following:

- For the lack of a full GNC model, the landing precision cannot be studied well either way. By leaving out perturbations, all results will therefore be traceable *only* to guidance. This also allows to clearly separate guidance from other influences, so its performance can be better evaluated in an isolated fashion.
- In practice perturbations are usually corrected by a trajectory-tracking controller. Any residual errors at landing would thus not only be caused by guidance imperfections, but also the controller. This makes the result dependent on a subject that is not under study in this report, and would be unfair for comparing guidance algorithms.
- If, for example, Lunar gravity-field irregularities are modelled, these need to be taken into account up to a high degree, because there is no single dominating component (such as there is J_2 on Earth) [Zuber et al. 2013]. This will significantly slow down simulations.
- As soon as irregular gravity is taken into account, it becomes necessary to model third-body perturbations as well, because Earth still exerts a noticeable gravitational pull at the Moon due its disproportionately larger size [Vallado 2007].
- Taking into account perturbations requires extra labor during the thesis, which does not fit the schedule.
- Motion can now no longer be modelled in a fixed inertial frame, but rather the simulation complexity increases, because the rotation of the Moon has to be taken into account. This means that guidance would also have to be adapted to be fairly evaluated.
- Taking into account perturbations makes simulations epoch and landing-site dependent. The results are no longer generally applicable.

In conclusion, there is a strong case for modelling only simple thrust and gravity-forces. The result will be a basic, but sufficiently accurate simulation of the lander's motion.



Chapter 4

FUNDAMENTAL GNC CONCEPTS

> Surely there is a time to submit to guidance and a time to take one's own way at all hazards.
>
> *Thomas Huxley (Evolutionary Biologist, 1825-1895)*

Many spacecraft subsystems interact during planetary landings. When the engineers of the 1960s and 1970s built the first landers, one of the biggest challenges was to design GNC systems that would safely steer the spacecraft to touchdown softly on the surface. This had to be done within the limits of what was possible by using the hardware of the other sub-systems, such as the propulsive plant or the reaction control system (RCS). The successful implementation of GNC systems in harmony with the spacecraft technology was one of the biggest accomplishments of those missions. GNC systems were very complex developments, and still are today.

In planetary landers the GNC system constrains the landing path, and is thus a major part of the global mission scenario. The system acts almost like a conductor, that knows what is currently happening in the sequence, what has to be done next, and then commands the other systems how this is to be done. In layman's terms, navigation answers "Where am I?", guidance "Where am I going?", and control "How do I get there?". This chapter goes into detail about GNC systems, and which special needs there are for the ESA LL. This is directly relevant to guidance development, which is intertwined with the other systems.

In this chapter, some fundamental GNC concepts that do not only apply to landers are first explained Section 4.1. The specifics of GNC developments for future planetary landers are detailed in Section 4.2. The characteristics of optimal lading architectures are then discussed in Section 4.3. An overview of guidance heritage is given in Section 4.4. As the ESA LL addresses some new developments, the project's take at GNC is the subject of Section 4.5. Finally, based on all these discussions, the mathematical guidance problem is formulated in Section 4.6.

4.1 ELEMENTS OF GNC SYSTEMS

The GNC system of the LL is an advanced system enabling precise and safe landing on the Lunar surface. This is done using next-generation approaches such as vision-based navigation and HDA, which only recently have come into reach. However, before these intricacies can be discussed, the fundamentals of GNC shall be introduced first with a focus on landing guidance. This is the purpose of this section, which first defines the terms, then explains how GNC systems generally work and interact, and finally formulates the general guidance problem.

4.1.1 Definition of guidance, navigation, and control

As a basis for the subsequent discussion, it is vital to have clear definitions for the individual terms in GNC – guidance, navigation, and control – which are adapted from Pfeiffer [1968] in the following:

Guidance "is the task of calculating and executing a realizable acceleration profile which will cause the trajectory of the space vehicle to attain desired end conditions."

Navigation "is the task of estimating the state of the space vehicle system from sensed data."

Attitude Control "is the task of attaining and stabilizing the vehicle in the attitude configuration for the guidance system."

These definitions are intriguing because they define guidance, navigation, and control without any recursive dependency on other terms of GNC jargon. This comes at the price that the definitions are rather specific, and slightly biased towards landing GNC. More general definitions are documented in ECSS-E-60A [ECSS 2004] as follows:

Guidance function of the controller to define the current or future desired state

Navigation function of the controller to determine the current or future estimated state from the measured state

Control function of the controller to derive control commands to match the current or future estimated state with the desired state

In these definitions and this context, the term "controller" is synonymous with "GNC system".

From the definitions above it becomes clear that the GNC system is not a purely mechanical one. Rather, it is made up of:

- Guidance algorithms, navigation filters, and control laws, implemented as on-board software.
- Hardware such as star trackers, cameras, or IMUS for navigation and attitude determination, and thrusters for attitude control.

The *guidance "system" itself* thus does not use any hardware other than an on-board computer. Overall, the GNC system is more algorithm- and software-based than other subsystems.

4.1.2 GNC architecture and system interaction

The duties of a GNC system for a generic spacecraft are illustrated in Figure 4.1, inasmuch as it is possible at all to draw up a generic architecture for a system that is usually very specifically tailored. For the sake of clarity, the illustration is assumed to be general enough to explain the basic points. The master function is the guidance and control manager, or mission vehicle management (MVM). Its primary function is the

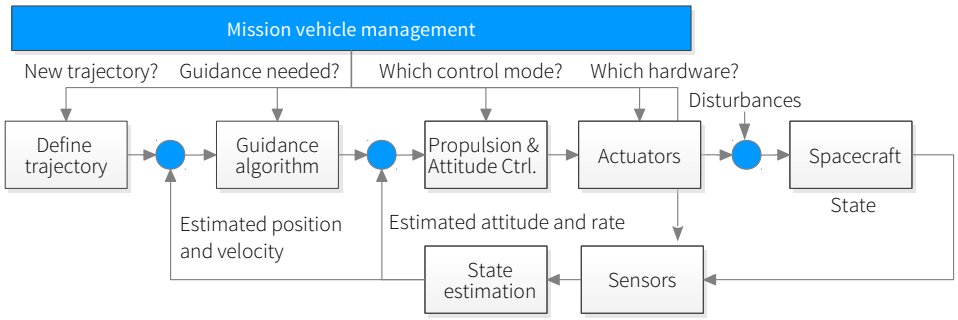


FIGURE 4.1 System interaction within a generic GNC architecture.

operations mode management, meaning it selects the GNC mode for steering the spacecraft according to different modes in all varieties of mission phases and circumstances. The MVM also selects the GNC hardware according to the current mode. It is further responsible for safe spacecraft operations by fault identification, detection, and recovery (FDIR), and thus also for reconfiguring the GNC system into a nominal or redundant mode.

Once a target is defined, a trajectory will be generated or preloaded from a lookup table on board. This input is used by guidance to determine the appropriate acceleration profile and spacecraft orientation to reach the destination. Most of the time these two functions are combined and the guidance algorithm also generates the trajectory.

Once an acceleration profile is defined, the attitude controller is responsible for establishing this attitude on the actual spacecraft. For that, it uses the reference (guidance) and the estimated state (navigation) as an input, and commands actuators to adjust the attitude. Additionally, a propulsion controller realizes the commanded thrust levels according to the capabilities of the system hardware, *e. g.*, it chooses an appropriate combination of thrusters. It shall be underlined that navigation errors are unavoidable, and the position and velocity of the spacecraft will thus only be known up to a certain accuracy. This will directly impact and limit the achievable landing precision for landers.

Based on the commanded acceleration profile, the propulsion controller will also set the appropriate thrust level for the engines, if possible. In case of non-throttleable engines, the propulsion controller can also modulate the thrust accordingly, or command a suitable sequential-shutdown strategy. Some systems also use gimballed engines for thrust-vector control, which is also taken of by the propulsion controller.

Disturbances act on the spacecraft, and sensors will measure the real attitude, angular rate, position, and velocity of the spacecraft and a navigation filter will estimate them in an optimal way. These are then fed back to complete the loop. Adjustments can follow in the next iteration if necessary.

The guidance function in particular is thus a commanding function that will point the spacecraft in a certain direction and command a thrust. It uses the current best estimate of the state to generate the trajectory, because it will have to guide the vehicle from the current position to the final position.

4.1.3 Separation of guidance and navigation

Following the remarks above it is fair to question whether guidance and navigation can really be split into two separate problems, as is done almost exclusively in derivations of guidance laws [Pfeiffer 1968]. After all, the guidance algorithm relies on estimated data, which is of stochastic nature. Navigation is never perfect in reality, but is *assumed to be* for developing guidance equations. Consequently, the state should thus in fact be expressed in terms of statistical moments. This in turn would lead to a stochastic control problem, and thus a stochastic guidance algorithm.

Fortunately, Pfeiffer [1968] points out that the stochastic nature of the considered problem will not pose a major issue if the navigation errors are small. For the LL, this may be assumed, because the state will be initialized with precise ground-based tracking-data before the landing-phase [Schoenemaekers 2011]. Pfeiffer [1968] also shows that the guidance problem is not necessarily required to be treated as a stochastic one if the guidance phase is executed with continuous thrust.

Henceforth, the following assumption will be made:

Determinism The guidance problem can be treated as deterministic, and stochastic influences from imperfect navigation are ignored (ideal navigation is assumed).

Despite this assumption, it is important to perform a sensitivity analysis to determine the impact of imprecise navigation on guidance. In this manner, the impact of the stochastic problem on trajectory generation can be brought into the picture. Although large errors are not expected, this could potentially help to tune the guidance algorithm to better cope with uncertainties, or lead to required adaptations.

4.1.4 Formulation of the general guidance problem

The guidance problem can now finally be formulated. In Section 4.1.1 guidance was already defined as the task of computing a suitable acceleration profile to attain desired end conditions. Here, the problem will now be posed as a two-point boundary-value problem (TPBVP) as illustrated in Figure 4.2 and stated in Problem 1 below.

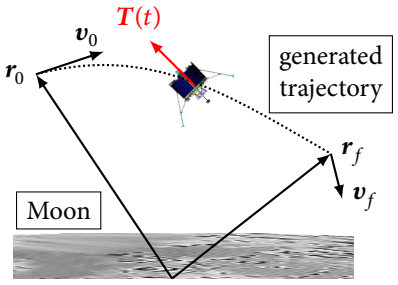


FIGURE 4.2 The guidance problem, formulated as a two-point boundary-value problem. Simplified here in a two-dimensional notional sketch.

▶ PROBLEM 1 The general guidance problem.

Given a set of initial states at time $t = t_0$

$$\boldsymbol{r}(t_0) = \boldsymbol{r}_0, \quad \boldsymbol{v}(t_0) = \boldsymbol{v}_0 \tag{4.1}$$

where $\boldsymbol{r}(t)$ is the position at time t and $\boldsymbol{v}(t)$ the velocity, find an acceleration profile and thus a trajectory

$$\boldsymbol{a}(t) = \frac{\boldsymbol{T}(t)}{m(t)}, \quad \forall \quad t \in [t_0, t_f] \tag{4.2}$$

where $\boldsymbol{a}(t)$ is the acceleration, $\boldsymbol{T}(t)$ is the thrust, $m(t)$ is the instantaneous spacecraft mass, and t_f is the final time, such that the spacecraft reaches a final state

$$\boldsymbol{r}(t_f) = \boldsymbol{r}_f \quad \boldsymbol{v}(t_f) = \boldsymbol{v}_f \tag{4.3}$$

with *some* given constraints

$$\begin{aligned} \boldsymbol{g} &\leq \boldsymbol{0} \\ \boldsymbol{h} &= \boldsymbol{0} \end{aligned} \tag{4.4}$$

Many methods are available for solving this problem, and a particularly promising novel method – convex guidance – will be extended in this thesis.

4.2 TECHNOLOGICAL DEVELOPMENTS FOR ROBOTIC LANDINGS

Having established a basic understanding of GNC, it can be explained why a simple system will not suit all missions. Requirements for some of today's landers, such as the LL, require precision-landing capabilities and will need to evade hazards on the planet's surface. Later it will become clear that this calls for special technologies and also impacts guidance, which is central to this report.

4.2.1 Reasons for precision landing

One of the conclusions from the call for the continued exploration of the Moon by Crawford et al. [2012] is that landers will greatly benefit the scientific value, and *polar* landers especially so. Mission designers will face problems when aiming for the poles, however, because most of the terrain is in darkness for extended periods of time. The operation of a lander in a location with long periods of darkness is only possible with the use of nuclear power sources.

Unfortunately, Europe currently does not possess these technologies and must rely on solar thermal and electrical power. This complicates the mission design significantly, because landing sites are now constrained to locations that are in sunlight sufficiently long. On the bright side (literally and figuratively), there are areas on the South Pole that have a very long so-called "longest quasi-continuous illumination-period (LQCIP)". At these places, the spacecraft could be in sunlight for as long as, and even more than, three months *continuously*. This makes polar missions feasible for ESA, and

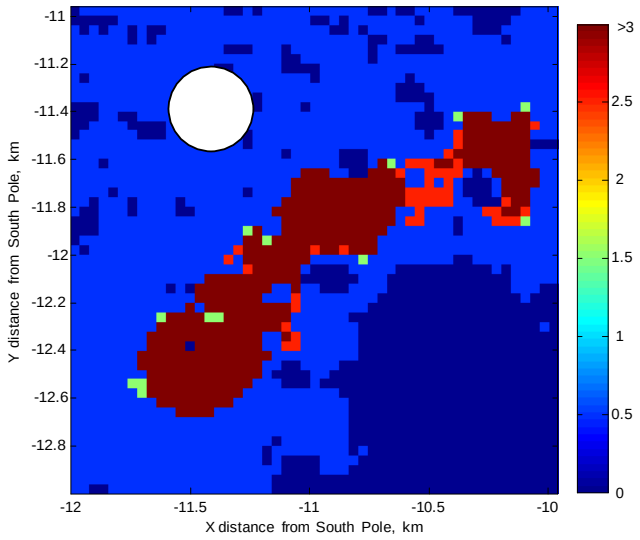


FIGURE 4.3 Longest quasi-continuous illumination-period in months at a potential site (connecting ridge), for year 2019. Darkness survivability of 100 hours and an observer height of 3 m. Grid size is 40 m. A circle of 200 m radius is reported for reference. Source: [De Rosa et al. 2011]

thus gives the opportunity to perform valuable science and demonstrate landing technologies.

An illumination map for a candidate site at the Lunar South Pole is shown in Figure 4.3. Red-colored pixels represent landing areas that fulfil the illumination requirements, as they have a sufficiently long LQCIP. To give a scale of reference, the white circle is drawn with a 200 m radius. It is thus noticed that the potential landing areas are very small. This is the primary reason necessitating precision-landing capabilities for landing at the South Pole if nuclear space technologies are unavailable.

Scientific reasons and the above-mentioned constraints are not the only reasons to develop precision-landing technology. Most notably, the Apollo 12 mission shall be recalled once again. Apollo 12 landed only 150 m away from its target, Surveyor 3—this is better than LL requirements stipulate. In Apollo 14, pilot Edgar Mitchell even landed only 26.5 m away from the intended site—although there is no direct mission rationale that would have required a particularly precise landing. The second reason for precision landing technologies is thus to enable *landing near surface assets* like previously landed spacecraft, such as sample-return cache-vehicles, or near human outposts.

The achievement of Apollo 12 is depicted in the remarkable picture Figure 4.4, that also shows how the Surveyor programme supported Apollo. The picture was taken during a sortie of the astronauts. It is the only detailed picture of a robotic lander on another planet's surface, the only picture with an astronaut next to a robotic lander, and the only picture showing two landers in one scene.

To illustrate what "precise robotic landing" really means as compared to previous missions, consider the illustration in Figure 4.5. This drawing shows the landing ellipses of two previous landings as compared to a landing with Lunar South Pole constraints to be performed by the LL, which is indicated as the tiny white dot. This dot is by requirement in fact not elliptical, but a 200 m radius circular area [Diedrich 2011].

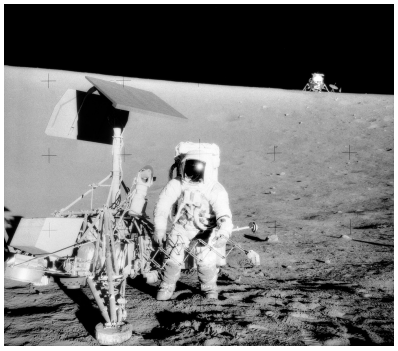


FIGURE 4.4 Peter Conrad next to Surveyor 3 with the LEM of Apollo 12 in the background. Source: NASA

As compared to Surveyor 3, the LL will have to improve its precision by a factor of 200 when one considers the semi-major axis of the landing ellipse. If one considers the size of the landing area, the landing precision will have to be improved by a factor of:[1]

$$\frac{39\,\text{km} \cdot 15\,\text{km}}{0.4\,\text{km} \cdot 0.4\,\text{km}} = 3656$$

The system implementation of the LL that will be able to meet these stringent requirements is explained in Section 4.5, dedicated to these technologies.

4.2.2 The need for hazard detection and avoidance

According to Strandmoe et al. [1999], the cumulative survivability rate for robotic planetary landers was only 80 % in 1999, where all of them were *blind* landing systems. Although forensic engineering could not trace all failures, it is likely that some missions were lost because they landed in hazardous areas. To increase the safety of robotic planetary landings, hazards should be known in advance or detected during landing.

Using sub-meter resolution imagery of the Moon's surface obtained by Lunar Reconnaissance Orbiter (LRO) in recent years, significant progress on automatic HD has been made. Hazards such as slopes, craters, boulders, or shadows can be identified to a

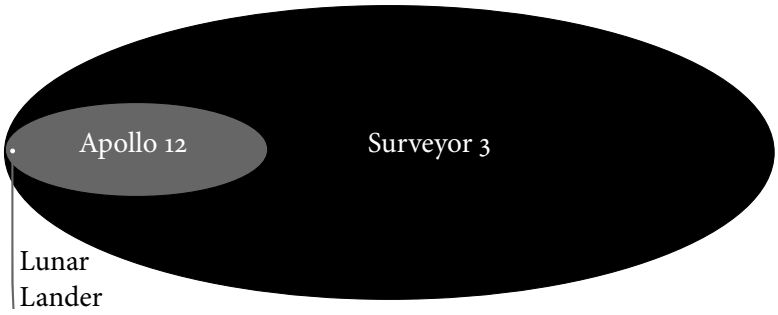


FIGURE 4.5 Illustration of the evolution in landing precision for Surveyor 3 (1967, 39 km by 15 km) [Ribarich 1968], Apollo 12 (1969, 13.3 km by 4.8 km, assuming a fully automatic landing without human intervention) [Bennett 1970], and the ESA Lunar Lander (planned for 2020 or later, 0.4 km by 0.4 km). To scale.

1 It has to be noted that Surveyor 3 was no *pinpoint* landing, flying only a gravity turn trajectory. In basic gravity-turn algorithms, the precise landing site is not specifiable.

certain extent from these orbital data. However, the resolution is insufficient to resolve smaller hazardous features like rocks, that could damage or tip over the lander during touchdown. To improve the probability of safe landing, > 98 % per LL requirement, one solution would be to employ automatic HDA technologies. Such systems shall be able to characterize the sensed surface in terms of hazardous features (HD) within the designated landing area in real time. Finally, the system must determine whether the chosen landing site is safe, and otherwise command a diversion to a safer one (safe-site selection). Huertas et al. [2007, p. 1] have determined "that this approach will reduce the probability of failed landing by at least a factor of 4 in any given terrain" for Mars. But also from a scientific point of view, the successful implementation of HDA systems would have a positive impact. For the example of MSL, Huertas et al. [2007, p. 11] show "that the vision system would allow access to roughly triple the fraction of the planet as a blind landing."

Being a technology demonstration mission, the ESA LL would have been an enabling precursor to fly these kind of scenarios robotically in the future. The HDA implementation on the LL is discussed in Section 4.5.

4.2.3 Relevance to the guidance system

With the discussion of these technologies, it is not obviously clear how this directly relates to landing guidance. After all, precision landing is a matter mostly of knowing the state very well, and HD is done by other instruments. Nevertheless, the preceding discussion is very relevant for the following reasons:

- Most importantly, vision-based technologies enforce several new constraints on guidance. First of all, it must of course be ensured that the landing site is permanently visible during landing. Guidance must constrain the lander's flight-path and attitude such that the instruments point at the landing site.
- Another constraint is that the attitude rate must not exceed certain limits. Otherwise vision-based data can smear and become unusable.
- Precision landing requires the guidance mode to be able to precisely target a position on the body. This is not the case for all algorithms.
- HDA implies that the spacecraft needs to divert to an alternative landing site if the original one is found to be dangerous. This calls for an algorithm that allows on-line re-planning of the trajectory.
- Diversions also mean that more propellant may be required as compared to the nominal trajectory. Thus, efficiency becomes even more important.
- Finally, the above-mentioned technologies are complex, and will need many computational resources. Thus, guidance will need to be particularly efficient when used in combination.

The need for precision landing and HDA thus directly impacts guidance. Having discussed the need for new technologies, the following section will now describe how GNC technologies are used to realize real D/L architectures, and highlights what characterizes their optimality.

4.3 OPTIMAL DESCENT AND LANDING ARCHITECTURES

As the previous section strongly stressed, optimality in terms of the required propellant mass is an essential feature of guidance algorithms. In recent years, researchers have made great strides in the optimality analysis of landing scenarios, and in particular it has been proven how the optimal control-profile should look like [Topcu et al. 2005; Açıkmeşe and Ploen 2007]. To be able to judge the guidance output to be derived in Chapter 6 and the quality of the scenario that is going to be designed (Chapter 9), it is thus important to study these results in this section.

To start with, it is essential to recognize that the D/L architecture and guidance are indeed one and the same thing. As a matter of fact, guidance generates a thrust-steering program that results in a landing trajectory.[2] The landing architecture is then defined as *the sequence of different guidance and control modes that is executed to perform the landing*. The designer must compose a D/L architecture that optimally suits a given mission, by scheduling maneuvers executed by corresponding guidance modes—which is not a trivial task given the host of available algorithms. To be capable of designing such a sequence, one must be aware of different options, which will be introduced here.

Assuming a start from an LLO parking orbit, the absolute theoretical optimum is shown in Figure 4.6 [Ball et al. 2007]. An impulsive shot is given opposite of the landing-site, such that a Hohmann-transfer trajectory occurs that intersects the Moon exactly at the landing site. At this point, another impulsive shot is given to eliminate the remaining velocity. Naturally, such an approach is not physically possible, because of the limited thrust and the irregular shape of the Moon. However, the most efficient way to get to the landing site from a circular LLO is indeed using a Hohmann transfer, which is hence applied in other architectures as well.[3]

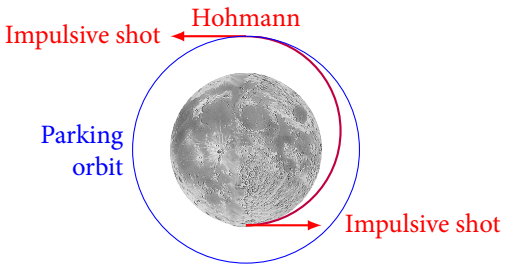


FIGURE 4.6 Theoretical optimum for a D/L architecture.

More realistic landing scenarios are shown in Figure 4.7. Consider first the straight blue line, which is a variation of the previous architecture. In this scenario, the spacecraft arrives at perilune near the landing site, where an impulsive shot is given. The spacecraft will then drop down vertically towards the site, and a second shot is given at the surface. This scenario is still unfeasible because of the assumption of impulsive shots, but theoretically allows for a precise targeting of the landing site with arbitrary topography. Meditch [1964] proofed the optimality of such a maneuver using optimal control theory.

2 An exception would be look-up table guidance, where a nominal trajectory was generated off-line and a controller is used for tracking this trajectory. Such modes are not considered in this study, as they are generally unfit for planetary landings and retargetings.

3 It is also possible to fly a direct approach, but this has not been done since Surveyor, because of other constraints in mission analysis, mainly involving issues with risk and constellation requirements.

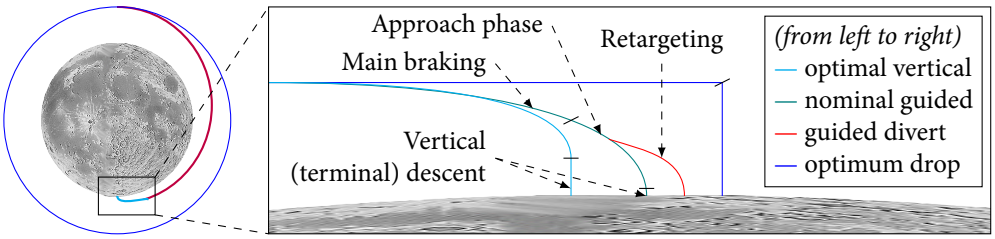


FIGURE 4.7 Overview of other other optimal D/L architectures. Not to scale.

The quest for the most efficient, yet feasible D/L architecture is thus to identify a scenario that gets close to this optimal one. The result is shown in cyan in the left-most graph in Figure 4.7. First, all horizontal velocity is eliminated by applying the maximum possible finite thrust in that direction. After that, the spacecraft will drop vertically towards the ground. Finally, the maximal finite thrust is engaged at the moment where it will eliminate all vertical velocity, such that the lander reaches 0 m/s at the surface. The thrust programme for the latter is given in [Meditch 1964].

This D/L architecture appears to be propellant-optimal and realistically feasible. The optimality has two physical reasons. First, gravity losses are reduced, because the thrust times are shortened by thrusting at maximum levels. And second, the propulsive maneuvers gain in efficiency, because they are executed at perilune. They are thus slightly more efficient because of the Oberth effect, which increases the propulsive efficiency due to higher orbital speed at lower altitudes.[4] Note that it would be even more efficient if there was no intermediate vertical drop at all, and the spacecraft would thrust maximally all the way fro PDI up to AG. However, such a scenario would likely not be practically feasible, because it would descent too quickly and leave no margin for error, because the controls are permanently saturated.

The drop architecture comes with problems on its own. Targeting a landing site precisely is very hard, because the powered descent-phase is kept as short as possible. This reduces the available time for any corrections and navigation. The sequence needs to be timed very precisely as well, and an excellent knowledge of the altitude above ground is critical for safety—otherwise the vehicle might crash. Hazard avoidance is not possible with a purely vertical descent either. This makes it an arguably more risky approach than others. Nevertheless, the Soviets flew their Ye-8-based missions in this way and were successful with it [Harvey 2006].

A real-life limitation is that engines can rarely ever be shut-off completely in between thrust arcs, so that the spacecraft does not freely drop, but the engines still operate at the minimum possible thrust. This yields a "bang-bang", or "max-min-max" type of control policy, depending on the capabilities of the propulsion system. It was proven by Topcu et al. [2005], in fact, that such a max-min-max profile is the minimum-

4 The Oberth effect is described by the following equation for the change in energy $\Delta\mathcal{E}$ per change in velocity Δv [Cornelisse et al. 1979]:

$$\Delta\mathcal{E} = \tfrac{1}{2}(\Delta v)^2 + \boldsymbol{v}\cdot(\Delta\boldsymbol{v})$$

The change in energy is thus the largest when applied in the direction of the vehicle's current velocity vector $\boldsymbol{v}$, and if v is largest. An intuitive explanation for this effect is that because energy equals force times distance, $\mathcal{E} = Fs$, the thrust acts over a longer distance s per interval of time if the vehicle flies faster.

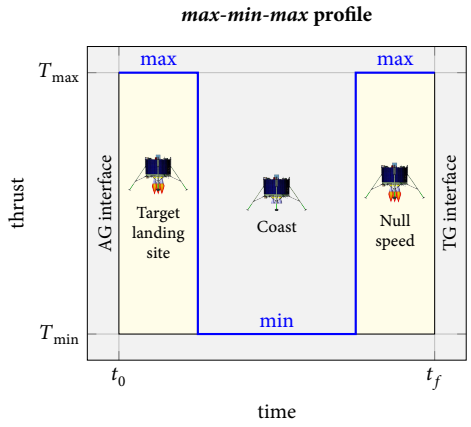


FIGURE 4.8 Optimal approach-phase thrust profile.

propellant solution (up to first-order conditions) for the most general case. The controls for this are shown in Figure 4.8. For the approach phase, the thrust arcs corresponding to a corrective burn that targets the landing site first, a coasting arc in between, and a final braking burn. Topcu et al. [2005] also show that degenerate cases of this general profile can occur that are still propellant optimal, namely max and min-max. An example for a max profile would be the direct Hohmann transfer to the landing site. However, the thrust-pointing direction or an actual guidance law is not shown in that paper—if it was, a globally optimal guidance law would have been analytically derived.

The designers of Apollo conceived the sequence shown in green and red in Figure 4.7 (middle curve), which was also adopted with slight modification for the LL (see Section 4.5.1 for a broad discussion). In this scenario, a main braking phase at maximum thrust first takes place, which is steered by an open-loop optimal guidance-mode that does not target the landing site yet. When the landing site comes into view, a different terminal-state-vector guidance mode takes over and steers the lander precisely to the landing site with a throttled, continuously decreasing thrust profile. Diversions for hazard avoidance are feasible. Targeting the landing site is easy, and the guidance law can generate new steering commands quickly.

If a max-min-max law could be used to precisely target a landing site, this would most likely be the most propellant-optimal solution that is feasible at the same time. An analytical solution for such an approach has not been found yet [Açıkmeşe and Ploen 2007]. However, the numerical algorithm designed in this report in fact satisfies this requirement.

4.4 GUIDANCE HERITAGE

In scope of the literature study that was conducted as preparation for this thesis, an extensive survey of guidance algorithms for Lunar landing has been conducted [Gerth and Mooij 2014]. More than 50 unique algorithms have been studied. Because this information is now in the public domain, there is no need to reproduce all of the results here. However, some general results and the outcome of the trade-off deserve mentioning, as this is what has led to the selected baseline guidance algorithm in this report.

On the highest level, three classes of guidance algorithms can be identified: open-loop guidance, explicit algorithms based on closed-form expressions, and algorithms rooted in numerical techniques. Open-loop algorithms are those that have no state (or output) feedback. An example would be a trajectory based on pre-computed guidance commands that is generated at mission control and then uploaded to a spacecraft. While this is a typical procedure for interplanetary spacecraft, this is by no means suitable for planetary landers. The main reasons are that errors cannot be corrected and retargetings would be impossible, so that overall the robustness is very low. Gravity-turn guidance is another example of open-loop guidance, because it only depends on the initial state and cannot fully constrain the final state. A gravity turn once initiated can not be corrected anymore [Citron et al. 1964].

The second class, explicit methods, make up the most-used guidance algorithms. In these methods, simplifying assumptions are made that yield analytical expressions for the acceleration command. Apollo guidance is one of them, as it is based on constraining the acceleration profile to be a quadratic polynomial of time [Cherry 1964]. Explicit guidance laws can also be derived using optimal-control theory [D'Souza 1997]. Such laws are obviously extremely computationally efficient, as they provide closed-form expressions. On the downside, though, it is generally not possible to include viewing-constraints of any form into account for HDA. Moreover, these algorithms are incapable of delivering the globally optimal bang-bang type of control solution. They are only suboptimal [Rea 2009].

Due to the increased computational power of on-board computers and recent developments in optimization theory, the GNC community has recently seen the advent of guidance algorithms rooted in numerical optimization. Such algorithms are capable of delivering globally-optimal solutions of bang-bang type, while at the same time constraining the trajectory. Particularly promising out of all options is *convex guidance*, that is based on convex optimization theory [Boyd and Vandenberghe 2004]. This has several significant advantages over other numerical algorithms, such as NLP [Açıkmeşe and Ploen 2007]:

- Convex optimization theory proofs that a well-posed convex problem (which this is) is *guaranteed* to converge.
- It can also be shown that the obtained solution will be the *global* optimum.
- There is a number of different, very efficient solvers for this kind of problem. The authors apply an interior-point method. These can be provided "with a deterministic stopping criteria [*sic.*], and with prescribed level of accuracy" [Açıkmeşe and Ploen 2007].
- Constraints and penalties can be imposed.

For these reasons, convex guidance has been selected for study in this report. Up to this point, the algorithm has only been studied in Mars-landing scenarios, which strongly differ from Lunar landings. Moreover, it is still underdeveloped in several areas, such as the use of gravity-field models and constraints. These will therefore be developed in this thesis, see Section 7.5 in particular. As mater of fact, this law is so promising that NASA is now performing hardware tests with terrestrial rocket-powered demonstration vehicles, and is considering to use it for the upcoming Mars 2020 lander [Williams 2013].

4.5 BASELINE LUNAR LANDER GNC FRAMEWORK

The precise- and safe-landing challenges will be overcome by applying four major functions, which are currently under development. These are:

Vision-based navigation A system that uses optical cameras to track features on an object's surface to estimate the translational state. This will enhance navigation accuracy and enable precise landings.

Hazard mapping The function that evaluates sensor data of the landing area and its vicinity, that associates a quantitative risk to each measurement point for a number of different hazards, and thus generates hazard maps from this information.

Autonomous piloting A function that will determine safe landing sites from the previously mapped hazards and commands to divert to a safer one if and only if the designated site is found to be unsafe.

Adaptive guidance Algorithms that allow to re-plan the trajectory on-line and steer the spacecraft to a new landing site if the piloting function commands this.

The integrated landing GNC architecture for the LL is conceptualized in Figure 4.9. Following the diagram from the top left, instruments will map the surface. Which hardware would have been used exactly remained undecided, but a camera for vision-based capabilities would have been part of the suite. Potential additions were a LIDAR for obtaining a digital elevation model (DEM) to evolute slope and roughness hazards, and (stereo-vision) cameras that could fulfill the same task. The arrow pointing down to navigation indicates the first major function, namely vision-based navigation. The instruments are also used by hazard mapping, which creates risk maps from their input. The HD function will then create an overall risk map, which also contains reachability and propellant-need estimates from a simplified guidance function. Once piloting has evaluated these maps, it either commands a retargeting, or confirms the current landing site. Being fed back to guidance, the loop is then iterated until low gate (LG). Further details can be found in [De Rosa et al. 2011].

4.5.1 Reference scenario

To get a better picture in which conditions the to-be-developed guidance algorithm has to work, a reference D/L scenario is defined in this section. The scenario for the entire landing phase is shown in Figure 4.10. Before the discussion continues, it is essential to establish some terms that also appear in the chart:

High gate (HG) The position vector associated to the point in time that the landing site comes into the field of view of the camera. (High gate may be encountered before approach gate.)

Approach gate (AG) The position vector associated to the point in time after main braking that the engines are throttled back and the approach-phase guidance is initiated.

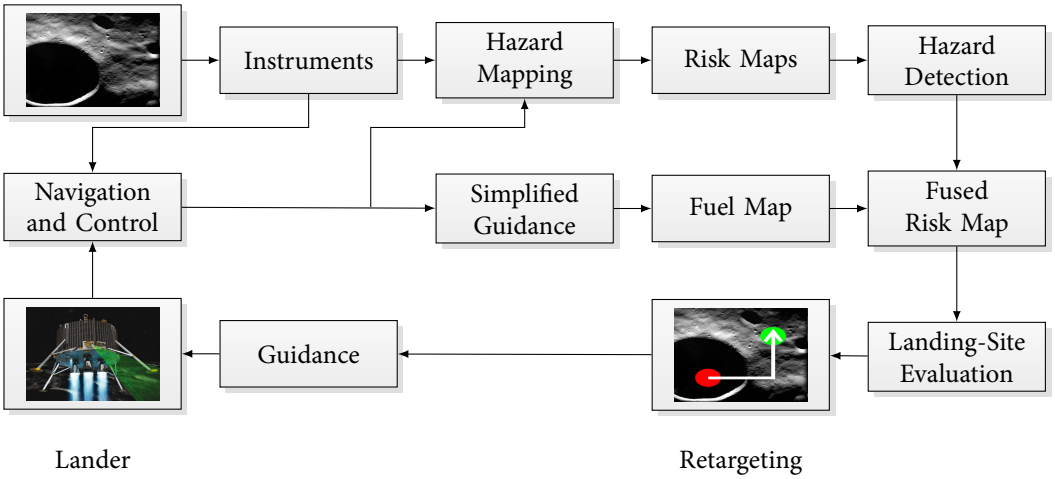


FIGURE 4.9 Illustration of the top-level GNC system design to be used for active HDA on the LL.

Low gate (LG) The position vector associated to the point in time of the where the visual to the landing site is lost.[5]

Terminal gate (TG) The position vector associated to the point in time where the terminal (vertical) descent is initiated.

Touchdown (TD) The position vector associated to the point in time where all of the vehicle's landing legs come to a rest on the central body's surface.

The individual phases will be briefly explained in the following.

4.5.1.1 *Low Lunar Orbit (LLO)*

The reference scenario starts in a 100 km LLO. The lander LL would have stayed in this orbit for one month, given that no contingencies occur. This would have been used for systems check-out and to wait until the orbit lines up with the landing site. Before the landing is started, ground-based navigation-data obtained from tracking using ESA's deep-space antennae would have been uplinked to the spacecraft to initialize the navigation filter. This provides an accurate state to the spacecraft, which helps with precise navigation.

4.5.1.2 *Descent*

The DOI marks the start of the descent phase. This is a short burn which brings the lander on a Hohmann transfer trajectory with a perilune of 10 km. During the coasting phase up to perilune absolute vision-based navigation is applied to increase precision of the inertial state.

At perilune, PDI begins, which means that all engines are turned on for the main-braking phase. During this flight phase most of the orbital velocity is eliminated, bringing it down from 1630 m/s to 70 m/s. About 500 km are covered in this leg.

During or shortly after the main braking, the landing site will come into the FOV of the HD camera. This marks HG.

5 Sometimes low-gate is also defined as the point in time where visual to the landing site is lost.

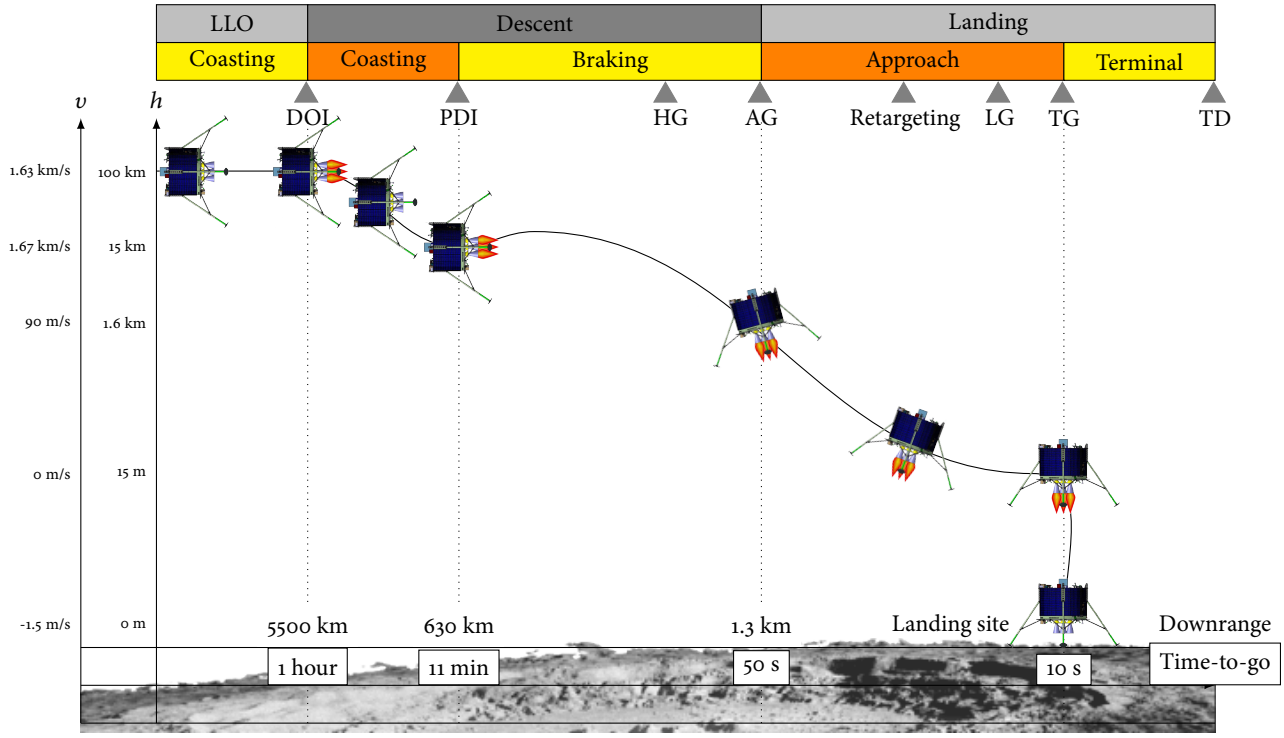


FIGURE 4.10 Reference landing scenario for the ESA LL. Adapted from [Delaune et al. 2010].

4.5.1.3 *Landing*

When the engines start throttling back AG is reached, which starts the targeting descent towards the landing site. During this phase the HDA system is active, and will command the vehicle either towards the preselected landing site, or to an actively selected new one. The spacecraft will perform camera-based navigation relative to the landing site, complemented with IMU and altimeter measurements during this phase.

At some point the lander will come close to the chosen landing site, and direct visual to the landing site can be lost. This marks LG if no retargeting was necessary. The lander will then rely only on its IMU for navigation. Once it is estimated that the lander is above the landing site, all horizontal velocity will be eliminated and the lander will be in vertical orientation. This point at 15 m altitude marks the TG. From TG onwards the LL will fly vertically to the surface at a constant velocity. Once contact with the surface is established, all engines are shut-down and landing has been completed.

This completes the reference scenario, which sets the constraints for the guidance and control system to be developed.

4.6 REFINED PROBLEM-FORMULATION

Now that the reference scenario has been defined, the problem formulation can be narrowed down. Açıkmeşe and Ploen [2007, p. 1354] phrase the guidance problem more appropriately for a pinpoint-landing mission:

> *The powered descent guidance problem for pinpoint landing is defined as finding the fuel optimal trajectory that takes a lander with a given initial state (position and velocity) to a prescribed final state in a uniform gravity-field, with magnitude constraints on the available net thrust, and various other constraints.*

By "uniform gravity-field" the authors mean a constant-magnitude, flat-Moon model.

In the lines of Açıkmeşe and Ploen [2007], this can be written mathematically as:

▶ PROBLEM 2 Refined problem formulation with constraints and precision landing requirements.

$$
\begin{aligned}
&\text{minimize} \int_0^{t_f} \sqrt{\boldsymbol{T}(t)^\intercal \boldsymbol{T}(t)}\, \mathrm{d}t \\
&\text{subject to} \\
&\textit{equations of motion} \\
&\dot{m}(t) = -\alpha \|\boldsymbol{T}(t)\|, \quad 0 < T_l \leq \|\boldsymbol{T}(t)\| \leq T_h, \quad r_z \geq 0, \quad \zeta_1 \geq \beta \geq \zeta_2 \\
&m(0) = m_0, \quad \boldsymbol{r}(0) = \boldsymbol{r}_0, \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0, \quad \boldsymbol{r}(t_f) = \boldsymbol{r}_f, \quad \dot{\boldsymbol{r}}(t_f) = \dot{\boldsymbol{r}}_f
\end{aligned}
\tag{4.5}
$$

where $\dot{m}(t)$ is the mass flow, α is the thrust-to-mass flow conversion-factor (a real constant), T_l and T_h the lower and upper thrust bounds (positive real numbers), r_z the vertical component of the position vector $\boldsymbol{r}(t)$, ζ_1 and ζ_2 are real numbers, and β is the viewing angle between the landing site and the camera.

This imposes constraints on the magnitude of the thrust, disallows subterranean trajectories, and can enforce that the landing site remains in the camera's FOV.

Chapter 5

CONVEX OPTIMIZATION THEORY

> An algorithm must be seen to be believed.
>
> *Donald Knuth*

Optimality plays a key role not only in spacecraft guidance, but in spacecraft design in general. While one could imagine guidance algorithms that are not explicitly optimal, *guidance for planetary landing* most often means finding *optimal trajectories* connecting initial to final conditions. The quantity to be minimized is typically the time integral of thrust, to limit the landing system's mass by decreasing the propellant flow. It is this section's goal to provide the background in the optimization theory needed to understand the guidance algorithm to be developed. For a general overview of trajectory optimization methods see [Betts 1998] or [Conway 2010].

This chapter on optimization theory is organized as follows. Some basic definitions on optimization theory are discussed in Section 5.1. What exactly the term "convexity" entails is described in Section 5.2. The subject of Section 5.3 is convex sets. Finally, Section 5.4 treats the convexity of functions.

5.1 PRELIMINARY CONCEPTS

Mathematical optimization in general is the task of finding the subset of extreme values, either maxima or minima, in a given set. The purpose of this section is to lay the foundations for the remainder of this chapter by explaining and defining the concepts touched upon in this definition, such as: "What is an optimum?", or "How are optimization problems formulated?" The discussion and notation here will closely follow the example of the excellent, but mathematically challenging, reference on convex optimization theory by Boyd and Vandenberghe [2004], and the definitive text on optimal control theory by Bryson and Ho [1975].

To do an optimization, it needs to be written in a special format that precisely describes the task to be executed. This mathematical formalization is called an *optimization problem*. One usually writes them in *standard form*, that reads:

$$\begin{aligned} &\text{minimize} && f_0(\boldsymbol{x}) \\ &\text{subject to} && g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \dots, m \\ &&& h_j(\boldsymbol{x}) = 0, \quad j = 1, \dots, p \end{aligned} \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ holds the *optimization variables* and the function $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*.[1] This formulation thus describes the problem of finding an $\boldsymbol{x}$ that minimizes $f_0(\boldsymbol{x})$, that satisfies all the conditions listed under "subject to". These conditions are called the *constraints*, where $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \dots, m$ are the *inequality*

1 The objective function is also often referred to as *cost function* or *performance index*.

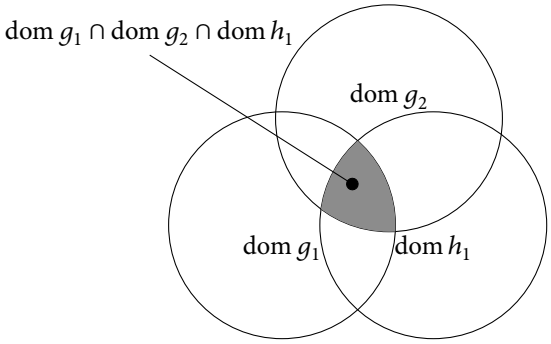


FIGURE 5.1 Domain of an arbitrary optimization problem with two inequality constraints and one equality constraint ($m = 2$, $p = 1$). The domain of the problem is the intersection of all sets of the domains of the constraints.

constraint functions, and $h_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1 \dots p$ are the *equality constraint functions*. Thus, the problem is subject to n variables and $m+p$ constraints. All problems in this report will be minimization problems. A maximization problem can simply be obtained by negating the objective function, $-f_0$. A simplified and abbreviated example for a guidance optimization-problem formulation is given below.

☑ EXAMPLE 5.1 Example of a bare-bones guidance optimization-problem with constraints on the thrust magnitude. The EOMs are considered through an equality constraint.

minimize

$f_0 : \mathbb{R}^n \to \mathbb{R}$

$\boldsymbol{T}(t) \mapsto \int \sqrt{\boldsymbol{T}(t)^\intercal \boldsymbol{T}(t)} \mathrm{d}t$ (*n. b.*, $\boldsymbol{x} = \boldsymbol{T}(t)$)

subject to

$g_1 : \mathbb{R}^n \to \mathbb{R}$ (lower thrust bound)

$\boldsymbol{T}(t) \mapsto T_{\min} - \|\boldsymbol{T}\| \leq 0$

$g_2 : \mathbb{R}^n \to \mathbb{R}$ (upper thrust bound)

$\boldsymbol{T}(t) \mapsto \|\boldsymbol{T}\| - T_{\max} \leq 0$

$h_1 : \mathbb{R}^n \to \mathbb{R}^n$ (EOMs)

$\boldsymbol{T}(t) \mapsto \boldsymbol{g} + \boldsymbol{T}(t)/m(t) - \ddot{\boldsymbol{r}}(t) = \boldsymbol{0}$

The constraints of the optimization problem restrict the possible values that $\boldsymbol{x}$ may attain, in other words: it constrains the search space. This *domain* of the optimization problem is defined as

$$\mathcal{D} = \bigcap_{i=1}^{m} \mathrm{dom}\, g_i \cap \bigcap_{i=1}^{p} \mathrm{dom}\, h_i \qquad [5.2]$$

This means that only values that are allowed for all constraints are admissible for the optimization problem. An example is shown in Figure 5.1.

An *optimal solution* found through the optimization is one for the vector $\boldsymbol{x}^*$, if it has the smallest objective value of all possible vectors in the set. Mathematically speaking, $\boldsymbol{x}^*$ is the optimum if for any vector $\boldsymbol{z}$:

$$f_0(\boldsymbol{z}) \geq f_0(\boldsymbol{x}^*) \tag{5.3}$$

Or, more rigorously, the lowest cost is:

$$p^* = \inf\{f_0(\boldsymbol{x}) \mid g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, h_i(\boldsymbol{x}) = 0, i = 1, \dots, p\} \tag{5.4}$$

with inf the infimum, indicating the smallest element (or subset) in the set. The set of all optimal points is then:

$$\mathcal{X}^* = \{\boldsymbol{x} \mid g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, h_i(\boldsymbol{x}) = 0, i = 1, \dots, p, f_0(\boldsymbol{x}) = p^*\} \tag{5.5}$$

The types of functions that specify the objective f_0 and the constraints g_i and h_i determine the class of an optimization problem. Centered around each of these classes, many different optimization (or solution) methods for finding the minimum exist. Under certain circumstances, an analytical optimum can be found. One of the most-used tools for this is *optimal control theory*. A second special case occurs if all functions and the search space $\mathcal{D}$ are convex. Even though problems are often not analytically solvable and nonlinear in such cases, very efficient and robust methods for finding solutions exist in the field of *convex optimization theory*. This forms the theoretical basis for the guidance algorithms developed in this thesis and is thus is explained in the following section.

5.2 GENERAL CONVEXITY

Convex optimization problems are a family of optimization problems that possess some extraordinarily beneficial properties. The past decade has seen much progress in the field of convex optimization. It is now becoming more and more well known that many problems can be formulated in a convex fashion, and that "[like] least-squares or linear programming, there are very effective algorithms that can reliably and effectively solve even large convex problems" [Boyd and Vandenberghe 2004, p. 4]. The "surprisingly difficult to solve" general optimization problem thus becomes surprisingly "simple" and effective to solve [p. 3]. It was under this assumption that Açıkmeşe and Ploen [2007] set out for their work to find a soft-landing guidance-algorithm based on convex optimization theory—and succeeded. To understand their approach, the basics of convex optimization are introduced here.

As the name suggests, convex optimization problems are those in which the objective and constraint functions are convex, meaning that they satisfy the inequality:

$$g_i(\alpha\boldsymbol{x} + \beta\boldsymbol{y}) \leq \alpha g_i(\boldsymbol{x}) + \beta g_i(\boldsymbol{y}) \tag{5.6}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and for all $\alpha, \beta \in \mathbb{R}$, with $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$. This may be interpreted as follows: a function f is convex if the function of the weighted average is smaller or equal to the weighted average of the function.

☑ EXAMPLE 5.2 Simple numerical example illustrating the property of convexity.

A simple parabola is given by $f(x) = x^2$, $f: \mathbb{R} \rightarrow \mathbb{R}$. For simplicity, choose $\alpha = \beta = 0.5$, fulfilling the requirement $\alpha + \beta = 1$.

$$f(0.5 \cdot 4 + 0.5 \cdot 8) = 36$$
$$0.5f(4) + 0.5f(8) = 40$$
$$f(0.5 \cdot 4 + 0.5 \cdot 8) < 0.5f(4) + 0.5f(8)$$

For comparison, a *linear program* would be one satisfying:

$$g_i(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) = \alpha g_i(\boldsymbol{x}) + \beta g_i(\boldsymbol{y}) \quad [5.7]$$

with *arbitrary* α, β. Convex problems are thus on the one hand a generalization of linear programs, because the equality is replaced by an *in*equality, and on the other hand a restriction, because α and β are not free.

In the following, convex sets are discussed to more detail in Section 5.3. Details on the convexity of functions are then given in Section 5.4. Ways to formulate convex optimization problems are the subject matter of the subsequent Section 5.5. The theory explained throughout these sections is then applied to the selected convex guidance mode in Chapter 6. In addition to [Boyd and Vandenberghe 2004], this section is based on the overview by [Hindi 2004].

5.3 CONVEX SETS

Although it would go much too far to give a complete introduction to convex optimization here, a few fundamentals need to be established. At the very core of the theory are *convex sets*, which are introduced in this section.

First define a *line* between two points, with $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$. Then

$$\boldsymbol{y} = \theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \quad [5.8]$$

is a line passing through the two points if $\theta \in \mathbb{R}$. The interval $0 \leq \theta \leq 1$ describes the *line segment* between the points. This can be used to define affine and convex sets.

An *affine set* C is one where the line that is constructed through any two points in the set lies in the set again. This means that $C \subseteq \mathbb{R}^n$ is affine, if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and $\theta \in \mathbb{R}$ it holds that $\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \in C$. The same holds for affine combinations, such that $\theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k$, with $\theta_1 + \cdots + \theta_k = 1$, also lies in C.

Convex sets are closely related to affine sets, but restrict the parameter θ to be between 0 and 1—whereas θ may take on *any* value in affine sets, as long as $\sum \theta_i = 1$. Thus, a convex set must contain the line segments between any two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in the set. A set $C \subseteq \mathbb{R}^n$ is convex if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and any $\theta, 0 \leq \theta \leq 1$, it holds

$$\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \in C \quad [5.9]$$

Boyd and Vandenberghe [2004, p. 24] explain this property intuitively: "if every point in the set can be seen by every other point in the set, along an unobstructed straight

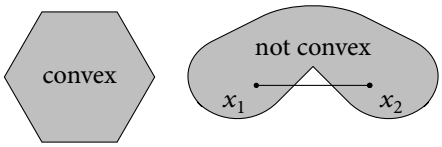
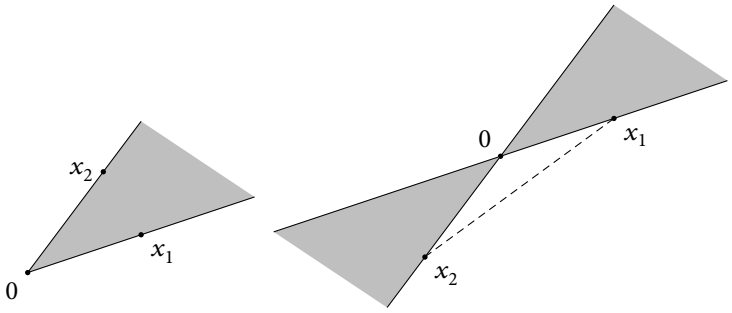


FIGURE 5.2 Examples of a convex and a non-convex set. The set on the right is not convex because the line connecting the two points passes through its exterior.



(a) Convex. $\theta_1, \theta_2 \in \mathbb{R}_+$ **(b)** Not convex. $\theta_1, \theta_2 \in \mathbb{R}$

FIGURE 5.3 Examples of cones.

path" that lies in the set as well, then it is convex. This is the case if the set is always bulging outwards, which is illustrated in Figure 5.2.

A special type of convex sets are *convex cones*, which are applied by Açıkmeşe and Ploen [2007] in the formulation of the SOCP. A general (not necessarily convex) cone is a set for which it must hold that if $\boldsymbol{x} \in C$ and $\theta \geq 0$, it is the case that $\theta \boldsymbol{x} \in C$. Convex cones are those for which the convexity condition holds but $\sum \theta_i$ is not restricted to unity, such that $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and $\theta_1, \theta_2 \geq 0$, and thus

$$\theta_1 \boldsymbol{x}_1 + \theta_2 \boldsymbol{x}_2 \in C \tag{5.10}$$

The cone is thus the set spanned by an origin and the lines going through $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, as drawn in Figure 5.3(a). Clearly, any two points in this set can be connected without the line leaving the set, intuitively explaining the convexity of cones. An example for a cone that is not convex is shown in Figure 5.3(b).

Finally, the second-order cone, a special case of convex cones, can be introduced. The second-order cone corresponding to the Euclidean norm is described by

$$C = \{(\boldsymbol{x}, t) \mid \|\boldsymbol{x}\| \leq t\} \subseteq \mathbb{R}^{n+1} \tag{5.11}$$

$$= \left\{ \begin{pmatrix} \boldsymbol{x} \\ t \end{pmatrix} \middle| \begin{pmatrix} \boldsymbol{x} \\ t \end{pmatrix}^{\mathsf{T}} \begin{bmatrix} \mathsf{I}_{n \times n} & 0 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ t \end{pmatrix} \leq 0, t \geq 0 \right\} \tag{5.12}$$

The second-order cone in $\mathbb{R}^3$, sometimes referred to as "ice-cream cone" for obvious reasons, is shown in Figure 5.4.

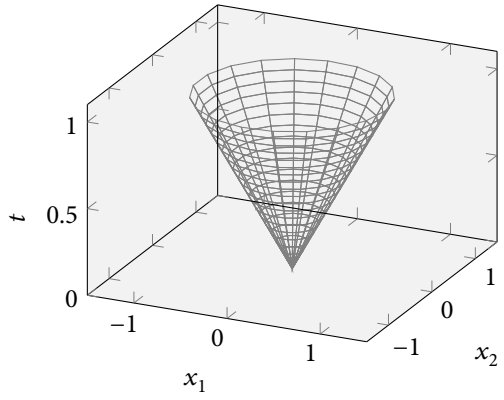


FIGURE 5.4 Boundary of the second-order cone in $\mathbb{R}^3$. The cone is filled.

5.4 CONVEX FUNCTIONS

Convex optimization problems require all functions involved to be convex. In many cases, it is far from obvious to recognize whether a function is convex or not, and sophisticated techniques for checking this are available. Fortunately, Açıkmeşe and Ploen [2007] already demonstrated this for the convex guidance algorithm, but this still will be verified in Chapter 6. The basic definitions of function convexity need to be known either way to motivate why the algorithm performs so well, and these are thus explained here. Again, all definitions given here are taken from [Boyd and Vandenberghe 2004].

A function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is *convex* under three conditions: (i) dom f is a convex set; (ii) All $\boldsymbol{x}, \boldsymbol{y} \in$ dom f; and (iii) $0 \leq \theta \leq 1$. It must then hold:

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}). \tag{5.13}$$

This inequality allows for a geometric interpretation, which is shown in Figure 5.5. For a function to be convex, all lines connecting any two points of a function must lie above that function's graph.

Convex functions fulfill first- and second-order necessary conditions. The first-order condition reads as follows. If f is differentiable, then f is convex if and only if dom f is convex and

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x_0}) + \nabla f(\boldsymbol{x_0})^{\mathsf{T}} (\boldsymbol{x} - \boldsymbol{x_0}) \tag{5.14}$$

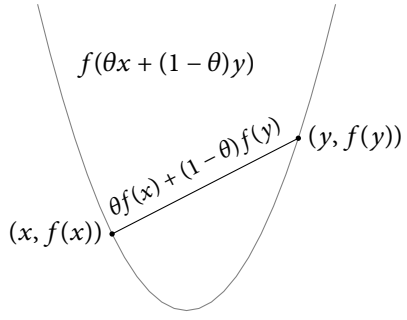


FIGURE 5.5 Graph of a convex function. The line between any two points lies above the graph.

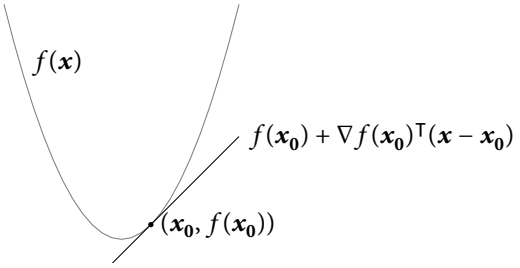


FIGURE 5.6 The first-order condition for convex functions.

holds for all $\boldsymbol{x}, \boldsymbol{y} \in \text{dom } f$. The right-hand side of the inequality is not only an affine function, but also a first-order Taylor approximation of the function $f(x)$. This is plotted in Figure 5.6. In fact, the inequality states that this Taylor approximation is a *global underestimator* if the function is convex: the approximated line *always lies below the graph* of the convex function. Thus, from local information, global information is derived. Without going into further detail, in the words of Boyd and Vandenberghe [2004, p. 69], "[this] is perhaps the most important property of convex functions, and explains some of the remarkable properties of [...] convex optimization problems.".

The second-order condition is rather intuitive, namely that the Hessian must be positive semidefinite. Assume that f is twice differentiable, thus the Hessian $\nabla^2 f$ exists at each point in dom f. Then f is convex if and only if dom f is convex and $\nabla^2 f$ is positive semidefinite: for all $\boldsymbol{x} \in \text{dom } f$

$$\nabla^2 f \succeq \mathbf{0}. \tag{5.15}$$

This simply means that the derivative must be nondecreasing—it must have upward curvature at $\boldsymbol{x}$.

Although these tests are based on differentiability, this is not a general requirement for convex functions. There are also non-differentiable functions that can be convex. One such example is $f(x) = |x|$. This function has a V-shaped graph, and a kink at $x = 0$. Whereas the function is not differentiable at $x = 0$, the function is still convex.

5.5 CONVEX OPTIMIZATION PROBLEMS AND SECOND-ORDER CONE PROGRAMS

The general form of optimization problems was defined in Equation [5.1]. There it was claimed that convex optimization problems possess some special properties, that make their solution particularly simple, efficient, and robust. This section tries to answer why this is the case on the highest level, and introduce the SOCP which is used for the convex guidance-mode.

Convex optimization problems are those of the form

$$\begin{aligned} &\text{minimize } f_0(\boldsymbol{x}) \\ &\text{subject to } g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \dots, m \\ &\qquad\qquad \boldsymbol{a}_j{}^\mathsf{T}\boldsymbol{x} = b_j, \quad j = 1, \dots, p \end{aligned} \tag{5.16}$$

in which $f_0, g_1 \ldots, g_m$ are convex functions. Compared to the general optimization problem Equation [5.1], three further requirements must be fulfilled for convex optimization problems:

- The objective function f_0 must be *convex*; and
- The inequality constraint functions must be *convex*; and
- The equality constraint functions $h_i(\boldsymbol{x}) = \boldsymbol{a}_i{}^\mathsf{T}\boldsymbol{x} - b_i$ must be *affine*.

As a consequence of these requirements, the feasible set of the problem (its domain or solution space)

$$\mathcal{D} = \bigcap_{i=0}^{m} \operatorname{dom} g_i \qquad [5.17]$$

is also a convex set (the intersection of convex sets is a convex set again).

A special class of convex optimization problems are SOCPs. These are optimization problems, which fulfill some additional requirements:

- The objective function must be affine (linear); and
- The inequality constraints are convex cones; and
- The equality constraints are affine.

The form reads as follows:

$$\begin{aligned} &\text{minimize } f^\mathsf{T}\boldsymbol{x} \\ &\text{subject to } \|A_i\boldsymbol{x} + \boldsymbol{b_i}\| \leq \boldsymbol{c}_i{}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}_i, i = 1, \ldots, m \\ &\qquad\qquad F\boldsymbol{x} = \boldsymbol{g} \end{aligned} \qquad [5.18]$$

where $A \in \mathbb{R}^{k \times n}$ and $F \in \mathbb{R}^{b \times n}$. To understand why this is called a "cone problem", return to Equation [5.11] and Figure 5.4 on Page 54. A set was described by a second-order cone in $\mathbb{R}^{n+1}$ in Equation [5.11] as:

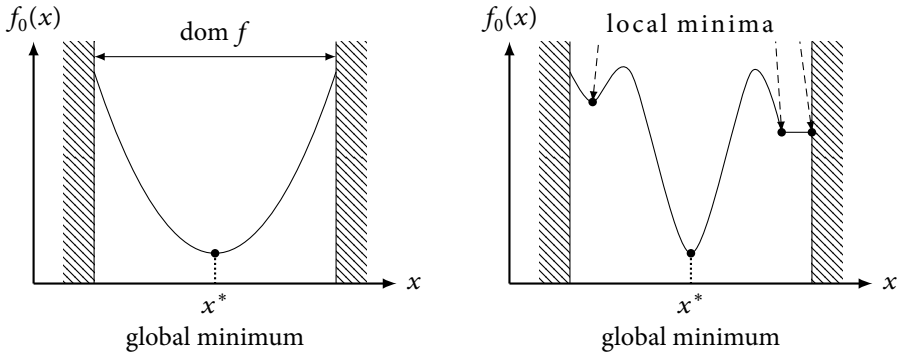
$$C = \{(\boldsymbol{x}, t) \mid \|\boldsymbol{x}\| \leq t\} \subseteq \mathbb{R}^{n+1}$$

By analogy it can be seen that the inequality constraint in Equation [5.18] also describes a cone:

$$C = \{(A_i\boldsymbol{x} + \boldsymbol{b_i}, \boldsymbol{c}_i{}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}_i) \mid \|A_i\boldsymbol{x} + \boldsymbol{b_i}\| \leq \boldsymbol{c}_i{}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}_i, i = 1, \ldots, m\} \subseteq \mathbb{R}^{k+1} \qquad [5.19]$$

Note that SOCPs are equivalent to other types of optimization problems, such as quadratically-constrained quadratic-programs (by setting $c_i = 0$ and squaring the constraints) or linear programs (by setting $A_i = 0$).

Although the following analogy has to be considered with caution, a simplified explanation for why convex problems lead to global minima and are easily solvable is shown in Figure 5.7. If a function is convex over its domain dom f, then there can



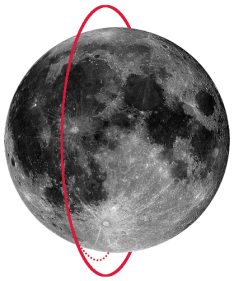
(a) Convex function **(b)** Non-convex function

FIGURE 5.7 Optimal solutions in convex and non-convex functions. Convex functions can only have one single, globally optimal solution. Note that in (b) the entire straight line is a local minimum.

necessarily be *only one global optimum*. It is thus impossible to get stuck on a local optimum. Moreover, the optima can be found rather easily because the negative gradient will always lead to the optimum.

Convexity does not necessarily mean that a function looks like the parabola shown here. Convex sets can be harder or even impossible to visualise. Nevertheless, imagining this analogy helps the intuitive understanding of the problem.

SOCPS possess some remarkable properties which make their application very beneficial. Hindi [2004, p. 10] says that "[…] [if] a real problem can be cast into one of these forms [*including SOCPS*], then it can be considered as essentially solved." The reason for this is that very efficient solvers have emerged in this field lately. The solution methods for this have only been discovered in the mid 1990s, and applications of SOCPs have spread rapidly since then. The speed of solving SOCPS is now almost as fast as solving linear programs, and the solvers are thus approaching a mature technology. Although far from obvious at first, the guidance problem can be posed as an SOCP, which is described in the Chapter 6.



Chapter 6

CONVEX GUIDANCE

> "The synthesis of guidance laws for rocket-propelled flight has been called a multi-dimensional, nonlinear, boundary-valued, variational problem. It is worth analyzing this formidable-sounding description. Each of these adjectives describes an aspect of the problem which makes its solution far from trivial."
>
> *George W. Cherry (Apollo Guidance Engineer, 1964)*

In this chapter, the convex guidance mode by Açıkmeşe and Ploen [2007] that was selected for implementation is derived step-by-step. The starting point is the constrained formulation of Problem 2, Page 48. This formulation states the TPBVP in continuous time—and is nonconvex, rendering it unsuitable to convex-optimization solvers. However, nothing is lost: with a few smart mathematical manipulations the problem can be "convexified". Arriving at a convex, solver-compatible problem calls for the following steps:

1. *Convexify the problem.* In the original form, nonconvex control-constraints are present. This needs to be resolved through *lossless convexification*.
2. *Change the variables.* The convexification introduces another nonconvexity, and the EOMs are also not convex by nature. These issues can be circumvented by changing the problem variables and rewriting the problem.
3. *Discretize.* The solution of any general convex optimization is a batch process that leads to the output of an optimal, *finite dimensional* decision variable. However, Problem 2 is still written in continuous time, and is thus a constrained infinite-time optimal-control problem (CITOC). To make it solvable using convex-optimization techniques, it needs to be discretized to a constrained finite-time optimal-control problem (CFTOC).
4. *Formulate the problem.* Once the problem has been convexified and discretized, it has to be cast into a final, solver-readable form.

The following sections will shed light on these steps, starting with the convexification in Section 6.1. The change of variables is done in Section 6.2. Section 6.3 discusses the discretization of the problem. The guidance problem is then formulated in Section 6.4. A method to determine the optimal time-of-flight is then described in Section 6.5. Finally, the verification of the algorithms is the subject of Section 6.6.

6.1 LOSSLESS CONVEXIFICATION

It has previously been pointed out that guidance is a constrained two-point boundary-value optimization problem. The mathematical formulation of this problem is repeated here for further reference, where the EOMs are now filled in:

▶ PROBLEM 3 Constrained terminal-state-vector propellant-optimization problem subject to constant gravity.

$$
\begin{aligned}
&\text{minimize} \int_0^{t_f} \sqrt{\boldsymbol{T}(t)^\intercal \boldsymbol{T}(t)}\, \mathrm{d}t \\
&\text{subject to} \\
&\ddot{\boldsymbol{r}}(t) = \boldsymbol{T}(t)/m(t) + \boldsymbol{g}, \quad r_z(t) > 0 \\
&\dot{m}(t) = -\alpha \|\boldsymbol{T}(t)\|, \quad 0 < T_l \leq \|\boldsymbol{T}(t)\| \leq T_h, \quad \zeta_1 \geq \beta \geq \zeta_2 \\
&m(0) = m_0, \quad \boldsymbol{r}(0) = \boldsymbol{r}_0, \quad \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0, \quad \boldsymbol{r}(t_f) = \boldsymbol{r}_f, \quad \dot{\boldsymbol{r}}(t_f) = \dot{\boldsymbol{r}}_f \\
&\forall\, t \in (0, t_f)
\end{aligned}
$$

As it turns out, this problem is inherently nonconvex and can thus not be solved using convex optimization techniques in this form. The nonconvexities are caused by the lower thrust-bound, which will be resolved in the sequel, and the EOMS, which will be overcome by the discretization and a change of variables.

The thrust is limited as follows:[1]

$$0 < T_l \leq \|\boldsymbol{T}(t)\| \leq T_h \tag{6.1}$$

The nonconvexity due to the lower bound is illustrated in Figure 6.1. Without loss of generality, this figure has been constructed under the assumption that the z-component of the thrust is nonexistent.[2] The annular shape of the control constraints is easily recognized from Equation [6.1] because

$$\sup\left\{\|\boldsymbol{T}\| \mid T_z = 0, 0 < T_l \leq \|\boldsymbol{T}\| \leq T_h\right\} = \left\{\|\boldsymbol{T}\| \mid \sqrt{T_x^2 + T_y^2} = T_h\right\} \tag{6.2}$$

$$\inf\left\{\|\boldsymbol{T}\| \mid T_z = 0, 0 < T_l \leq \|\boldsymbol{T}\| \leq T_h\right\} = \left\{\|\boldsymbol{T}\| \mid \sqrt{T_x^2 + T_y^2} = T_l\right\} \tag{6.3}$$

which evidently is identical to the circle equation.

To understand why this constraint is not convex, return to the definition of convexity Equation [5.9] on Page 52, repeated here for convenience. A set $C \subseteq \mathbb{R}^n$ is convex if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in C$ and any $\theta \in (0, 1)$ it holds

$$\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \in C$$

This means that any two points in a set need to be connectible by a line, where the entire line segment between the points needs to stay within the set. Because the set in Figure 6.1 is annulus shaped, line segments can pass through its empty interior: this set is thus *not* convex.[3]

1 The lower thrust bound $T_l > 0$ is larger than zero because virtually all engines available today are not hot restartable within the time frame of a planetary landing. A thrust of $\|T\| = 0$ would mean an engine cut-off and must thus be avoided by throttling down to the lowest possible setting.

2 The z-component is only ignored to be able to visualize the sets, which would become impossible for later illustrations that would be in $\mathbb{R}^4$ (Figure 6.2(b)). The drawing in Figure 6.1 could be imagined as a partially hollow sphere, if T_z was not equal to zero.

3 Note that setting only an upper limit, as in $0 \leq \|\boldsymbol{T}\| \leq T_h$, would result in a convex control constraint.

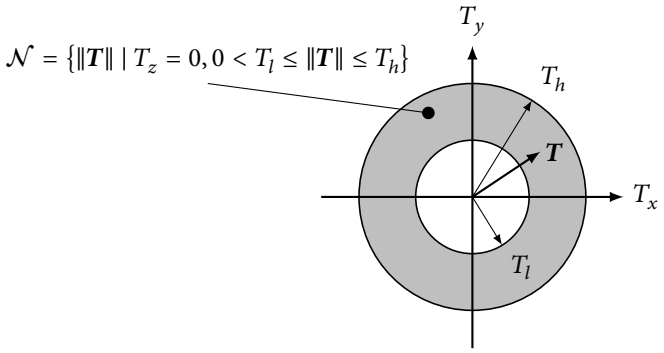


FIGURE 6.1 Illustration of the set described by the original thrust-magnitude control-constraint Equation [6.1]. Lines can be constructed passing through the center of the annulus: the set is nonconvex.

The major innovation of [Açıkmeşe and Ploen 2007] was to convexify this control constraint. This approach has since been termed "lossless convexification", and is further detailed in [Açıkmeşe and Blackmore 2011]. The process is surprisingly simple but effective. The central idea is to introduce a scalar slack variable Γ that relaxes the control constraint, because it maps the controls into an *affine space*, which is by definition convex, see Figure 6.2(a). One further retains the convex upper bound on the control constraint. Mathematically, one thus converts the constraint Equation [6.1], $0 < T_l \leq \|\boldsymbol{T}\| \leq T_h$, to the two constraints:

$$0 < T_l \leq \Gamma \leq T_h \tag{6.4}$$

$$\|\boldsymbol{T}\| \leq \Gamma \tag{6.5}$$

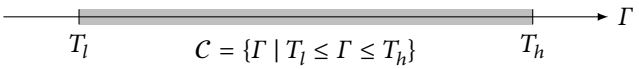
which are both convex.

Although the constraints in Equations [6.4] and [6.5] above are now convex, it is not immediately obvious why a solution satisfying these would also be a solution satisfying the original constraint $0 < T_l \leq \|\boldsymbol{T}\| \leq T_h$ (Equation [6.1]). As said before, the control space is indeed now *relaxed*. This means that the control space defined by Equation [6.1] is now a subset of that defined by Equations [6.4] and [6.5], or:

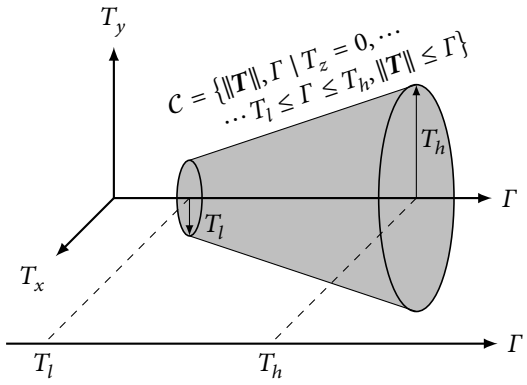
$$\{\boldsymbol{T} \mid 0 < T_l \leq \|\boldsymbol{T}\| \leq T_h\} \subset \{\boldsymbol{T}, \Gamma \mid 0 \leq \Gamma, 0 < T_l \leq \Gamma \leq T_h\} \tag{6.6}$$

From Equation [6.5] it can be noted that the thrust is now unbounded below. This means that a thrust of magnitude lower than $\|\boldsymbol{T}\| < T_l$ would be feasible in the relaxed problem, but is not feasible in reality and for the original constraint. However, this conclusion from plain eye-ball inspection of the constraints is misleading in this case. It is mathematically proven by Açıkmeşe and Ploen [2007, pp. 1356-1357] that *an optimal solution satisfying the relaxed constraints will also satisfy the original thrust constraints.* The same proof also yields the result that the profile is a *max-min-max type* of profile, as was already explained in Section 4.3. Unfortunately, this proof is too long to be discussed at this point. In mathematical terms, the main outcome is that the optimal control history for Γ is also that of $\|\boldsymbol{T}\|$, or:

$$\|\boldsymbol{T}^*(t)\| = \Gamma^*(t) \tag{6.7}$$



(a) Introduction of scalar slack variable Γ. The set C is convex because the constraint is affine.



(b) The convexified set of controls $\mathcal{C}$, remapped to a truncated cone.

FIGURE 6.2 Convexification of the control space using the scalar slack variable Γ.

and thus $\|\boldsymbol{T}^*\|$ must also fulfill the thrust-boundary constraint Equation [6.4]. The interested reader is pointed at the original source [Açıkmeşe and Ploen 2007].[4]

A more intuitive explanation for this is obtained if one visualizes the control space, and takes the fact that the constraints are "hiddenly linked" for granted. The resulting control space when combining both constraints is shown in Figure 6.2(b). After circumventing the nonconvexity, the set of controls now lies within a convex, truncated cone. In this drawing, the Γ-axis is the same as the affine control space shown in Figure 6.2(a). Consequently, the arrows of length T_l and T_h, indicating the dimension of the cone caps, are also located at the points T_l and T_h on the Γ axis. The sliced cone thus has a half angle of 45°.

Naturally, more constraints might need to be imposed on the trajectory. One could think of a glide-slope constraint, or a constraint on the landing-site viewing-angle, which is especially important for the LL and its HDA system. In fact, almost any constraint can be taken into account in the convex-programming framework. To be able to harness the beneficial properties of SOCPS, the requirement is to formulate constraints as second-order cones, which are of the form (cf. Equation [5.18]):

$$\|A_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{x} + d_i, \quad i = 1, \ldots, m$$

Any of the individual elements in this constraint may be set to zero, of course. The simplest example for such a constraint would be a bound on the velocity, which could

4 A brief outline of the proof is as follows. First, the Hamiltonian is formed. Then the necessary conditions for optimality are applied. From this the co-states can be derived. It is proven that the co-states associated to Γ and the thrust can never become zero. Having knowledge of the co-states, Pontryagin's maximum principle can be applied, from which it can be shown that Γ^* and $\|\boldsymbol{T}^*\|$ must be identical.

be expressed as $\|\dot{\boldsymbol{r}}\| \leq v_{\max}$.[5] This is a second-order cone constraint with $A = 0$ and $\boldsymbol{c} = \boldsymbol{0}$.

With these modifications, the convexified problem is:

▶ PROBLEM 4 Convexified soft-landing CITOC with terminal-state constraints.

$$
\begin{aligned}
&\text{minimize} \int_0^{t_f} \Gamma(t)\,\mathrm{d}t \\
&\text{subject to} \\
&\ddot{\boldsymbol{r}}(t) = \boldsymbol{g} + \boldsymbol{T}(t)/m(t),\ r_z(t) \geq 0 \\
&\dot{m}(t) = -\alpha\Gamma(t) \\
&\|\boldsymbol{T}(t)\| \leq \Gamma(t),\ 0 < T_l \leq \Gamma(t) \leq T_h \\
&\|A_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{x} + d_i,\ i = 1, \ldots, m \\
&m(0) = m_0,\ \boldsymbol{r}(0) = \boldsymbol{r}_0,\ \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0 \\
&\boldsymbol{r}(t_f) = \boldsymbol{r}_f,\ \dot{\boldsymbol{r}}(t_f) = \dot{\boldsymbol{r}}_f
\end{aligned}
$$

6.2 CHANGE OF VARIABLES

Although the nonconvex control-constraint has now been convexified, the problem formulation is not fit for convex solvers yet. This is because the EOMS in Problem 4 are nonlinear in $m(t)$. Because the system needs to be of linear form for the time-discretization to be shown in the sequel, this nonlinearity must be avoided. The EOMS will thus be linearized by a change of variables that requires no approximations. This section demonstrates how the control variables $\boldsymbol{T}(t)$ and $\Gamma(t)$ can be changed to yield a linear second-order differential-equation that can eventually be cast into a linear, time-invariant, state-space system (LTI).

Following this argument, consider the following change of variables:

$$
\sigma(t) \triangleq \frac{\Gamma(t)}{m(t)}, \quad \boldsymbol{\tau}(t) \triangleq \frac{\boldsymbol{T}(t)}{m(t)} \tag{6.8}
$$

This change casts the control variables as control *accelerations*, rather than thrust *forces*. Then the EOMS can then be written with direct acceleration input as:

$$
\ddot{\boldsymbol{r}}(t) = \boldsymbol{\tau}(t) + \boldsymbol{g} \tag{6.9}
$$

$$
\frac{\dot{m}(t)}{m(t)} = -\alpha\sigma(t) \tag{6.10}
$$

The translational EOM Equation [6.9] is now already linear, while the mass dynamics in Equation [6.10] require further changes to be linearized.

5 Such a constraint can be important for limiting the airspeed during a Mars-landing scenario, but is most likely not required for a Lunar landing scenario. During Mars landings, the airspeed must not increase above the equivalent of supersonic Mach numbers, because otherwise thruster-flow interactions can lead to severe problems.

It is important to remark that the optimal controls are still equivalent, as was the case for $\boldsymbol{T}(t)$ and $\Gamma(t)$:

$$\|\boldsymbol{\tau}^*(t)\| = \sigma^*(t) \quad \forall\, t \in (0, t_f) \tag{6.11}$$

Note that the change to control accelerations comes as no surprise, because most other guidance laws also generate acceleration inputs rather than thrust values (such as *E*-Guidance, for example [Cherry 1964]). This change of variables impacts the entire mathematical formulation, and the individual elements of the optimization problem are thus discussed in the next few sections.

6.2.1 Cost function

The first step is to verify that the optimization criterion will still minimize the propellant mass—even with the change of variables. Consider Equation [6.10]: a linear, first order, ordinary differential-equation (ODE). This ODE can be solved using elementary differential-equation theory, by applying the integrating-factor technique [Boyce and DiPrima 2008]. The solution is then given by:

$$m(t) = m_0 \exp\left(-\alpha \int_0^{t_f} \sigma(t)\, \mathrm{d}t\right) \tag{6.12}$$

From this it is clear that maximizing the final mass $m(t_f)$ is equivalent to minimizing the time-integral of the new control variable σ, because α is a positive constant. The objective function can thus be written as:

$$\text{maximize}\, m(t_f) = \text{minimize} \int_0^{t_f} \sigma(t)\, \mathrm{d}t \tag{6.13}$$

which is still a valid cost-function.

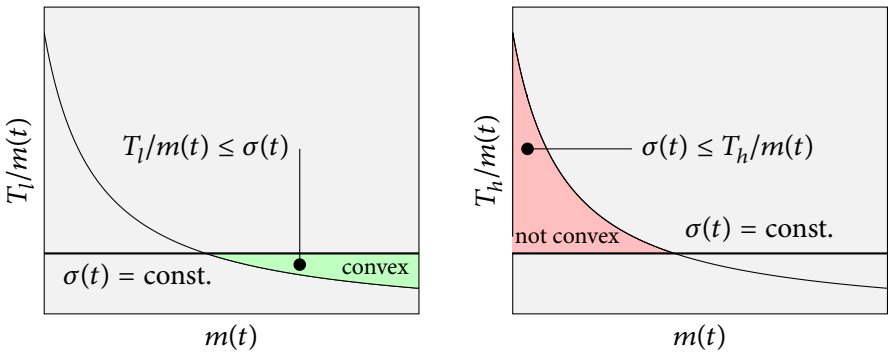
6.2.2 Control and state constraints

The control constraints can be expressed with the changed variables as:

$$\|\boldsymbol{\tau}(t)\| \leq \sigma(t) \tag{6.14}$$

$$\frac{T_l}{m(t)} \leq \sigma(t) \leq \frac{T_h}{m(t)} \qquad \forall\, t \in (0, t_f) \tag{6.15}$$

Unfortunately, the latter constraint introduces a new nonlinearity and a new nonconvexity, because of the division by the time-dependent problem-variable $m(t)$. This is illustrated in Figure 6.3. This drawing was created assuming that $\sigma(t)$ was constant, while the thrust-bound constraints were changing with the time-varying mass. This is a reasonable assumption, considering that a bang-bang thrust-profile is expected, for which the thrust magnitude is constant for long time intervals. In essence, this is an inequality plot for $f(x) = 1/x$, with $x = m(t)$. In this plot, the shaded areas define the constrained control-space. It can clearly be seen that the upper thrust-bound describes a nonconvex area, whereas the lower one does not present an issue.



(a) Convex lower bound **(b)** Nonconvex upper bound

FIGURE 6.3 Nonconvexity in the upper thrust-bound introduced by the change of variables, assuming the thrust command was constant.

The nonconvexity can be removed by another change of variables. Let:

$$z(t) \triangleq \ln m(t) \tag{6.16}$$

Then, the ODE for the mass flow, Equation [6.10], simplifies to:

$$\dot{z}(t) = -\alpha\sigma(t) \tag{6.17}$$

The inequality constraint, Equation [6.15], can be rewritten as:

$$T_l e^{-z(t)} \leq \sigma(t) \leq T_h e^{-z(t)} \tag{6.18}$$

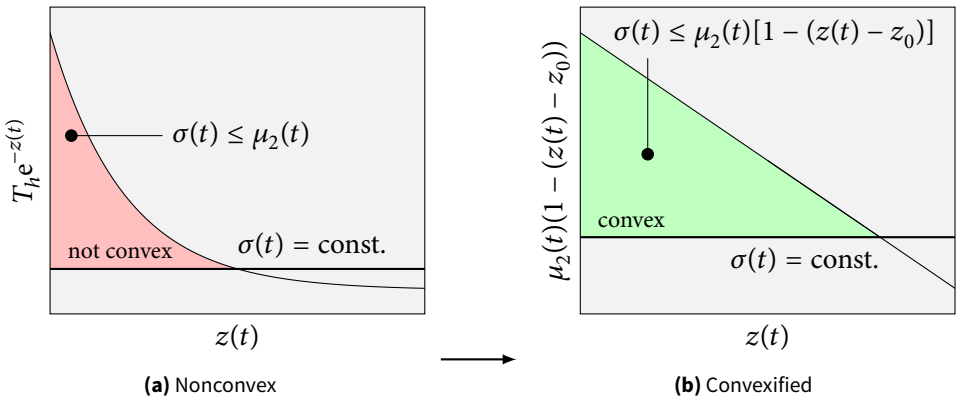
Given the similarity of the graphs of $f(z) = 1/z$ and $f(z) = e^{-z}$, it is easy to see that the new constraint still defines a convex lower bound and nonconvex upper bound (see Figure 6.3). Another problem is that the exponent, being a transcendental function, must not be part of an SOCP constraint. To resolve this, the constraints can be approximated by the Taylor series of $f(z) = e^{-z}$. Series expansions are described by [Boyce and DiPrima 2008]:

$$f(z) \approx \sum_{n=0}^{\infty} \frac{f^{(n)}(z_0)}{n!} (z - z_0)^n \tag{6.19}$$

where n denotes the n-th order derivative and z_0 is the nominal value around which the series is expanded. Retaining terms up to second order ($n = 2$), the lower thrust-bound constraint can be written as:

$$T_l e^{-z_1(t)} \left[1 - (z(t) - z_1(t)) + \tfrac{1}{2}(z(t) - z_1(t))^2\right] \leq \sigma(t) \tag{6.20}$$

which describes a second-order cone. Here, z_1 denotes the nominal point for the lower constraint. Its convexity can be visualized in the same manner as Figure 6.3 (a). Lower and higher orders would be possible for the series as well, but the second-order approximation is convex and provides enough precision.



(a) Nonconvex **(b)** Convexified

FIGURE 6.4 Nonconvex upper thrust-bound after the second change of variables, and the resulting convex control-space through linearization.

The upper thrust bound can be convexified by linearizing the constraint ($n = 1$):

$$\sigma(t) \leq T_h e^{-z_2(t)} [1 - (z - z_2(t))] \quad [6.21]$$

Note that anything above first order would not be convex anymore, see Figure 6.4.

The constant factors pre-multiplying the constraints can be abbreviated as:

$$\mu_1(t) \triangleq T_l e^{-z_1(t)}, \quad \mu_2(t) \triangleq T_h e^{-z_2(t)} \quad [6.22]$$

This convexification of the control space is illustrated in Figure 6.4.

Taylor series-expansions approximate the given function about a nominal point, in this case the mass-equivalent variables z_1 and z_2. Therefore, for Equations [6.20] and [6.21] to be sufficiently accurate, sensible nominal points are needed (they do not need to be identical). These can be derived by starting with the definition of z again:

$$z(t) = \ln m(t) = \ln(m_0 - \alpha \Gamma(t) t) \quad [6.23]$$

To provide bounds, one may choose to set $\Gamma(t) = T_l$ for the lower constraint, and $\Gamma(t) = T_h$ for the upper:

$$z_1(t) = \ln(m_0 - \alpha T_l t) \quad [6.24]$$
$$z_2(t) = \ln(m_0 - \alpha T_h t) \quad [6.25]$$

An additional constraint may be enforced to ensure that the physical bounds on $z(t)$ are not violated:

$$z_2(t) \leq z(t) \leq z_1(t) \quad [6.26]$$

With this, the constraints have been fully convexified. Before moving on to the discretization of the equations, an intermediate CITOC can be formulated as follows:

▶ PROBLEM 5 Convexified soft-landing CITOC with changed control- and constraint-variables.

$$
\begin{aligned}
&\text{minimize } \int_0^{t_f} \sigma(t)\,\mathrm{d}t \\
&\text{subject to} \\
&\ddot{\boldsymbol{r}}(t) = \boldsymbol{\tau}(t) + \boldsymbol{g}, \quad r_z(t) \geq 0 \\
&\dot{z}(t) = -\alpha\sigma(t) \\
&\|\boldsymbol{\tau}(t)\| \leq \sigma(t) \\
&\mu_1(t)\left[1 - [z(t) - z_1(t)] + \tfrac{1}{2}[z(t) - z_1(t)]^2\right] \leq \sigma \leq \mu_2(t)[1 - (z(t) - z_2(t)] \\
&z_1(t) \leq z(t) \leq z_2(t) \\
&\|A_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^{\mathsf{T}}\boldsymbol{x} + d_i,\ i = 1, \dots, m \\
&m(0) = m_0,\ \boldsymbol{r}(0) = \boldsymbol{r}_0,\ \dot{\boldsymbol{r}}(0) = \dot{\boldsymbol{r}}_0,\ \boldsymbol{r}(t_f) = \boldsymbol{r}_f,\ \dot{\boldsymbol{r}}(t_f) = \dot{\boldsymbol{r}}_f
\end{aligned}
$$

6.3 DISCRETIZATION

Having established a fully convex set of constraints, the control problem can now be discretized to a constrained finite-time optimal-control problem (CFTOC). A conversion from finite time to discrete time fundamentally entails dividing a given time interval $t \in [t_0, t_f]$ into a number of steps N. If the temporal nodes are spaced apart by a time Δt, then the time at node k is simply described by:

$$t_k = k\Delta t + t_0, \quad k = 0, 1, \dots, N \tag{6.27}$$

If the problem is not explicitly time-dependent, one may set $t_0 = 0\,\mathrm{s}$ without loss of generality. The given final time is then $t_f = N\Delta t$. The following sections will outline how this discretization is applied to the EOMs with the changed variables, starting with an expression of the control variables.

6.3.1 Control parametrization

The most important step in the discretization is the definition of a parametrization of the control variables $\boldsymbol{\tau}(t)$ and $\sigma(t)$. This parametrization needs to map the finite-time optimal-controls found by the solver to continuous time for use in the actual system. This will be achieved with a *zero-order hold* discretization, which will be described in the sequel.

To start the derivation, let $\boldsymbol{p}_j \in \mathbb{R}^4$ be the vector containing the magnitude of the constant optimum-controls at the nodes $j = 1, 2, \dots, N$, as determined by the solver.

Then the control can be expressed in continuous time using some prescribed basis functions $\varphi_j(t) : \mathbb{R}_+ \to \mathbb{R}$:[6]

$$\boldsymbol{u}(t) = \begin{pmatrix} \boldsymbol{\tau}(t) \\ \sigma(t) \end{pmatrix} = \sum_{j=1}^{N} \boldsymbol{p}_j \varphi_j(t), \quad t \in [0, t_f) \tag{6.28}$$

For said zero-order hold discretization, $\varphi_j(t)$ must be defined as follows:

$$\varphi_j(t) = \begin{cases} 1 & \text{when } t \in [t_{j-1}, t_j[\\ 0 & \text{otherwise} \end{cases} \tag{6.29}$$

This can be interpreted as follows: The j-th basis function $\varphi_j(t)$ is only *active* ($\varphi_j(t) = 1$) starting from time $t = t_{j-1} = (j-1)\Delta t$ up until *but not including* the time $t = t_j$. On this interval, it activates the optimal controls given by $\boldsymbol{p}_j$. During the remaining time-interval it is *inactive* ($\varphi_j = 0$), and the controls $\boldsymbol{p}_j$ do not contribute. This is illustrated for the simplified case of scalar controls $p_j \in \mathbb{R}$ in Figure 6.5. The basis functions φ_j become active at time step $j-1$. From this point on, the control command p_j is held up until the next basis function becomes active. Because a bang-bang control profile is expected to be the output, this approach should ideally suit the algorithm. The following example further clarifies the principle:

☑ EXAMPLE 6.1 The concept of zero-order hold discretization.

Let $u(t) \in \mathbb{R}$ be the control variable. Further let $N = 3$, and thus $j = 1, \dots, 3$ and $k = 0, \dots, 3$. The time step is given as $\Delta t = 1\,\text{s}$. Assume that the magnitude of the optimal controls is given by:

$$p_1 = 1, \quad p_2 = 0, \quad p_3 = 1$$

Then the continuous-time control on the interval $t \in [0, \Delta t[$ is specified by the basis functions:

$$\begin{aligned} &\varphi_1 = 1 && \leftarrow \text{(active)} \\ &\varphi_2 = 0, \quad \varphi_3 = 0 && \leftarrow \text{(inactive)} \end{aligned}$$

The controls on the same interval are now defined by the sum:

$$\left.\begin{aligned} u(t) &= \sum_{j=1}^{3} p_j \varphi_j(t) = p_1\varphi_1(t) + p_2\varphi_2(t) + p_3\varphi_3(t) \\ &= 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 = 1 \end{aligned}\right\} 0 \le t < \Delta t$$

6 Note that in Equation [6.28] the upper summation limit is not generally N. In the case of the zero-order hold basis-functions defined in the sequel, it is indeed necessary to have N basis functions. If one chose for example, Chebychev polynomials instead, the number of basis functions could probably be reduced to some other value $M < N$, which could speed up the optimization. This is not considered for the sake of simplicity at this point, however. For details, see [Açıkmeşe and Ploen 2005].

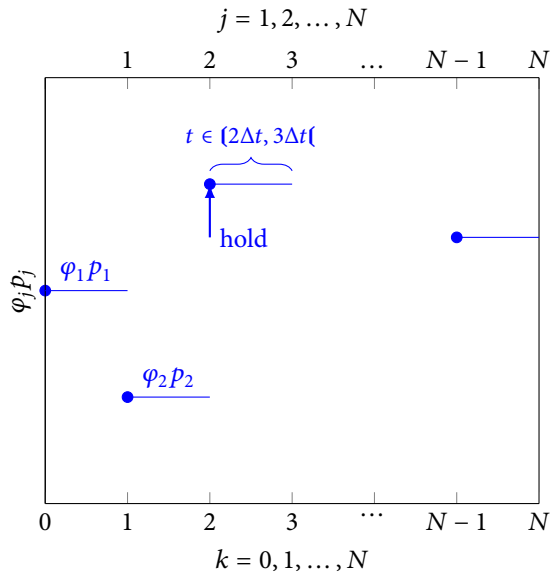


FIGURE 6.5 Illustration of the zero-order hold discretization of the controls, assuming some scalar controls.

Similarly, the controls for the remaining time can be found to be:

$$u(t) = 0, \quad \Delta t \leq t < 2\Delta t$$
$$u(t) = 1, \quad 2\Delta t \leq t < 3\Delta t$$

Note that the controls are actually undefined at exactly $t = 3\Delta t$. However, given a sufficiently small time step in the time integration of the plant, this will not have any impact.

As an alternative implementation, a first-order hold discretization was considered. The basis function for this is [Blackmore et al. 2010, p. 1166]:

$$\varphi_j(t) = \begin{cases} \dfrac{t_j - t}{\Delta t} & \text{when } t \in (t_{j-1}, t_j(\\ \dfrac{t - t_j}{\Delta t} & \text{when } t \in (t_j, t_{j+1}(\\ 0 & \text{otherwise} \end{cases} \qquad [6.30]$$

It was quickly found that the saw-tooth like pattern resulting from the first-order hold lead to unacceptable errors. This happens because of the bang-bang control profile, with which this discretization scheme cannot cope. The zero-order hold produces errors of nearly negligible magnitude and is thus the preferred scheme in this study. It has been successfully put to practice in Chapter 9.

6.3.2 Continuous-time state-space system definition

Next, it is necessary to set up the *state-space system* for the continuous-time EOMS, from which the discrete-time equations can be derived. State-space systems are a convenient way to express and analyze physical processes that are linear in the state and controls,

or can be linearized [Antsaklis and Michel 2006]. Consider a state vector $\boldsymbol{x}(t)$ and a control vector $\boldsymbol{u}(t)$. Then the physical process can be represented with the following – generally nonlinear – *state equation*:

$$\dot{\boldsymbol{x}}(t) = f(t, \boldsymbol{x}(t), \boldsymbol{u}(t)), \quad f : \mathbb{R}_+ \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \tag{6.31}$$

This is also sometimes called the *process equation*. Note that differential equations of arbitrary order can be expressed as a first-order state-equation, see [Antsaklis and Michel 2006]. One may also write:

$$\dot{\boldsymbol{x}} = A_c(\boldsymbol{x}, t)\boldsymbol{x}(t) + B_c(\boldsymbol{x}, t)\boldsymbol{u}(t), \quad A_c \in \mathbb{R}^{n\times n},\ B_c \in \mathbb{R}^{n\times m} \tag{6.32}$$

If $f(t, \boldsymbol{x}(t), \boldsymbol{u}(t))$ is linear in the state and the controls, then the model can be written as a *linear time-varying state-equation*:

$$\dot{\boldsymbol{x}}(t) = A_c(t)\boldsymbol{x}(t) + B_c(t)\boldsymbol{u}(t), \quad A_c \in \mathbb{R}^{n\times n},\ B_c \in \mathbb{R}^{n\times m} \tag{6.33}$$

where the time-varying matrices A_c and B_c are also called the *system matrix* and *input matrix*, respectively. This is a system of first-order linear ODEs that can be solved in terms of $A_c(t)$ and $B_c(t)$ using techniques shown in many books, for example [Luneberger 1979; Rugh 1995; Antsaklis and Michel 2006]. For clarity of the derivations, this will be demonstrated in the sequel. If the system and input matrices are constant, the important special case of a *linear time-invariant system (LTI)* occurs:

$$\dot{\boldsymbol{x}}(t) = A_c\boldsymbol{x}(t) + B_c\boldsymbol{u}(t) \tag{6.34}$$

To bring the EOMS into this form, let the state vector for this problem be composed of position, velocity, and mass (see also Section 3.2):

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{r}^\mathsf{T}(t) & \dot{\boldsymbol{r}}^\mathsf{T}(t) & z(t) \end{bmatrix}^\mathsf{T} \tag{6.35}$$

such that $\boldsymbol{x}(t) \in \mathbb{R}^7$. The control vector remains:

$$\boldsymbol{u}(t) = \begin{bmatrix} \boldsymbol{\tau}^\mathsf{T}(t) & \sigma(t) \end{bmatrix}^\mathsf{T} \tag{6.36}$$

with $\boldsymbol{u}(t) \in \mathbb{R}^4$. The EOMS were given in Problem 4 as:

$$\ddot{\boldsymbol{r}}(t) = \boldsymbol{\tau}(t) + \boldsymbol{g} \tag{6.37}$$

$$\dot{z}(t) = -\alpha\sigma(t) \tag{6.38}$$

With these definitions, the EOMS can be rewritten in state-space form:

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} \dot{\boldsymbol{r}}(t) \\ \ddot{\boldsymbol{r}}(t) \\ \dot{z}(t) \end{pmatrix} = \underbrace{\begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times1} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & 0 \end{bmatrix}}_{A_c} \underbrace{\begin{pmatrix} \boldsymbol{r}(t) \\ \dot{\boldsymbol{r}}(t) \\ z(t) \end{pmatrix}}_{\boldsymbol{x}(t)} + \underbrace{\begin{bmatrix} 0_{3\times3} & 0_{3\times1} \\ I_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & -\alpha \end{bmatrix}}_{B_c} \underbrace{\begin{pmatrix} \boldsymbol{\tau}(t) \\ \sigma(t) \end{pmatrix}}_{\boldsymbol{u}(t)} + \cdots$$

$$\cdots \underbrace{\begin{bmatrix} 0_{3\times3} & 0_{3\times1} \\ I_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 0 \end{bmatrix}}_{B_g} \underbrace{\begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix}}_{\boldsymbol{g}_c(t)} \quad [6.39]$$

which turns out to be an LTI, greatly easing further developments. Compared to the general definition of an LTI in Equation [6.34] above, gravity has been added as an additional constant input to the system. Writing the continuous-time gravity-vector in the same dimension as the control vector, $\boldsymbol{g}_c \in \mathbb{R}^4$, the input matrix B_g is formulated. By noticing that the products $B_g\boldsymbol{g}_c$ and $B_c\boldsymbol{g}_c$ lead to the same result, the input matrix B_c will be used for all practical purposes in the following. This saves memory in computer implementations, which is in fact a major issue for the algorithm at hand. The full LTI for the EOMS is thus:

$$\dot{\boldsymbol{x}}(t) = A_c\boldsymbol{x}(t) + B_c\boldsymbol{g}_c(t) + B_c\boldsymbol{u}(t) \quad [6.40]$$

6.3.3 Continuous-time state-space solution

The general solution for linear time-varying systems can be derived using standard techniques, as derived, for example, in [Antsaklis and Michel 2006, p. 55]. This general solution can be shown to be:

$$\boldsymbol{x}(t, t_0, \boldsymbol{x}_0) = \underbrace{\Phi(t, t_0)\boldsymbol{x}_0}_{\text{homogeneous ...}} + \underbrace{\int_{t_0}^{t} \Phi(t, s)B_c(s)\boldsymbol{u}(s)\,\mathrm{d}s}_{\text{...particular solution}} \quad [6.41]$$

where $\Phi(t, t_0) \in \mathbb{R}^{n\times n}$ is called the *state-transition matrix* and $\boldsymbol{x}_0 \in \mathbb{R}^n$ is the initial state. This solution expresses the state $\boldsymbol{x}$ at time t, given an initial epoch t_0 and the corresponding initial state. The first term is called the *homogeneous solution*, because it corresponds to the solution of the homogeneous system $\dot{\boldsymbol{x}} = A_c\boldsymbol{x}$. This solution allows the interpretation of the state-transition matrix as a means of propagating the initial state in time. The *particular solution* corresponds to the solution of the system with controls and gravity as forcing terms.

Clearly, to find a solution, the state transition matrix has to be determined. Antsaklis and Michel [2006, pp. 144-156] list several methods for this, along with a number of important properties of the matrix. Fortunately, in case of an LTI, the state-transition matrix can easily be found. First, consider again the vector differential-equation for the homogeneous solution:

$$\dot{\boldsymbol{x}}(t) = A_c\boldsymbol{x}(t) \quad [6.42]$$

In scalar form, this equation and its solution would read:

$$\dot{x}(t) = ax(t) \tag{6.43}$$

$$x(t) = ce^{at} \tag{6.44}$$

As it turns out, for constant A_c the solution of the vector differential-equation is in fact of similar form:

$$\boldsymbol{x}(t, t_0, \boldsymbol{x}_0) = e^{A_c(t-t_0)}\boldsymbol{x}_0 \tag{6.45}$$

where $\exp(A_c(t - t_0)) \in \mathbb{R}^{n\times n}$ is called the *matrix exponential*. A general matrix exponential can be expressed with a series representation that is, in fact, the matrix equivalent of the series expansion of the scalar exponential-function:

$$e^a = 1 + \sum_{k=1}^{\infty} \frac{a^k}{k!} \tag{6.46}$$

$$e^A = \mathsf{I} + \sum_{k=1}^{\infty} \frac{A^k}{k!} \tag{6.47}$$

Seeing that the matrix exponential is the state-transition matrix in this case, the general homogeneous solution to the LTI can be written as:

$$\boldsymbol{x}(t, t_0, \boldsymbol{x}_0) = e^{A_c(t-t_0)}\boldsymbol{x}_0 = \Phi(t, t_0)\boldsymbol{x}_0 = \left[\mathsf{I} + \sum_{k=1}^{\infty} \frac{A_c^k(t - t_0)^k}{k!}\right]\boldsymbol{x}_0 \tag{6.48}$$

such that the total solution to the continuous-time state-space system derived in Equation [6.39] reads:

$$\boldsymbol{x}(t, t_0\boldsymbol{x}_0) = e^{A_c(t-t_0)}\boldsymbol{x}_0 + \int_{t_0}^{t} e^{A_c(t-s)}B_c\boldsymbol{u}(s)\,\mathrm{d}s + \int_{t_0}^{t} e^{A_c(t-s)}B_c\boldsymbol{g}_c\,\mathrm{d}s \tag{6.49}$$

where particular solutions have been introduced for the control and gravity vectors.

Although one could attempt to solve the integrals in the solution at this point, this is not required, because the solution still needs to be transformed to discrete time in any case. The discretization is done in the next section.

6.3.4 Discrete-time state-space system definition

A key result from linear system theory is that continuous-time systems can be transcribed to discrete time, where the discrete-time system and input matrices are derived from their continuous-time counterparts. The state-space system then reads:

$$\boldsymbol{x}(k + 1) = A_d(k)\boldsymbol{x}(k) + B_d(k)\boldsymbol{u}(k) \quad \leftarrow \text{linear time-varying} \tag{6.50}$$

$$\boldsymbol{x}(k + 1) = A_d\boldsymbol{x}(k) + B_d\boldsymbol{u}(k) \quad \leftarrow \text{linear time-invariant} \tag{6.51}$$

For the guidance algorithm, it is the goal to express the EOMS in form of a discrete LTI, so that the SOCP solver can determine a finite number of control commands. This will be shown next.

For time-invariant systems, Antsaklis and Michel [2006, pp. 182-185] show that the state and input matrices can be converted to discrete-time counterparts as follows:

$$A_d = \mathrm{e}^{A_c \Delta t} \tag{6.52}$$

$$B_d = \int_0^{\Delta t} \mathrm{e}^{A_c \Delta t s}\, \mathrm{d}s\, B_c \tag{6.53}$$

Another possibility to calculate B_d is to use the following relationship, if A is nonsingular [Stengel 1986]:

$$B_d = A_c^{-1}(A_d - \mathsf{I})B_c \tag{6.54}$$

Antsaklis and Michel [2006, p. 185] propose yet another, numerically efficient method for determining the discrete time state matrices that requires no analytical integral-evaluations. First compute:

$$\Psi(\Delta t) = \sum_{j=0}^{\infty} \frac{\Delta t^j}{(j+1)!} A_c^j \tag{6.55}$$

Then the matrices can be determined by:

$$A_d = \mathsf{I} + \Delta t A_c \Psi(\Delta t) \tag{6.56}$$

$$B_d = \Delta t \Psi(\Delta t) B_c \tag{6.57}$$

With the discretized state-space system in place, its solution will be established in the following section.

6.3.5 Discrete-time state-space solution

Antsaklis and Michel [2006, p. 64] prove that the discrete-time solution of the discretized LTI Equation [6.51] is expressed by the following formula:

$$\boldsymbol{x}(k) = \Phi_d(k, k_0)\boldsymbol{x}(k_0) + \sum_{j=k_0}^{k-1} \Phi_d(k, j+1) B_d \boldsymbol{u}(j) \tag{6.58}$$

which differs from the continuous-time solution, Equation [6.41], by the discretized matrices and the sum, which replaces an integral. Antsaklis and Michel [2006, pp. 63-64] also show that the state-transition matrix propagating the homogeneous solution from step k_0 to step k is, generally:

$$\Phi_d(k, k_0) = \prod_{j=k_0}^{k-1} A_d(j), \quad k > k_0, \quad \Phi_d(k, k) = \mathsf{I} \tag{6.59}$$

If the system matrix is constant, then this can simply be written as:

$$\Phi_d(k, k_0) = A_d^{k-k_0} \tag{6.60}$$

It is easy to see how Φ_d is derived by writing out a few terms of the linear time-invariant state-space system, Equation [6.51]:

$$\begin{aligned}
\boldsymbol{x}(k_0+1) &= A_d\boldsymbol{x}(k_0) &&= \Phi(k_0+1,\ k_0)\boldsymbol{x}(k_0)\\
\boldsymbol{x}(k_0+2) &= A_d\boldsymbol{x}(k_0+1) = A_d(A_d\boldsymbol{x}(k_0)) &&= A_d^2\boldsymbol{x}(k_0) = \Phi(k_0+2,\ k_0)\boldsymbol{x}(k_0)\\
\boldsymbol{x}(k_0+3) &= A_d\boldsymbol{x}(k_0+2) = A_d(A_d^2\boldsymbol{x}(k_0)) &&= A_d^3\boldsymbol{x}(k_0) = \Phi(k_0+3,\ k_0)\boldsymbol{x}(k_0)
\end{aligned}$$

where it has been made use of the property that $\Phi(k+2,k) = \Phi(k+2,k+1)\Phi(k+1,k)$.

To finally arrive at the total solution, gravity must be added as constant input again, which yields:

$$\boldsymbol{x}(k) = A_d^{k-k_0}\boldsymbol{x}(k_0) + \sum_{j=k_0}^{k-1} A_d^{k-(j+1)}B_d\boldsymbol{u}(j) + \sum_{j=k_0}^{k-1} A_d^{k-(j+1)}B_d\boldsymbol{g}_c \qquad [6.61]$$

Without loss of generality, one may set $k_0 = 0$, slightly simplifying the solution:

$$\boldsymbol{x}(k) = A_d^k\boldsymbol{x}_0 + \sum_{j=0}^{k-1} A_d^{k-(j+1)}B_d\boldsymbol{u}(j) + \sum_{j=0}^{k-1} A_d^{k-(j+1)}B_d\boldsymbol{g}_c \qquad [6.62]$$

With the discrete-time state-space system and its solution in place, the problem can be cast into its final form, as demonstrated in the next section.

6.4 FINAL PROBLEM FORMULATION

At this point, all quantities except for the controls $\boldsymbol{u}(j)$ are known in the solution of the discrete-time LTI. These results are summarized below:

$$\boldsymbol{x}(t) = \begin{pmatrix}\boldsymbol{r}^\mathsf{T}(t) & \dot{\boldsymbol{r}}^\mathsf{T}(t) & z(t)\end{pmatrix}^\mathsf{T} \qquad [6.63]$$

$$\boldsymbol{u}(t) = \begin{pmatrix}\boldsymbol{\tau}^\mathsf{T}(t) & \sigma(t)\end{pmatrix}^\mathsf{T} \qquad [6.64]$$

$$\boldsymbol{x}(t_0) = \begin{pmatrix}\boldsymbol{x}_0^\mathsf{T} & \dot{\boldsymbol{r}}^\mathsf{T}(t_0) & z(t_0)\end{pmatrix}^\mathsf{T} \qquad [6.65]$$

$$\boldsymbol{g}_c = \begin{pmatrix}0 & 0 & g & 0\end{pmatrix}^\mathsf{T} \qquad [6.66]$$

$$A_c = \begin{bmatrix}0_{3\times3} & \mathsf{I}_{3\times3} & 0_{3\times1}\\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1}\\ 0_{1\times3} & 0_{1\times3} & 0\end{bmatrix} \qquad [6.67]$$

$$B_c = \begin{bmatrix}0_{3\times3} & 0_{3\times1}\\ \mathsf{I}_{3\times3} & 0_{3\times1}\\ 0_{1\times3} & -\alpha\end{bmatrix} \qquad [6.68]$$

$$\Psi(\Delta t) = \sum_{j=0}^{\infty}\frac{\Delta t^j}{(j+1)!}A_c^j \qquad [6.69]$$

$$A_d = \mathsf{I} + \Delta t A_c\Psi(\Delta t) \qquad [6.70]$$

$$B_d = \Delta t\Psi(\Delta t)B_c \qquad [6.71]$$

As the only unknown is now the finite-sized control-vector $\boldsymbol{u}(j)$, it becomes possible to solve for its elements in a batch approach. For this, the solution must be rewritten as *a single equation* that contains the solution of the state *at all time steps* as a function of

the control input. This can be achieved by stacking all matrices and vectors on top of one another. The convex solver will then solve this equation in a batch approach and return the optimal control vector.

6.4.1 Stacking of the linear time-invariant system

Borrelli et al. [2013, p. 144] show that the stacking of all solutions yields the following system that is achieved by expanding the terms in Equation [6.62] (recalling that the series are only defined for $k > k_0$):

$$\underbrace{\begin{pmatrix} \boldsymbol{x}(0) \\ \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix}}_{\boldsymbol{X}} = \underbrace{\begin{bmatrix} A_d^0 & 0 & \cdots & \cdots & 0 \\ 0 & A_d^1 & \ddots & \cdots & \vdots \\ 0 & \ddots & A_d^2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & A_d^N \end{bmatrix}}_{S_x} \underbrace{\begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \vdots \\ \boldsymbol{x}_0 \end{pmatrix}}_{\boldsymbol{X}_0} \cdots$$

$$\cdots + \underbrace{\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_d & 0 & \cdots & 0 \\ A_d B_d & B_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_d^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix}}_{S_u} \underbrace{\begin{pmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix}}_{\boldsymbol{U}} \cdots$$

$$\cdots + \underbrace{\begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_d & 0 & \cdots & 0 \\ A_d B_d & B_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_d^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix}}_{S_u} \underbrace{\begin{pmatrix} \boldsymbol{g}_c \\ \boldsymbol{g}_c \\ \boldsymbol{g}_c \\ \vdots \\ \boldsymbol{g}_c \end{pmatrix}}_{\boldsymbol{G}} \quad [6.72]$$

With the notation introduced in the equation, this can be written more compactly:

$$\boldsymbol{X} = S_x \boldsymbol{X}_0 + S_u \boldsymbol{U} + S_u \boldsymbol{G} \quad [6.73]$$

where $\boldsymbol{X}$ is the stacked solution vector, S_x denotes the stacked system matrix, $\boldsymbol{X}_0$ represents the stacked initial conditions, S_u is the stacked input matrix, $\boldsymbol{U}$ describes the stacked decision / control vector, and $\boldsymbol{G}$ is the stacked gravity input-vector. The zeros in the equation above are sized as $0_{7\times7}$ in S_x, and $0_{7\times4}$ in S_u and S_g. The dimensions are $(\boldsymbol{X}, \boldsymbol{X}_0) \in \mathbb{R}^{((N+1)\times7)}$, $S_x \in \mathbb{R}^{((N+1)\times7)\times((N+1)\times7)}$, $S_u \in \mathbb{R}^{((N+1)\times7)\times(N\times4)}$, and $(\boldsymbol{U}, \boldsymbol{G}) \in \mathbb{R}^{(N\times4)}$.

6.4.2 Cost function

Based on this stacked solution, the previously established continuous-time Problem 5 can now be converted to the final discrete-time problem that will be used with the solver. To be converted first is the cost function that was given as:

$$\text{minimize} \int_0^{t_f} \sigma(t)\,\mathrm{d}t \tag{6.74}$$

This continuous-time integral can be expressed in a discretized variant as:

$$\int_0^{t_f} \sigma(t)\,\mathrm{d}t \approx \sum_{k=1}^{N} \boldsymbol{e}_\sigma^\mathsf{T} \boldsymbol{p}(k)\Delta t \tag{6.75}$$

where

$$\boldsymbol{e}_\sigma = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^\mathsf{T} \tag{6.76}$$

can be considered a "selection vector," that extracts σ from $\boldsymbol{p}(k)$. In the following, $\boldsymbol{e}$ and E will denote selection vectors and matrices, where the respective indices indicate which element is extracted. Although the integral has now been discretized, Equation [6.75] is still not in its final form. It was stated in Section 5.5 that an SOCP must be subject to a cost of the following, affine form:

$$\text{minimize } \boldsymbol{f}^\mathsf{T}\boldsymbol{x} \tag{6.77}$$

With some simple manipulation, the sum in Equation [6.75] above can be replaced by inner products:

$$\sum_{k=1}^{N} \boldsymbol{e}_\sigma^\mathsf{T} \boldsymbol{p}(k)\Delta t = \begin{bmatrix} \Delta t \boldsymbol{e}_\sigma^\mathsf{T} & \overset{\times N}{\cdots} \end{bmatrix} \begin{pmatrix} \boldsymbol{u}(0) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix} = \boldsymbol{e}_{N\sigma\Delta t}^\mathsf{T} \boldsymbol{U} \tag{6.78}$$

where the selection vector is of dimension $\boldsymbol{e}_{N\sigma\Delta t} \in \mathbb{R}^{(4\times N)}$.

6.4.3 Constraints

Before the constraints can be discretized, it is important to realize that they apply only at the temporal nodes $k = 1, \ldots, N$, and not at $k = 0$,[7] which corresponds to the initial conditions. This is because no controls act at the initial epoch in discrete time. However, the solution to the LTI, $\boldsymbol{X}$, contains all states from $k = 0, \ldots, N$, and therefore *does* include the initial epoch. To make sure that there are only N constraints, the solution $\boldsymbol{X}$ must reduced from size $N + 1$ to N. This simply achieved by eliminating the first

7 This holds in *discrete time*. If a control was applied at $k = 0$, then the equation $\boldsymbol{x}(0) = \boldsymbol{x}_0$ would not hold anymore. In *continuous time* a control acts from the beginning on.

row in $\boldsymbol{X}$ to remove the initial state. To illustrate the need for this, consider again the solution that was given in Equation [6.125]:

$$\begin{pmatrix} \boldsymbol{x}(0) \\ \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix} = S_x \boldsymbol{X}_0 + S_u \boldsymbol{G} + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_d & 0 & \cdots & 0 \\ A_d B_d & B_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_d^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix} \begin{pmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix} \tag{6.79}$$

Clearly, there are only N controls $\boldsymbol{u}$, but $N+1$ states $\boldsymbol{x}$. For that reason, the first row of S_u contains only zeros, so that the matrix has $N+1$ element rows and the product $S_u \boldsymbol{U}$ will result in a vector with $N+1$ elements. To reduce the system to N rows, the first row is simply eliminated. Writing out the example of the controls, this then reads:

$$\underbrace{\begin{pmatrix} \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix}}_{\tilde{\boldsymbol{X}}} = \tilde{S}_x \tilde{\boldsymbol{X}}_0 + \tilde{S}_g \boldsymbol{G} + \underbrace{\begin{bmatrix} B_d & 0 & \cdots & 0 \\ A_d B_d & B_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_d^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix}}_{\tilde{S}_u} \underbrace{\begin{pmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix}}_{\boldsymbol{U}} \tag{6.80}$$

where the tilde has been introduced to denote the reduced matrices and vectors. There is no need to reduce $\boldsymbol{U}$ and $\boldsymbol{G}$ because these vectors already contain only N elements. No harm has been done by this operation, because the initial condition is still part of the homogeneous solution through the vector $\tilde{\boldsymbol{X}}_0$.

To begin with the reformulation of the constraints, consider the inequality constraint relating the slack variable $\Gamma(t)$ to the thrust command $\boldsymbol{T}(t)$. It has been defined with the changed variables in Equation [6.14] as:

$$\|\boldsymbol{\tau}(t)\| \leq \sigma(t) \tag{6.81}$$

In the discretized equations, the controls $\boldsymbol{\tau}$ and σ have been lumped in the vector $\boldsymbol{U}$. Consequently, to apply the constraint at the temporal nodes, the control vector $\boldsymbol{\tau}(k)$ has to be extracted from $\boldsymbol{U}$ for the relevant time step. This is done with a similar approach as for the cost function: by defining "extraction matrices". With these matrices, the constraint can be written as:

$$\begin{aligned} &\left\| \begin{bmatrix} \mathsf{I}_{3\times 3} & 0_{3\times 1} \end{bmatrix} \begin{bmatrix} 0_{4\times 4} & \overset{\times(k-1)}{\cdots} & \mathsf{I}_{4\times 4} & 0_{4\times 4} & \overset{\times[N-(k-1)]}{\cdots} \end{bmatrix} \boldsymbol{U} \right\| \leq \cdots \\ &\qquad \cdots \boldsymbol{e}_\sigma^\mathsf{T} \begin{bmatrix} 0_{4\times 4} & \overset{\times(k-1)}{\cdots} & \mathsf{I}_{4\times 4} & 0_{4\times 4} & \overset{\times[N-(k-1)]}{\cdots} \end{bmatrix} \boldsymbol{U} \end{aligned} \tag{6.82}$$

$$\|E_{\boldsymbol{\tau}} E_{k,\boldsymbol{U}} \boldsymbol{U}\| \leq \boldsymbol{e}_\sigma^\mathsf{T} E_{k,\boldsymbol{U}} \boldsymbol{U}, \quad k = 1, 2 \ldots, N, \quad E_{\boldsymbol{\tau}} \in \mathbb{R}^{3\times 4}, \quad E_{k,\boldsymbol{U}} \in \mathbb{R}^{4\times(N\times 4)} \tag{6.83}$$

in which $E_{\boldsymbol{\tau}}$ extracts the $\boldsymbol{\tau}(k)$ component from the control vector $\boldsymbol{u}(k)$, and $E_{k,\boldsymbol{U}}$ selects the control vector $\boldsymbol{u}(k)$ at time step k out of the lump vector $\boldsymbol{U}$. This constraint has to be imposed N times to account for each temporal node k, and there are thus N constraints of this type. Therefore, $E_{k,\boldsymbol{U}}$ is different for each time step, and is of the structure as defined in Equation [6.82] above.

Next, the constraint defining the upper and lower thrust-bounds can be discretized. This was defined in Problem 5 as:

$$\mu_1(t)\left[1-[z(t)-z_1(t)]+\tfrac{1}{2}[z(t)-z_1(t)]^2\right]\leq\sigma\leq\mu_2(t)[1-(z(t)-z_2(t)] \tag{6.84}$$

Because of the dependency on the current mass $z(t)$, this constraint requires knowledge of the state $\boldsymbol{x}=\begin{pmatrix}\boldsymbol{r}^\mathsf{T} & \dot{\boldsymbol{r}}^\mathsf{T} & z\end{pmatrix}^\mathsf{T}$ at each time-step. Three steps are necessary to discretize this constraint: i) the solution $\boldsymbol{x}(k)$ is substituted into the constraint; ii) a new vector E_z is introduced, that selects the $z(t)$ from the solution; and iii) the resulting constraints are stacked on top of one another to end up with a single vector-inequality constraint.

For simplicity, this is first demonstrated by considering the individual time steps separately, before the constraint is stacked:

$$\begin{aligned}\mu_1(k)\left\{1-\left[\begin{pmatrix}0_{1\times6} & 1\end{pmatrix}\boldsymbol{x}(k)-z_1(k)\right]+\tfrac{1}{2}\left[\begin{pmatrix}0_{1\times6} & 1\end{pmatrix}\boldsymbol{x}(k)-z_1(k)\right]^2\right\}\leq\cdots\\ \cdots\boldsymbol{e}_\sigma^\mathsf{T}E_k\boldsymbol{U}\leq\mu_2(k)\left\{1-\left[\begin{pmatrix}0_{1\times6} & 1\end{pmatrix}\boldsymbol{x}(k)-z_2(k)\right]\right\},\quad k=1,2,\ldots,N\end{aligned} \tag{6.85}$$

with $\begin{pmatrix}0_{1\times6} & 1\end{pmatrix}=\boldsymbol{e}_z{}^\mathsf{T}\in\mathbb{R}^7$ being the r_z-selection vector. Now this equation can be stacked. To save space, first define the sparse z- and σ-selection matrices:

$$E_z=\begin{bmatrix}\boldsymbol{e}_z^\mathsf{T} & 0_{1\times7} & \cdots & 0_{1\times7}\\ 0_{1\times7} & \boldsymbol{e}_z^\mathsf{T} & \ddots & \vdots\\ \vdots & \ddots & \ddots & \vdots\\ 0_{1\times7} & \cdots & \cdots & \boldsymbol{e}_z^\mathsf{T}\end{bmatrix},\quad E_z\in\mathbb{R}^{N\times(7\times N)} \tag{6.86}$$

$$E_\sigma=\begin{bmatrix}\boldsymbol{e}_\sigma^\mathsf{T} & 0_{1\times4} & \cdots & 0_{1\times4}\\ 0_{1\times4} & \boldsymbol{e}_\sigma^\mathsf{T} & \ddots & \vdots\\ \vdots & \ddots & \ddots & \vdots\\ 0_{1\times4} & \cdots & \cdots & \boldsymbol{e}_\sigma^\mathsf{T}\end{bmatrix},\quad E_\sigma\in\mathbb{R}^{N\times(4\times N)} \tag{6.87}$$

It was shown that the reduced LTI solution is:

$$\tilde{\boldsymbol{X}}=\tilde{S}_x\tilde{\boldsymbol{X}}_0+\tilde{S}_u\boldsymbol{U}+\tilde{S}_u\boldsymbol{G} \tag{6.88}$$

With this, the constraint can be written as:

$$\begin{aligned}\begin{pmatrix}\mu_1(1)\\ \vdots\\ \mu_1(N)\end{pmatrix}\left\{\mathbf{1}_N-\left[E_z\tilde{\boldsymbol{X}}-\begin{pmatrix}z_1(1)\\ \vdots\\ z_1(N)\end{pmatrix}\right]+\frac{1}{2}\left[E_z\tilde{\boldsymbol{X}}-\begin{pmatrix}z_1(1)\\ \vdots\\ z_1(N)\end{pmatrix}\right]^2\right\}\leq\cdots\\ \cdots E_\sigma\boldsymbol{U}\leq\begin{pmatrix}\mu_2(1)\\ \vdots\\ \mu_2(N)\end{pmatrix}\left\{\mathbf{1}_N-\left[E_z\tilde{\boldsymbol{X}}-\begin{pmatrix}z_2(1)\\ \vdots\\ z_2(N)\end{pmatrix}\right]\right\}\end{aligned} \tag{6.89}$$

$$\begin{aligned}\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N-\left(E_z\tilde{\boldsymbol{X}}-\boldsymbol{z}_{1N}\right)+\tfrac{1}{2}\left(E_z\tilde{\boldsymbol{X}}-\boldsymbol{z}_{1N}\right)^2\right]\leq E_\sigma\boldsymbol{U}\cdots\\ \cdots\leq\boldsymbol{\mu}_{2N}\left[\mathbf{1}_N-\left(E_z\tilde{\boldsymbol{X}}-\boldsymbol{z}_{2N}\right)\right]\end{aligned} \tag{6.90}$$

so that $(\boldsymbol{\mu}_{1N}, \boldsymbol{\mu}_{2N}, \boldsymbol{z}_{1N}, \boldsymbol{z}_{2N}) \in \mathbb{R}^N$. This is a component-wise vector-inequality that compares the elements with respect to the thrust slack-variable σ. The constraint has other benefits as well. By substituting the solution of the differential equation into it, the equality constraints for the EOMs become redundant, and can be eliminated [Boyd and Vandenberghe 2004, p. 132, 142 f.].

At this point, the constraint on the physical bounds of the mass $z(t)$ remains to be put into appropriate form. This is defined in continuous time as (see Equations [6.24] to [6.26]):

$$\ln(m_0 - \alpha T_h t) \leq z(t) \leq \ln(m_0 - \alpha T_l t) \tag{6.91}$$

As for the previous constraint, the discrete-time solution can be substituted to establish N constraints that apply at the temporal nodes of the problem. Without showing this for the individual time steps, the total vector-inequality can be written as:

$$\underbrace{\begin{pmatrix} \ln(m_0 - \alpha T_h \Delta t \cdot 1) \\ \ln(m_0 - \alpha T_h \Delta t \cdot 2) \\ \vdots \\ \ln(m_0 - \alpha T_h \Delta t \cdot N) \end{pmatrix}}_{\boldsymbol{z}_{2N}} \leq E_z \tilde{\boldsymbol{X}} \leq \underbrace{\begin{pmatrix} \ln(m_0 - \alpha T_l \Delta t \cdot 1) \\ \ln(m_0 - \alpha T_l \Delta t \cdot 2) \\ \vdots \\ \ln(m_0 - \alpha T_l \Delta t \cdot N) \end{pmatrix}}_{\boldsymbol{z}_{1N}} \tag{6.92}$$

Before compiling the problem formulation, the last step is to formulate the final boundary conditions (BCs).[8] The initial BCs need not be specified explicitly again, because they are already contained in the solution of the LTI. For the continuous-time case, the final BCs were stated in Problem 5:

$$\boldsymbol{r}(t_f) = \boldsymbol{r}_f, \quad \dot{\boldsymbol{r}}(t_f) = \dot{\boldsymbol{r}}_f \tag{6.93}$$

This corresponds to time-step $t_f = N\Delta t$ in discrete time. The BC can be enforced as a terminal-state equality-constraint in a similar fashion as the previous constraints: the terminal-state-vector element $\boldsymbol{x}(N)$ is extracted from the total solution vector $\boldsymbol{X}$, which is then set equal to the BCs by means of an equality constraint. For this, the extraction matrix can be written as:

$$E_{\boldsymbol{x}(N)} = \left[0_{6\times 7} \overset{\times N}{\cdots} \quad I_{6\times 6} \quad 0_{6\times 1} \right], \quad E_{\boldsymbol{x}(N)} \in \mathbb{R}^{6\times((N+1)\times 7)} \tag{6.94}$$

It extracts the position and velocity vectors from the terminal-state vector, but not the mass (which should be left unconstrained). With this, the constraint can be written as:

$$E_{\boldsymbol{x}(N)} \boldsymbol{X} = (\boldsymbol{r}_f^\mathsf{T} \quad \dot{\boldsymbol{r}}_f^\mathsf{T})^\mathsf{T} \tag{6.95}$$

Note that because $\boldsymbol{X}$ is a linear system this is a convex constraint, suitable to SOCPs.

This completes the transcription of Problem 5 to discrete time with all constraints. Further constraints can be added either as affine equality constraints, or second-order cone inequalities, which both have to be discrete (see Chapter 7). The discrete-time SOCP is now finally formulated as follows:

8 Note that this is actually erroneously omitted from Problem 4 in [Açıkmeşe and Ploen 2007].

▶ PROBLEM 6 Convexified, discrete soft-landing CFTOC with changed control- and constraint-variables.

$$
\begin{aligned}
&\text{minimize } \boldsymbol{e}_{N\sigma\Delta t}^{\mathsf{T}}\boldsymbol{U} \\
&\text{subject to} \\
&\|E_{\boldsymbol{r}}E_{k,\boldsymbol{U}}\boldsymbol{U}\| \leq \boldsymbol{e}_{\sigma}^{\mathsf{T}}E_{k,\boldsymbol{U}}\boldsymbol{U}, \quad k = 1, 2 \ldots, N \\
&\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - (E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}) + \tfrac{1}{2}\left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right)^2\right] \leq E_{\sigma}\boldsymbol{U} \leq \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - (E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{2N})\right] \\
&\boldsymbol{z}_{2N} \leq E_z\tilde{\boldsymbol{X}} \leq \boldsymbol{z}_{1N} \\
&E_{\boldsymbol{x}(N)}\boldsymbol{X} = \begin{bmatrix}\boldsymbol{r}_f^{\mathsf{T}} & \dot{\boldsymbol{r}}_f^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}
\end{aligned}
$$

6.5 DETERMINING THE OPTIMAL TIME-TO-GO

With Problem 6 above, a discrete-time optimization-problem has been defined to find the guidance command. The principle of discretization has been introduced in Section 6.3. Transcribing the problem from continuous to discrete time was subject to one major assumption: the problem is defined for a *fixed* time-interval $t \in (0, t_f)$. Given a user-defined step-size, Δt, the problem is therefore discretized to a finite number of temporal nodes $N = t_f/\Delta t$. It is thus clear that Problem 6 finds the optimal solution *for a given number of steps*. However, the optimal final time t_f^*, still needs to be found. Therefore, an outer loop has to be established around the optimization problem, that again runs an optimization to find the optimum final time. The subject of this section is to determine and describe the best-suited method for this. For consistency of the terminology, t_f^* will be referred to as t_{go}^*, or optimal time-to-go, in the following.

6.5.1 Problem definition and optimization techniques

To survey possible algorithms to find the optimal t_{go}, it is a prerequisite to characterize the function that is to be optimized. In this case, Problem 6 becomes the "cost function." For the sake of finding the optimal t_{go}, it can be assumed that in Problem 6 the optimization is only subject to the number of steps N, and all other variables appearing in the problem are given constants. One can therefore write:

$$
\text{maximize}\, m_f = \text{minimize}\, J(N), \quad J(N) = m_0 - m_f,\ J : \mathbb{R}_+ \to \mathbb{R}_+ \qquad [6.96]
$$

where J represents the solution of Problem 6, expressed in terms of the used propellant mass. Clearly, no analytical optimum for J can be found. There are no derivatives, and the function is nonlinear. It is subject to only one variable. As Brent [2013] points out, one has to resort to numerically approximating the optimum iteratively in such cases. This source also defines the following requirements. A good search algorithm shall:

1. Guarantee convergence to a correct solution;
2. Converge with a prescribed tolerance; and
3. Converge as fast as possible.

The latter point translates to requiring a minimal number of expensive function evaluations. Although Brent [2013, p. 2] points out that "[most] algorithms for minimizing a nonlinear function of one or more variables find, at best, a local minimum," the reader is reminded that the solution of a convex optimization is always the global optimum. Thus, if an optimum for J is found, it is also guaranteed that for this particular problem it presents the *global* optimum.

Several algorithms that are suited to this problem can be identified:

Grid search The simplest approach to find the optimal time-to-go is to sample discrete values on a predefined interval $t_{go} \in (t_{go,min}, t_{go,max})$ with a given step-size Δt_{go} [Deb 2002]. Once everything has been sampled, the optimal sampled value is simply:

$$t_{go}^* = \underset{t_{go} \in \mathcal{T}}{\arg\min}\, J(t_{go}),\ \mathcal{T} = \{t_{go} : t_{go} = t_{go,min} + i\Delta t_{go}, i = 0, 1, \dots, N\} \quad [6.97]$$

Thus, a tolerance can be specified by choosing Δt_{go} accordingly. Although such a brute-force algorithm *will* find a solution and is easy to implement, it most certainly *will not* find it fast: most of the time is wasted on sampling sub-optimal points. This violates Requirement 3.

Stochastic search Stochastic search-methods and meta-heuristics improve on simple grid-searches by using cleverer sampling techniques, saving valuable execution time [Spall 2003; Engelbrecht 2007]. The most basic stochastic search is the Monte Carlo (MC) method. It samples N points on $t_{go} \in (t_{go,min}, t_{go,max})$ using pseudo-random numbers with a uniform probability density function (PDF). Spall [2003] refers to many more methods that are more efficient, including evolutionary algorithms. However, although feasible, such methods violate essentially all requirements: convergence is not guaranteed, tolerances are hard to specify, and they are – comparably – slow. In essence, stochastic searches remain a (very) smart brute-force approach. The application of such methods is more suited to problems with multiple local minima, where it is hard to find the global optimum. Because there is only a single (global) optimum in this problem, stochastic searches are an overkill.

Section searches A faster and thus more efficient way to find an optimum is to apply a section search. Such methods work on bounded intervals, and evaluate the objective function at points successively closer to the minimum by moving into the search interval. The search is terminated as soon as the last few function evaluations are as close to each other as the specified tolerance. Brent [2013] shows that the *golden-section search* in particular fulfills all the requirements.

Out of the methods presented above, the *golden-section search* appears to be the most popular method throughout the literature, mostly due to its proven efficiency. The classical book "Numerical Recipes: The Art of Scientific Computing" recommends it as the algorithm of choice for this application [Press et al. 2007, p. 492], and MATLAB provides it as the function `fminbnd` as part of the optimization toolbox [Mathworks 2014]. Blackmore et al. [2010] also propose a simple golden-section search for finding t_{go}^* in combination with a modified convex-guidance algorithm. Based on these

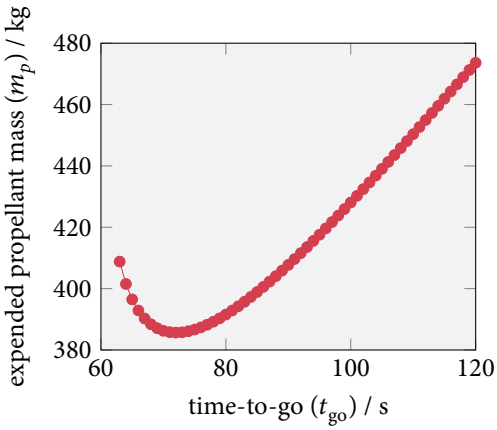


FIGURE 6.6 Expended propellant mass for different flight times for the scenario defined by Açıkmeşe and Ploen [2007], see also Section 6.6.4. The solution of Problem 6 as a function of the time-to-go clearly shows unimodal behaviour, with the optimum being at 72 s. Time step: 1 s.

recommendations and the independent literature survey, the golden-section search is thus also selected for this application.

6.5.2 Prerequisites for the golden-section search

The golden-section search is a simple, yet effective search method for functions of a single variable, which do not need to possess a derivative. However, the golden-section search is not applicable to all problems. There are two main requirements: i) the function must be *unimodal*; and ii) the minimum must lie within a known interval $\mathbf{(}a, b\mathbf{)}$, or $t_{go}^* \in \mathbf{(}t_{go,min}, t_{go,max}\mathbf{)}$. For consistency with the notation in the literature, a and b will be used to denote these lower and upper bounds in the sequel.

To clarify the applicability, a function is *unimodal* over the interval $\mathbf{(}a, b\mathbf{)}$ if it has a unique minimum in that interval [Bertsekas 1999]. Brent [2013] gives a more rigorous mathematical definition called "δ-unimodality", which the interested reader is invited to study. An interesting observation in this context is that (strictly) convex functions are always unimodal. Although it is yet to be proven mathematically that $J(t_{go})$ is indeed unimodal (if possible at all), practical observations have shown that this is indeed the case. As an illustration of this, consider Figure 6.6. The figure has been generated by solving Problem 6 repeatedly for different time-to-gos, so that essentially a grid-search has been performed. The input was given by the first test case from [Açıkmeşe and Ploen 2007], see also Section 6.6. The clearly unimodal behaviour of this test case is typical for the problem, and similar plots could be generated for other initial conditions as well.

Having established the unimodality, the bounds on the search space within which the optimum must lie can next be established. A straight-forward approach with a physical meaning is to relate the time-to-go to the available propellant mass. The maximum time-to-go would occur if the spacecraft approached the landing site at the lowest pos-

sible throttle setting and burned up all the propellant mass in the process. To calculate this time, consider the equation for the thrust Section 3.5:

$$T = \dot{m} g_0 I_{\text{sp}} \tag{6.98}$$

With the lower thrust-bound given as T_l and the propellant mass as m_p, the maximally feasible time-to-go is:

$$b = t_{\text{go,max}} = \frac{m_p}{\dot{m}_{\text{min}}} = m_p \frac{g_0 I_{\text{sp}}}{T_l} \tag{6.99}$$

In practice it is questionable whether this yields realistic values for the time-to-go. Especially when the lower thrust-bound is low, or can even be set to 0 N, the equation above will fail or yield very large values. This unnecessarily enlarges the search space. Therefore, the tuning parameter α is proposed here so that the user can influence the search space:

$$b = \frac{m_p}{\dot{m}_{\text{min}}} = m_p g_0 I_{\text{sp}} \left(\frac{1 + \alpha}{T_l + \alpha T_h} \right) \tag{6.100}$$

This allows specifying $t_{\text{go,max}}$ as a weighted average of the thrust-bounds.

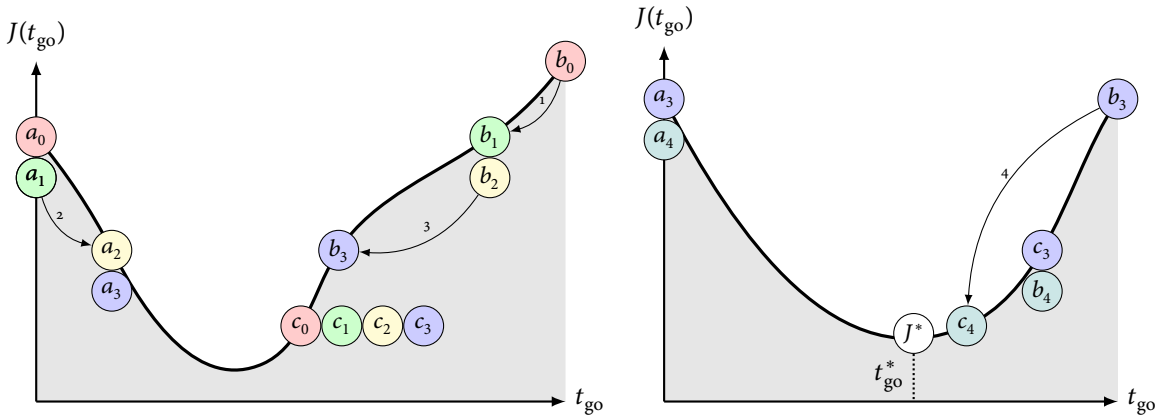
On first thought, the lower limit of the thrust bound could be established by repeating the procedure above taking into account the maximum thrust. However, this would be devoid of the real initial conditions. To give an extreme example, the vehicle could be almost directly above the surface at very low speed, in which the time-to-go would be extremely short. Estimating the lower limit directly based on the propellant is thus not viable: the initial conditions *must* be taken into account. To regard both, the initial conditions and vehicle parameters in terms of thrust and mass, Açıkmeşe and Ploen [2007] propose the following heuristic for the lower limit:

$$a = t_{\text{go,min}} = \|\dot{\boldsymbol{r}}_0\| \frac{m_{\text{dry}}}{T_h} \tag{6.101}$$

Although this has been shown to work in practice, initial test runs show that this underestimates the lower limit in most cases. Therefore, the search space for the time-to-go is unnecessarily enlarged. One reason for this is that the current distance to the target is not taken into account. To give another extreme example, it could happen that the vehicle hovers at nearly zero velocity a kilometer away from the landing site, in which case the equation above completely fails to estimate the lower bound. Therefore, the following alternative method is proposed:

$$a = 2 \sqrt{\left(\frac{x_f - x_0}{\dot{x}_f - \dot{x}_0} \right)^2 + \left(\frac{y_f - y_0}{\dot{y}_f - \dot{y}_0} \right)^2 + \left(\frac{z_f - z_0}{\dot{z}_f - \dot{z}_0} \right)^2} \tag{6.102}$$

This method expands on the idea of prescribing a linear acceleration profile in one axis for *E*-Guidance, extending it to all three degrees of freedom (see [Gerth and Mooij 2014]). Unfortunately, no vehicle-specific parameters are taken into account in this



(a) Decreasing interval size in the first steps. The interior point remains unchanged.

(b) This evaluation is closer to the minimum. The previously interior point becomes a boundary.

FIGURE 6.7 Successively bracketing a minimum by keeping three points: upper and lower boundaries a and b, and an interior point c.

formula. It can also lead to problems if the denominator of any of the fractions is (close to) zero, although this is not the case in most scenarios.

Having bracketed the minimum by the bounds a and b, the golden-section search can be commenced. This procedure will be explained in the following section.

6.5.3 Golden-section search algorithm

Having shown that the cost function $J(t_{go})$ for this problem is indeed unimodal, and that the minimum can be bracketed by a lower limit a and an upper limit b, the golden-section search algorithm can be introduced. The main idea behind this algorithm is to maintain a triplet of three points at any time: the two bracketing points a and b, and a point in their interior, c. Starting from this configuration, a new point x is evaluated that lies either between a and c, or b and c. Suppose x is evaluated between b and c. Then, if $f(c) < f(x)$, the new upper bound is given by x, so that the triplet is $\mathbf{(}a, c, x\mathbf{)}$. Otherwise, if $f(c) > f(x)$, $\mathbf{(}c, x, b\mathbf{)}$.

This is illustrated in Figure 6.7 (a). The initial triplet is given by $\mathbf{(}a_0, c_0, b_0\mathbf{)}$. The first evaluation is carried out between b_0 and c_0, for which it turns out to be that $f(c_0) < f(b_1)$. Therefore, b_1 becomes the new upper bound, and c_0 remains the minimum interior point. The same procedure is repeated twice, leading first to a new lower and then to a new upper bound. In the final configuration, a_3 and b_3 bracket the minimum, and c_3 is still identical to the initial point.

An interesting configuration occurs if the new evaluation is in fact lower than the neighbouring bound, as explained above. This is shown in Figure 6.7 (b). Here, in the fourth step, the evaluation turns out to be lower than the current upper bound c_3. Therefore, the previously interior point c_3 now becomes the new upper bound b_4, and the new point becomes the new interior point c_4. This procedure is repeated until the distance between the points becomes smaller than a specified tolerance.

So far, this method seems generic. The specific feature of the golden-section search is how to determine where the next trial, x, is evaluated: x is always evaluated by stepping

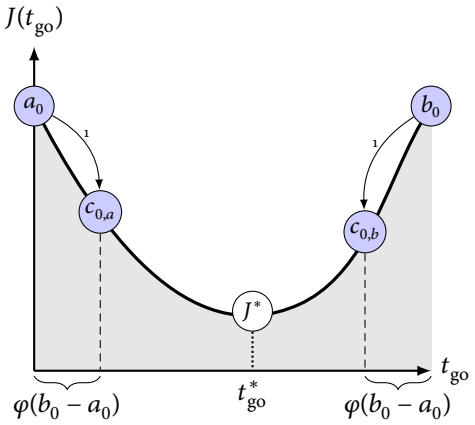


FIGURE 6.8 Geometry of the first bracketing step. The points are evaluated by moving in by the golden ratio.

into the interval by the *golden ratio* φ, so that the triplet of points (a, c, b) always keeps the same distance ratio between a and c, and b and c. The golden ratio is given by:

$$\varphi = \frac{3 - \sqrt{5}}{2} \approx 0.381\,966 \qquad [6.103]$$

This turns out to be the interval that leads to the fastest convergence, as shown by Press et al. [2007]. The algorithm can be put in words as follows: given a triplet of points, the next evaluation is to be performed in the larger of the two intervals (a, c) and (c, b). This evaluation must step into that interval by the golden ratio, which will keep the same distance between the points. This geometry is shown in Figure 6.8. This guarantees that with each step, the search space is decreased by a factor of $1 - \varphi \approx 0.618\,034$, which is the fastest of all methods. If one defines a tolerance for the precision of the optimum, ε, then the number of iterations that the algorithm will need is [Deb 2002, p. 48]:

$$\varphi^{n-1}(a - b) = \varepsilon \quad \Leftrightarrow \quad n = \log_\varphi\left(\frac{\varepsilon}{b - a}\right) + 1 \qquad [6.104]$$

Mathematically, this algorithm can be summarized as follows [Bertsekas 1999, p. 722]. Given the initial interval (a_0, b_0), first compute two interior trial points and their function values by stepping into the interval by the golden ratio from both sides (see Figure 6.8):

$$c_{0,a} = a_0 + \varphi(b_0 - a_0), \quad J(c_{0,a}) \qquad [6.105]$$

$$c_{0,b} = b_0 - \varphi(b_0 - a_0), \quad J(c_{0,b}) \qquad [6.106]$$

where the indices in c_b and c_a indicate in which half of the interval the values lie. Now the following cases can occur, and the new next variables at step $k + 1$ are assigned accordingly:

1. If $J(c_{k,a}) < J(c_{k,b})$:

$$\begin{aligned} a_{k+1} &= a_k,\ b_{k+1} = c_{k,a} \quad \text{if } J(a_k) \leq J(c_{k,a}) \\ a_{k+1} &= a_k,\ b_{k+1} = c_{k,b} \quad \text{if } J(a_k) > J(c_{k,a}) \end{aligned} \tag{6.107}$$

2. If $J(c_{k,a}) > J(c_{k,b})$:

$$\begin{aligned} a_{k+1} &= c_{k,b},\ b_{k+1} = b_k \quad \text{if } J(c_{k,b}) \geq J(b_k) \\ a_{k+1} &= c_{k,a},\ b_{k+1} = b_k \quad \text{if } J(c_{k,b}) < J(b_k) \end{aligned} \tag{6.108}$$

3. If $J(c_{k,a}) = J(c_{k,b})$:

$$a_{k+1} = c_{k,a}, \quad b_{k+1} = c_{k,b} \tag{6.109}$$

4. If $|b - a| \leq \varepsilon$ terminate.

This algorithm always keeps the lowest trial point as interior point, as discussed before. Moreover, note that it in fact requires only one function evaluation per step (except for the first step). This is because in the equations above one boundary value of the previous step always gets carried over to the next. For example, in Equation [6.107] the left boundary a is maintained, and the function value remains available, reducing the number of expensive function evaluations.

6.5.4 Faster convergence through parabolic interpolation

A straight-forward yet effective improvement of the golden-section search can be derived if one recognizes that the shape of *unimodal* cost functions often represents that of a parabola. Intuitively, it makes sense that if a function has only a single minimum, it is likely to be of parabolic shape. By inspecting the example plot for this problem, Figure 6.6, it appears that this applies here as well. Under such circumstances, faster convergence can be achieved by interpolating a parabola through the points $(a, J(a))$, $(c, J(c))$, and $(b, J(b))$ when appropriate. The minimum of the parabola through these points can be calculated with the following formula [Brent 2013]:

$$x = c - \frac{1}{2}\frac{(c-a)^2[J(c) - J(b)] - (c-b)^2[J(c) - J(a)]}{(c-a)[J(c) - J((b)] - (c-b)[J(c) - J(a)]} \tag{6.110}$$

This equation is applicable as long as none of the points are collinear. Once the parabola has been fitted, the cost function is evaluated again at the extremum, $J(x)$, after which the process is repeated.

Figure 6.9 illustrates the process. The minimum of $J(t_{go})$ is initially bracketed by a_0, c_0, and b_0. Fitting a parabola through these points, J is now evaluated its minimum x. The brackets are now adjusted according to the rules described in the previous section, and the process is repeated.

The question remains how the golden-section search and the parabolic interpolation can be combined. Brent [2013, p. 79f.] presents a consolidated algorithm for this, which has come to be know as "Brent's method". A golden-section step is performed if:

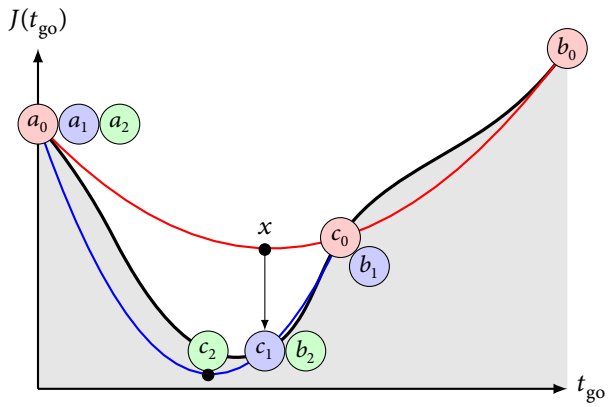


FIGURE 6.9 Converging to the minimum by parabolic interpolation.

- The denominator in Equation [6.110] is zero (any of the two points are collinear);
- The point x is outside of the bracket, $x \notin (a, b)$; or
- The interpolation approaches the tolerance value ε, $|e| \leq \varepsilon$ or $|x - c| \geq \frac{1}{2}|e|$, with e the smaller of the values $e = |x - a|$ or $e = |x - b|$.

An algorithm listing based on Brent's method is shown in Algorithm 1 (Appendix C), and is also used in the software implementation to find the optimal time-to-go.

6.6 VERIFICATION AND MODEL ANALYSIS

Having established all the fundamentals of convex guidance, it is essential to verify the claims made in this chapter. The problem formulation in this report differs slightly to that of the reference [Açıkmeşe and Ploen 2007], because some details were spared in that paper and the problem was not formulated in a stacked fashion. Thus, this section presents test cases for all subproblems discussed before. Based on the outcomes some conclusions on the models themselves can be drawn, which will also be presented in this section. For software-implementation details, see Chapter 8.

6.6.1 State-space system dynamics and state propagation

To verify that the implementation of the dynamical model works as expected, two main elements need to be tested: the impact of the gravitational acceleration on the vehicle, and whether the system models thruster dynamics correctly. For this, two tests are devised:

UNIT TEST 6.1 Verification of the discrete-time system dynamics model.

DYN-1 Drop the vehicle vertically without any thrust acting on it.
Expected outcome: position and velocity should change according to elementary kinematics.

TABLE 6.1 Problem parameters for unit test DYN-1.

(a) Input

r_0, m	$\dot{r}_0$, m/s	g, m/s^2	Δt, s	N
100	0	1.6249	1	10

(b) Output

EXPECTED		RESULT		ERROR	
r_f, m	$\dot{r}_f$, m/s	r_f, m	$\dot{r}_f$, m/s	ε_r, %	$\varepsilon_{\dot{r}}$, %
18.75	−16.25	18.75	−16.25	0	0

DYN-2 Let the vehicle hover.
Expected outcome: the vehicle should remain level and at zero speed, mass should decay, and the thrust should slowly decrease accordingly.

These tests are simple, and yet cover all functions of this block. It suffices to test only the vertical channel of the system, as the others work identically. These tests in the vertical channel have also been chosen because they overlap with more unit tests in the following.

6.6.1.1 *DYN-1: vertical drop*

This test verifies that the constant gravity model is realized correctly by the software. The fundamental EOMS for a vertical drop are easily derived as:

$$\ddot{r} = g \tag{6.111}$$

$$\dot{r} = \dot{r}_0 + gt \tag{6.112}$$

$$r = r_0 + \dot{r}_0 t + \tfrac{1}{2} g t^2 \tag{6.113}$$

The solution of the dynamical system modelling the system dynamics has been derived in Section 6.4.1. To test the influence of gravity only, any control input is omitted, and the solution forced by gravity reads then (cf. Equation [6.73]):

$$\boldsymbol{X}_g = S_x \boldsymbol{X}_0 + S_g \boldsymbol{G} \tag{6.114}$$

which should model the same dynamics as Equations [6.111] to [6.113].

The input to the unit test is given in Table 6.1(a). The output of test is documented in Table 6.1(b), where the expected results have been computed with Equations [6.111] to [6.113] and the results using code that implements the dynamical model Equation [6.119]. As can be seen, the results match perfectly (up to numerical accuracy), and the unit test is thus *passed.*

6.6.1.2 *Error analysis*

Unit test DYN-1 shows that the discretized dynamical model matches the analytical solution up to numerical accuracy, which might come as a surprise. Some analysis on the origin of this gives important insight, and is thus shown in this section.

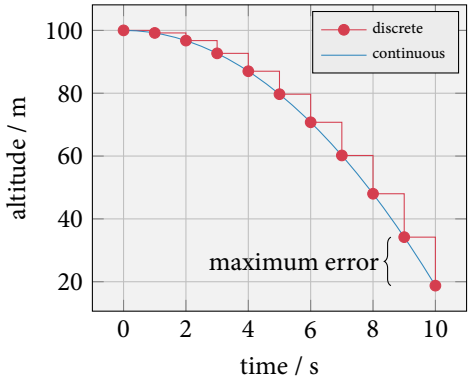


FIGURE 6.10 Solution of the unit-test scenario in continuous time according to Equation [6.113] and zero-order hold discretization according to Equation [6.119].

The reason why the numerical solution perfectly matches the analytical solution at the temporal nodes can be found by expanding the discrete model, which is given in Equation [6.119] above. The derivation for this is shown in Appendix B, and the result for the z-component is:

$$r_z(k+1) = r_z(k) + \dot{r}_z(k)\Delta t + \tfrac{1}{2} g \Delta t^2 \qquad [6.115]$$

$$\dot{r}_z(k+1) = \dot{r}_x(k) + g\Delta t \qquad [6.116]$$

Comparing this to Equation [6.111] to [6.113], it is clear that this is, in fact, an *exact discretization* for the temporal nodes.

Important information and insight on the accuracy of the solution can be derived from plotting the trajectory versus time. This is shown in Figure 6.10. It can be seen that, as expected, the discrete-time solution matches the analytical, continuous-time solution perfectly at the temporal nodes. However, it is not to be forgotten that the discretization is of zero-order hold type. If the discrete-time solution was to be translated back to continuous time, the staircase-like pattern that is visible in the figure would occur (see also Figure 6.5). This means that an error is made in between the time-steps.

It is important to characterize this error, allowing for an error analysis for continuous-time simulations later on. Clearly, the maximum error occurs at the last times steps, as shown in the figure. The reason for this is intuitively clear: at this point, the vehicle is the fastest, and moves the most in between two time steps. This leads to the largest error. The magnitude of the error can be derived by comparing the altitude at the last two time steps:

$$\varepsilon_{r,\max} = \tfrac{1}{2} g[(N-1)\Delta t]^2 - \tfrac{1}{2} g(N\Delta t)^2 = \tfrac{1}{2} g\left[(N-1)^2 - N^2\right]\Delta t^2 \qquad [6.117]$$

$$= \tfrac{1}{2} g(1-2N)\Delta t^2 \qquad [6.118]$$

so that the error is of order $\mathcal{O}(N\Delta t^2)$ *in between the nodes*. Thus, the step size has a significant influence on the accuracy.

It is to be noted that the maximum error is not necessarily reached in practice. This is due to the control input to the spacecraft through the main engine. This will decrease the velocity, and thus likely even decrease the maximum error with time. Assuming

that the vehicle is fastest at the initial epoch, and slows down after that, the maximum discretization error will thus occur right in the beginning.

6.6.1.3 *DYN-2: hovering flight*

For test case DYN-2, the basic input is the same as that given in Table 6.1a(a) for test DYN-1. However, an additional *command input* is added to the system, so that the system now reads:

$$\boldsymbol{X} = S_x \boldsymbol{X}_0 + S_g \boldsymbol{G} + S_u \boldsymbol{U} \tag{6.119}$$

The control vector was defined in Equation [6.36] as:

$$\boldsymbol{u}(t) = \begin{pmatrix} \boldsymbol{\tau}^\mathsf{T} & \sigma \end{pmatrix}^\mathsf{T}, \quad \boldsymbol{\tau} = \boldsymbol{T}/m, \quad \sigma = \|\boldsymbol{\tau}\| \tag{6.120}$$

Thus, the control input is, in fact, an acceleration.

For hovering flight, the gravitational acceleration and the thrust acceleration of the vehicle need to cancel out. Therefore, the control input is:

$$\boldsymbol{\tau} = \begin{pmatrix} 0 & 0 & -g \end{pmatrix}^\mathsf{T}, \quad \sigma = g \tag{6.121}$$

These controls are stacked into $\boldsymbol{U}$ as explained at length before. Because the control input is an *acceleration*, the thrust $\boldsymbol{T} = m\boldsymbol{\tau}$ is thus expected to decrease with time, because the spacecraft mass decreases.

To verify that the mass dynamics are modelled properly, the expected final mass needs to be computed. This can be done in an iterative fashion as follows. The mass flow at any given time is:

$$T = g_0 I_{\text{sp}} \dot{m}, \quad \Leftrightarrow \quad \dot{m} = \frac{T}{g_0 I_{\text{sp}}} \tag{6.122}$$

With the constant magnitude of the thrust acceleration given by σ, the thrust magnitude can be computed as $T(t) = \sigma m(t)$. Therefore, the mass at each time step can be computed as:

$$\dot{m}(k+1) = \frac{\sigma m(k)}{g_0 I_{\text{sp}}} \tag{6.123}$$

$$m(k+1) = m(k) - \dot{m}(k+1)\Delta t \tag{6.124}$$

where $k = 0, 1, \ldots, N$.

The in- and outputs to the test case are given in Table 6.2. Note that the final position and velocity match perfectly again. The final mass is not exactly the same, but at a discrepancy of twenty milligram and a difference of 0.002 %, the error is arguably acceptable. With these small errors, the unit test is thus considered to be *passed*.

6.6.2 Finite-time optimal-control problem

The SOCP, or CFTOC that describes the soft-landing guidance-problem, has been formulated in Problem 6 on Page 80. It solves the boundary-value problem for a given

TABLE 6.2 Problem parameters for unit test DYN-2. An additional thrust acceleration input is given by Equation [6.121].

(a) Input

r_0, m	$\dot{r}_0$, m/s	g, m/s^2	I_{sp}, s	g_0, m/s^2	m_0, kg	g, m/s^2	Δt, s	N
100	0	1.6249	300	9.806 65	100	1	100	

(b) Output

EXPECTED			RESULT			ERROR		
r_f, m	$\dot{r}_f$, m/s	m_f, kg	r_f, m	$\dot{r}_f$, m/s	m_f, kg	ε_r, %	$\varepsilon_{\dot{r}}$, %	ε_m, %
0	0	94.625	0	0	94.627	0	0	0.002

discretization, *i. e.*, given step size Δt and number of steps N. As this problem formulation is of the same form as that given by Açıkmeşe and Ploen [2007], the reference data given in that paper can fortunately be used for verification. The unit test is therefore the following:

UNIT TEST 6.2 Verification of the CFTOC solution for a given time of flight.

FT-1 Solve Problem 6 for the same input as specified for the first test case in the paper by Açıkmeşe and Ploen [2007].
Expected outcome: The output should match the reference data.

Note that because the SOCP in combination with the golden-section search algorithm makes up the convex guidance algorithm, further tests on this will, in fact, be performed in the system-integration test in Section 6.6.4 below. This test is elaborated upon in the following.

6.6.2.1 *FT-1: comparison to reference data*

The reference scenario used by Açıkmeşe and Ploen [2007] simulates a Mars landing starting at a severely misguided AG: the vehicle is at 1.5 km down range, and is, in fact, *flying away* from the landing site with a speed of 100 m/s down range. At the same time, it is descending quickly at 75 m/s from an altitude of 2 km. Guidance shall target a landing at the origin of the reference frame ($\mathcal{L}$), with completely nulled speed at TD. These challenging initial conditions have been chosen to test the performance of the algorithm under adverse conditions. If it works well in this case, one may interpret this as a sign of robustness, and the algorithm should work even better under nominal circumstances.

Problem 6 has been implemented as a prototype in MATLAB 2013b using CVX, a package for specifying and solving convex programs [Grant and Boyd 2008; Grant and Boyd 2014] (see Section 8.2.1). The solver used in combination is ECOS [Domahidi et al. 2013]. For details on the implementation, see Chapter 8. The problem input is given in Table 6.3.[9]

9 Zero-entries indicate matches up to numerical accuracy, which is considered to be smaller than 10^{-6} in relative accuracy.

TABLE 6.3 Input and output of the unit test FT-1. Omitted values are zero.

(a) Input

g, m/s^2	g_0, m/s^2	m_0, kg	I_{sp}, s	T_l, N	T_h, N	Δt, s	N	$r_{0,x}$, m	$r_{0,z}$, m
3.7114	9.807	1905	200.48	4971.8	13 258.2	1	72	2000	1500

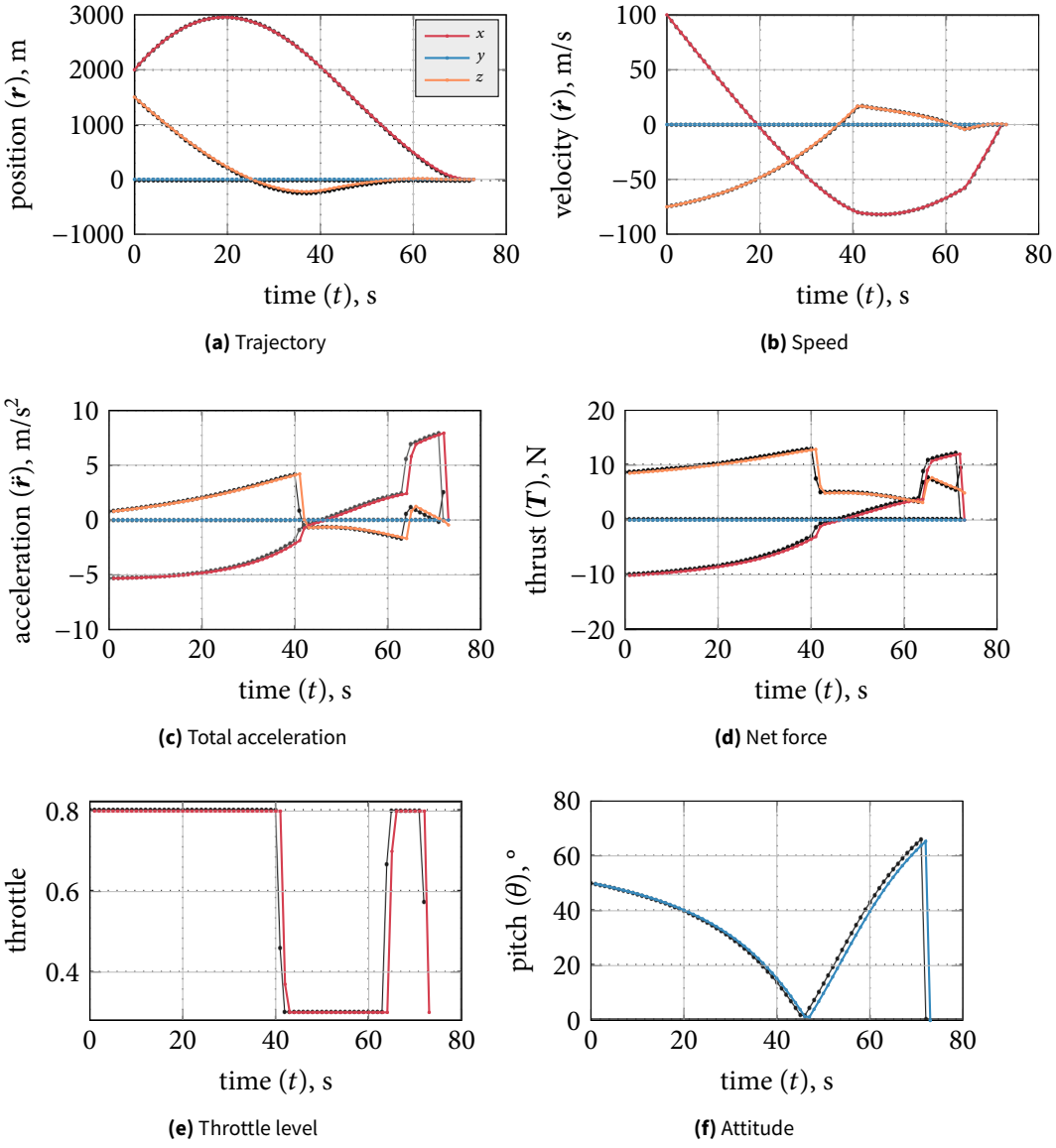
(b) Output and results

PARAMETER	REFERENCE	OUTPUT	DIFFERENCE, %
$r_{f,x}$, m	0	0	0
$r_{f,y}$, m	0	0	0
$r_{f,z}$, m	0	0	0
$\dot{r}_{f,x}$, m/s	0	0	0
$\dot{r}_{f,y}$, m/s	0	0	0
$\dot{r}_{f,z}$, m/s	0	0	0
m_p, kg	387.9	389.9	0.5

The results in Figure 6.11 and Table 6.3 indicate that the implemented algorithm very closely reproduces the reference data. Especially the position and speed seem to match perfectly. One curiosity can be seen in Figure 6.11(a): the altitude drops below the 0 m mark, and there is thus flight segment that is subterranean. The reasons for this and a mitigation measure are discussed in Section 7.1.

However, upon closer inspection it can be seen that the commands are in fact *not* identical (Figures 6.11(c) – (f)). In Figure 6.11(c) it can be seen that the controls appear to be shifted by one time-step to the right. Moreover, the final commands in the z-component do not match. Table 6.3 also shows that there is a small discrepancy of 0.5 % in the used propellant mass.

Because this is a deterministic problem, no difference should occur. Possible reasons to explain the discrepancies have been intensively studied, but have unfortunately remained inconclusive. Although no proof for this can be given, there is reason to assume that the difference is caused by a different number of temporal nodes where the controls are applied. In this scenario, the time-of-flight is 72 s and the time step is $\Delta t = 1$ s. This means that there are $1 + N = 73$ temporal nodes (0 s, 1 s, … , 72 s) and that there should be N controls (one at each node except for the initial node). However, for a reason that could not be determined, Açıkmeşe and Ploen [2007] claim to have controls at the initial node! After considering the problem at length, a rationale and a mathemati-



(a) Trajectory

(b) Speed

(c) Total acceleration

(d) Net force

(e) Throttle level

(f) Attitude

FIGURE 6.11 Verification of the solution of the convex guidance SOCP. The plots have been obtained by overlaying the simulation output with the data from the first test case of [Açıkmeşe and Ploen 2007].

cal explanation for this could not be found. To explain why their statement appears to be erroneous, consider the system equation again:

$$\begin{pmatrix} \boldsymbol{x}(0) \\ \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix} = \begin{bmatrix} A_d^0 & 0 & \cdots & \cdots & 0 \\ 0 & A_d^1 & \ddots & \cdots & \vdots \\ 0 & \ddots & A_d^2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & A_d^N \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \vdots \\ \boldsymbol{x}_0 \end{pmatrix} \cdots$$

$$\cdots + \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_d & 0 & \cdots & 0 \\ A_d B_d & B_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_d^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix} \begin{pmatrix} \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \boldsymbol{u}(3) \\ \vdots \\ \boldsymbol{u}(N) \end{pmatrix} + \cdots \quad [6.125]$$

Looking at the first line of this equation, it is certain that there cannot be a control at the first node where the initial conditions are applied. Currently, the first line reads:

$$\boldsymbol{x}(0) = \mathrm{I}_{7\times 7}\boldsymbol{x}_0 \quad [6.126]$$

which is a true statement. If there *was* a control at the first node, though, then this equation would read:

$$\boldsymbol{x}(0) = \mathrm{I}_{7\times 7}\boldsymbol{x}_0 + B_d\boldsymbol{u}(0) \quad [6.127]$$

which does not hold true anymore. This is thus considered to be the most likely explanation for the discrepancy, but it would not explain everything. If one considers the maximum or minimum mass flow per second and adds that to the solution found in this research, one ends up either above or below the result from [Açıkmeşe and Ploen 2007].

There is further reason to assume that the problem – if any – is on the side of Açıkmeşe and Ploen [2007] rather than that of this research. This is because a continuous time simulation has been developed in Chapter 8, that takes the discrete-time guidance command as an input and then integrates this independently with the equations of motion. Because this works and does not give any errors to speak of, it is proven that the implemented algorithm produces the correct output.

The following additional explanations have been considered:

- First it was tested whether the solver plays a role. After testing several different options (SDTP3, SEDUMI, ECOS) this hypothesis was rejected, because all solvers found the same solution.
- One possible explanation is that the problem is formulated differently in [Açıkmeşe and Ploen 2007]. However, it is not obvious how this could cause differences.
- Açıkmeşe and Ploen [2007] state that they use YALMIP as convex-optimization modelling-language, whereas in this thesis CVX was applied. This could explain the differences, but it is beyond the scope of this work to do a redundant software implementation.

TABLE 6.4 Parameters and results of unit test TGO-1.

INPUT				RESULTS			
a	b	tol	eps	x^*	$\cos x^*$	STEPS	$\varepsilon_{x^*}/\%$
3	4	10^{-8}	$\mathcal{O}(10^{-16})$	3.141 592 …	−1	9	$\mathcal{O}(10^{-8})$

A brief exchange with Prof. Açıkmeşe, the original author of the fundamental algorithm, did not help in resolving the issue either. In conclusion, despite the minor differences this unit test is *passed*.

6.6.3 Time-to-go optimization / Brent's method

In contrast to the previously tested functions, the verification of the time-to-go search algorithm is more straight forward. A helpful fact is that Brent's method is a long established technology, known to work, and no surprises are thus to be expected. Two tests will be performed:

UNIT TEST 6.3 Verification of the time-to-go search method, Brent's algorithm.

TGO-1 Find the optimum of $\cos x$ on the interval $\mathbf{(}3, 4\mathbf{)}$.
Expected outcome: The optimum should be an approximation of π up to the specified tolerance.

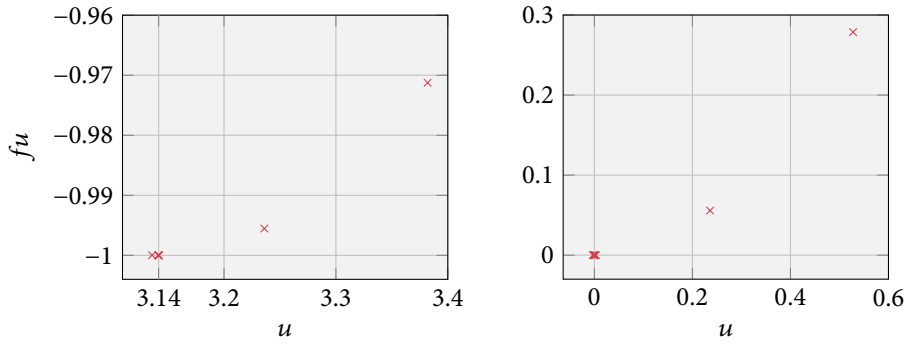
TGO-2 Find the optimum of x^2 on the interval $\mathbf{(}-1, 1\mathbf{)}$.
Expected outcome: The minimum should be rapidly found at $x = 0$.

Test case TGO-1 is also proposed in the documentation of the function `fminbnd()`, which is implemented in MATLAB [Mathworks 2014]. It thus presents a good test case, because comparison to a major software package is possible. Test case TGO-2 is designed to see whether parabolic interpolation works well: this should yield rapid convergence.

6.6.3.1 *TGO-1: optimizing the cosine*

This test optimizes the cosine function on the interval $\mathbf{(}3, 4\mathbf{)}$. Obviously, it has a minimum at $x = \pi$, and $\cos x = -1$. The input to the algorithm is shown in Table 6.4, see Algorithm 1 and Section 6.5.4 for details. The results are shown in the same table. The optimum is found with a precision on the order of 10^{-8} within just nine steps. All trials are shown in Figure 6.12(a). It is clearly visible how the search approaches the optimum rapidly.

To check whether the tolerance is handled properly, the algorithm was also executed for lower and higher tolerances. For lower tolerances, it was found that the approximation of π gets worse accordingly. For higher tolerances, the approximation could not be improved due to numerical limitations. Compared to MATLAB's `fmincon()`, no differences in the output can be observed. The test is thus considered to be *passed*.



(a) Cosine **(b)** Parabola

FIGURE 6.12 Verification of Brent's method for finding optima: the plots show all trials that have been made until convergence.

TABLE 6.5 Parameters and results of unit test TGO-2.

INPUT				RESULTS			
a	b	tol	eps	x^*	$\cos x^*$	STEPS	$\varepsilon_{x^*}/\%$
-1	1	10^{-2}	$\mathcal{O}(10^{-16})$	$-2.8 \cdot 10^{-17}$	0	8	$-2.8 \cdot 10^{-17}$

6.6.3.2 *TGO-2: optimizing a parabola*

The second unit test for Brent's method is very similar to the first, but optimizes x^2 to investigate how well parabolic interpolation works. To this end, a lax tolerance of only 10^{-2} is set. It is expected that the optimum is found at $x^* = 0$ despite this low tolerance, because the parabolic interpolation should find it straight away.

Table 6.5 shows the input and the result. All trials are printed in Figure 6.12(b). The optimum is found with much better precision than specified in only eight steps, which can only be possible with parabolic interpolation: a mere golden-section search would need much longer, because the interval is only decrease by a factor of $1 - \varphi$ per step. The test is thus *passed* with high certainty.

6.6.3.3 *Analysis of the final conditions*

One issue that might rise concerns is that several trials are performed in the vicinity of the optimum, visible by the several dots near 0 in Figure 6.12(b). To explain this, an analysis of the final conditions before terminating the search in Brent's method is in order.

At first thought, the repeated evaluation of points in the vicinity of the optimum seems redundant. However, the simple explanation is that the interval needs to be *bracketed*, as explained in Algorithm 1. This is required to be able to say with confidence that the optimum has been found up to the specified tolerance. Such a typical end-game situation for the algorithm is depicted in Figure 6.13, where u indicates the last trial, w the last optimum, and v the previous value of w as used in Algorithm 1 (Appendix C). Clearly, if a and b were not close enough to x^*, termination of the search would not be possible. Also, the drawing makes clear that if w, u, and v (or any other

three values) are available, the optimum should immediately be found by interpolation of $f = x^2$, which is indeed the case.

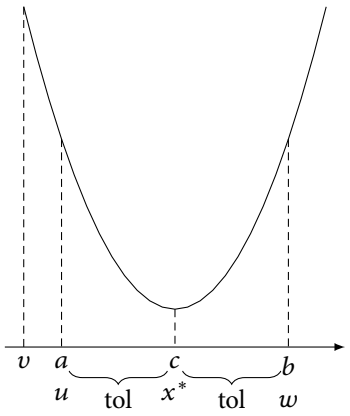


FIGURE 6.13 Typical end condition for Brent's algorithm: at least two trials near the optimum are necessary to ascertain that it is found with specified tolerance.

6.6.4 System-integration test

All the individual elements that make up convex guidance have now been verified. To make sure that these sub-functions work well together, a system-integration test needs to be performed. Two such tests will be made:

UNIT TEST 6.4 Verification of the integrated convex-guidance algorithm.

CVX-1	Find the optimal time-to-go and guidance command for a vertical-drop scenario. *Expected outcome:* The algorithm output should match known analytical results.
CVX-2	Find the optimal time-to-go and guidance command for the first reference test case from [Açıkmeşe and Ploen 2007]. *Expected outcome:* The results should match those given in the paper.

The first test adds certainty to the verification, because it presents a case independent from reference data. Both tests will be described in the following.

6.6.4.1 *CVX-1: vertical drop scenario*

When verifying algorithms and software implementations, it is very useful to have test cases that can be compared to known analytical results of the same problem. This greatly aids understanding and insight, and also makes the judgment of the unit test's outcome easier. For the analysis of convex guidance, a suitable test case is the *vertical-drop scenario*. In this, a lander is dropped from above the surface of a body from a known altitude, as shown in Figure 6.14. It can be proven that the optimum guidance-command for a soft landing is to have the lander free fall at first, until it reaches the point where it should engage its engines at maximum thrust, such that it *just* reaches the surface with exactly zero speed [Meditch 1964]. This particular epoch, the *switching time* t_s, can be established with analytically derived EOMs. If convex guidance works as expected, it should also command to turn on the lander's engines at exactly this point in time.

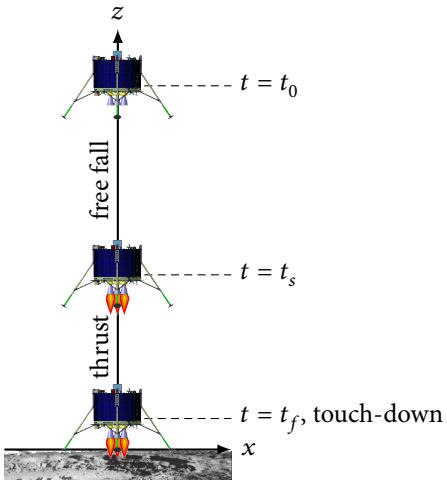


FIGURE 6.14 The vertical drop test case, unit test CVX-1. After falling freely, maximum thrust is engaged such that a soft landing occurs.

To determine the switching time t_s, the solution of the vertical EOMS of a mass-varying vehicle in a constant gravity field need to be known. As an additional restriction, the thrust is assumed to be constant. This is valid to assume, because the optimal guidance command is a bang-bang solution with maximum, constant thrust from t_s until t_f. The former are the same assumptions under which convex guidance has been derived. Fortunately, an analytical solution for this can be derived. The system to be solved is:

$$\ddot{r}(t) = \frac{T}{m(t)} + g, \quad \dot{m}(t) = \frac{T}{v_e} \tag{6.128}$$

where $v_e = g_0 I_{\text{sp}}$ is the engine's exhaust velocity. Meditch [1964] shows that the resulting trajectory is described by:[10]

$$r(t) = \frac{v_e m_0}{\dot{m}}\left(1 - \frac{\dot{m}}{m_0}t\right)\log\left(1 - \frac{\dot{m}}{m_0}t\right) + v_e t + \tfrac{1}{2}gt^2 + \dot{r}_0 t + r_0 \tag{6.129}$$

$$\dot{r}(t) = -v_e \log\left(1 - \frac{\dot{m}}{m_0}t\right) - gt + \dot{r}_0 \tag{6.130}$$

$$m(t) = m_0 - \dot{m}t \tag{6.131}$$

With this solution at hand, t_s can be found with a simple algorithm.[11] A function is written that computes the output of the EOMS for the constant mass, free-fall case (Equations [6.111] to [6.113]). At a given trial time time, the function then switches to computing the trajectory under thrust and with varying mass (Equations [6.129] to [6.131]). Now three cases can occur:

1. *Thrust is engaged too early:* The vehicle will start flying upward before reaching the surface. This case can be detected by noticing that $r(k+1) > r(k)$.

10 This is, in fact, a rare presentation of the analytical equations of motion taking into account *varying mass*.

11 Note that Meditch [1964] gives a way to analytically determine t_s. However, it was found that in practice this method was too imprecise, as it is based on approximations. Therefore, the more exact algorithm explained in this section was used.

TABLE 6.6 Input and output (where applicable) of the unit test CVX-1. The difference refers to the numerical solution with respect to the analytical solution, or the final conditions.

(a) Input

g, m/s^2	g_0, m/s^2	m_0, kg	I_{sp}, s	T_l, N	T_h, N	Δt, s	$r_{0,z}$, m
1.6249	9.807	100	300	0	243.735	0.25	1000

(b) Output and results

PARAMETER	ANALYTICAL	NUMERICAL	DIFFERENCE, %
$r_{f,x}$, m	0	0	0
$r_{f,y}$, m	0	0	0
$r_{f,z}$, m	0.04	0	0
$\dot{r}_{f,x}$, m/s	0	0	0
$\dot{r}_{f,y}$, m/s	0	0	0
$\dot{r}_{f,z}$, m/s	0.009	0	0
m_p, kg	3.231	3.233	0.06
t_{go}, s	59.48	59.50	0.03
t_s, s	20.48	20.50	0.11

2. *Thrust is engaged too late:* The vehicle reaches the surface ($r \leq 0$) with $\dot{r} > 0$.

3. *Thrust is engaged just right:* The vehicles reaches the surface with zero speed ($r(t_f) = 0$ m, $\dot{r}(t_f) = 0$ m/s).

Case 3 can be found by simply looping over t_s. Note that a certain tolerance needs to be placed on $r(t_f)$ and $\dot{r}(t_f)$, as it is unlikely to happen in a numerical framework that both will reach *exactly* zero at the same time. This is also a function of the step size that is used to compute the output. For all practical purposes, this tolerance has been set to 0.001 meters and meters per second.

The input to the unit test that shall determine t_s numerically is shown in Table 6.6. The scenario models a drop from 1 km above the surface of the Moon with no initial speed. The vehicle's wet mass is 100 kg, and its maximum thrust is one and a half times its weight ($T = m_0 g$). The table also shows the output of the analytical solution found with the algorithm given above, and the numerical output from convex guidance. The table shows that the algorithm to find t_s based on the analytical solutions terminates at $\dot{r} < 0.001$, at which the altitude is still 4 cm above the ground. This is the maximally achievable precision for a time-step resolution of 0.001 s. It should thus be reiterated that the solution is *not exact*, but approximated to maximum precision. A closed-form expression for t_s is not available. For this reason, no errors for the numerical expression can be established in the classical sense, as no perfect reference is available. The precision of the numerical solution can thus only be judged in terms of a difference.

Table 6.6 shows that for the given initial conditions and parameters, the engine is to be turned on after falling freely for $t_s = 20.478$ s. The total time of flight is $t_{\text{go}} = 59.48$ s. Convex guidance finds these values with a maximum difference of 0.11 %, which is very close to what would be expected from the analytical solution—which in itself is not exact. It must be pointed out that convex guidance can also only find this optimum

up to the specified time step, which is set 0.25 s for these simulations. The results thus match very closely. At the same time, convex guidance reaches the final conditions up to numerical precision, although in discrete and not in continuous time. This is better than what could be determined with the analytical EOMs.

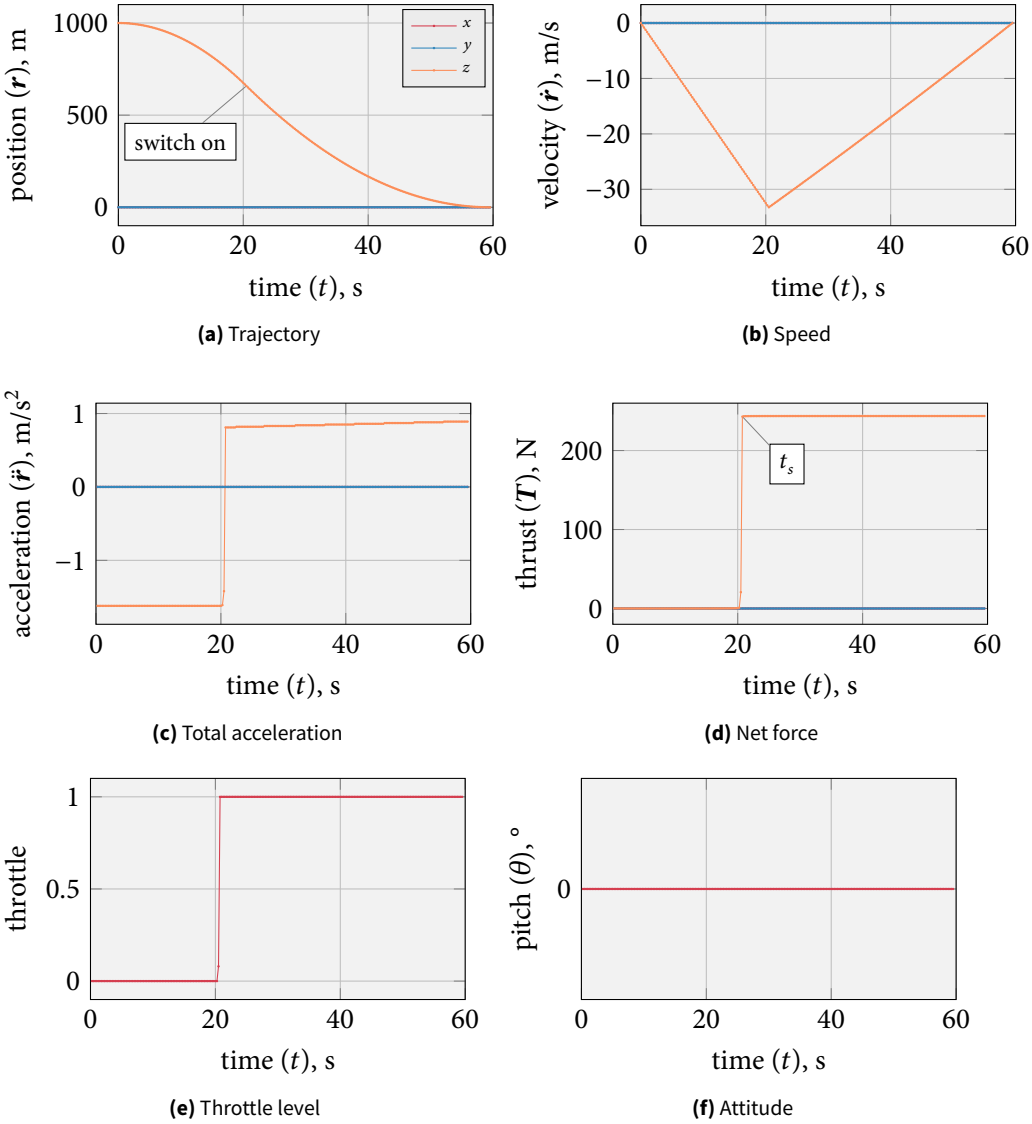
To further verify the output, the validity of the convex guidance output should also be visually inspected. This is shown in Figure 6.15. Subfigure (a) shows the trajectory as a function of time. As can be seen, the curve has an inflection point at $t = t_s$, as would be expected. It is also correctly shaped like a parabola. At the same epoch, the velocity starts to linearly decrease. Subfigure (d) shows that the thrust profile is of almost perfect bang-bang shape. However, one temporal node with $T < T_h$ is visible at 20.25 s. This occurs because of the limited temporal resolution. The switching time according to convex guidance has been considered to be the first point in time where $T > 0.9T_h$. Finally, Figure 6.15(c) shows that the vehicle indeed first drops at constant, vertical acceleration, and is then braked at increasing rate. This is an indication that the mass dynamics are modelled properly.

The final element that needs to be verified in the system integration test is the time-to-go search. The trials of Brent's method are shown in Figure 6.16. It can be seen that it converges steadily to the optimum and narrows down the interval in the process. This part of the algorithm also works well.

In conclusion, the output of convex guidance matches the expected values very closely. All individual sub-functions have been verified, and based on the system integration test CVX-1 no flaws could be identified. The test is thus considered to be *passed*.

6.6.4.2 *CVX-2: reference scenario*

The second test case for the system integration test resorts to the reference data given in [Açıkmeşe and Ploen 2007] again. As such, it is in fact the same test as FT-1 given in Section 6.6.2.1, with the added difficulty that the optimal time-to-go needs to be found. Therefore, the input data for CVX-2 is identical to that given in Table 6.3. The results of CVX-2 are also identical for the most part to FT-1 and are thus not repeated here. A small difference is that the optimal time-of-flight is found to be 73 s instead of 72 s as claimed by Açıkmeşe and Ploen [2007]. To check whether the golden-section search has worked properly in this case, this solution has been successfully verified by manually running th algorithm for time-to-gos in the vicinity of 72 s. The difference is caused by the discrepancies that have already been elaborated in Section 6.6.2.1 and is therefore no reason for concern. In conclusion, unit test CVX-2 is thus also *passed*.

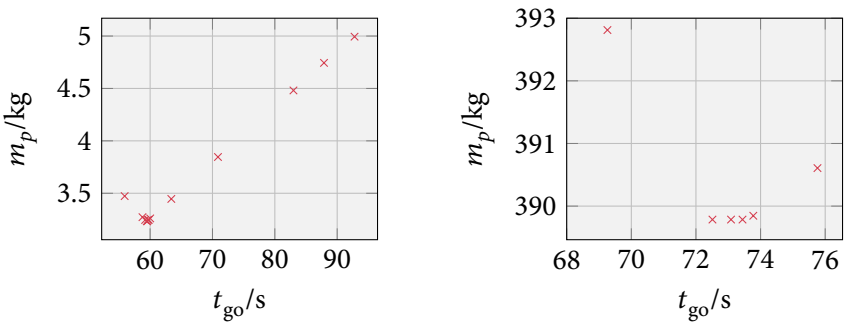


(a) Trajectory **(b)** Speed

(c) Total acceleration **(d)** Net force

(e) Throttle level **(f)** Attitude

FIGURE 6.15 System integration test: optimal result of the solution of the convex guidance SOCP in combination with Brent's search method for the time-to-go.

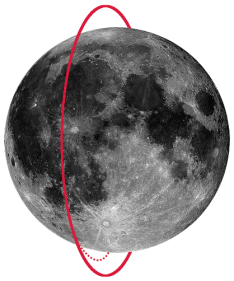


(a) Vertical drop scenario **(b)** Açıkmeşe reference case

FIGURE 6.16 Convergence to the optimal time-to-go in the system integration test with Brent's method and solving the SOCP.

Chapter 7

EXTENSIONS TO CONVEX GUIDANCE



> “Il semble que la perfection soit atteinte non quand il n’y a plus rien à ajouter, mais quand il n’y a plus rien à retrancher.”
>
> *Antoine de Saint-Exupéry (French aviator)*

Convex guidance has been derived and demonstrated to work in its basic form in the previous chapter. However, up until this point the promise of a “highly constrained” guidance law has not been fulfilled. The only two constraints integrated until now are thrust-bounds, as well as a constraint on the spacecraft mass.

To live up to the promise, the necessary modification for vision-based approaches and the main braking phase are made in this chapter. As such, Section 7.1 demonstrates the application of a no-subsurface-flight constraint. A glide-slope constraint, as also often applied in aviation, is derived in Section 7.2. The basics of constraining the spacecraft attitude are treated in Section 7.3. Section 7.4 then treats how this constraint can be used for an HDA-optimal approach phase. The possibility of adding spherical gravity-field models is discussed in Section 7.5. As there are still many more possible modifications beyond these extensions, Section 7.6 discusses a few of these. Finally, the chapter is concluded with a summary in Section 7.7.

7.1 NO-SUBSURFACE-FLIGHT CONSTRAINT

The convex guidance algorithm that was derived in the previous chapter successfully solves the TPBVP. It finds an optimal trajectory that leads from the initial conditions to the landing site. In the applied problem formulation, there are control constraints that ensure the feasibility of the trajectory with regard to the propulsion system. However, up until now no *path constraints* have been applied. This can potentially lead to problems, and one issue is discussed in this section: the need to avoid subterranean trajectories by use of a no-subsurface-flight constraint. Such a constraint, as proposed by Açıkmeşe and Ploen [2007], is derived in this section.

7.1.1 Need statement

The verification of convex guidance in the previous chapter was based on a reference scenario from [Açıkmeşe and Ploen 2007]. The verified results from this scenario are shown in Figure 6.11. As it turns out, for this scenario subsurface flight does indeed occur for 30 s in a flight time of 72 s. This is also shown more clearly in Figure 7.1. Naturally, any subsurface flight is impossible.

Before considering how subsurface flight can be avoided, it is worthwhile to consider why it happens here in the first place. The explanation lies in the initial conditions, that are violently wrong: as a matter of fact, the spacecraft is initially flying away at a speed of

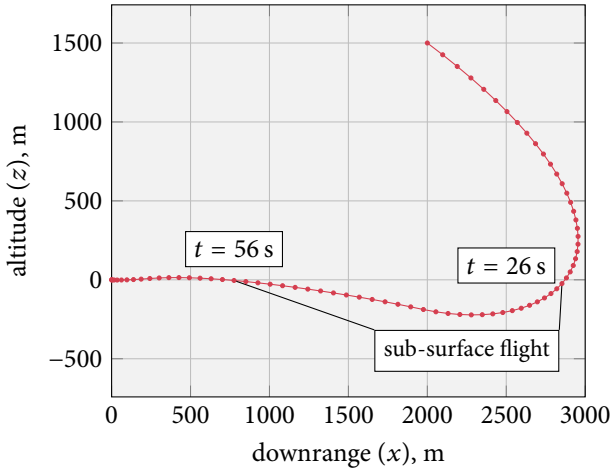


FIGURE 7.1 Trajectory for test case FT-1. The flight is subterranean for 30 s. See also Figure 6.11.

100 m/s downrange from the landing site. Guidance needs to command to cancel all of this velocity first, and then accelerate towards the landing site to reach it. However, the maximum available thrust is not high enough to cancel out the downrange and vertical velocity components quickly enough to avoid subterranean flight, and the trajectory thus ends up below the surface.

One may wonder whether such initial conditions are sensible to assume or not. In a realistic Moon-landing scenario, the spacecraft will initially *approach* the landing site, not fly away from it. Sub-surface flight is therefore much less likely to happen. However, a no-subsurface-flight constraint is still necessary for three main reasons:

1. Retargetings can lead to drastic changes in the trajectory by targeting a landing site that is offset from the original one. For the sake of robustness, the constraint should be applied in such cases.

2. Towards the end of the approach, when the lander gets closer to the landing site, subterranean segments can also occur. This naturally must also be avoided.

3. Implementing the constraint demonstrates the potential and flexibility of convex programming for guidance from an academic point of view.

In the end, the designer needs to trade whether the constraint is necessary or not, taking into account its impact on the execution time, *etc.*

7.1.2 Implementation

Within an SOCP framework the constraint can be implemented as follows. In basic mathematical notation, a no-subsurface-flight constraint is expressed by the following inequality for discrete time:

$$r_z(t_k) \geq 0\,\text{m}, \quad \forall\, t_k \in (t_1, t_N) \tag{7.1}$$

The constraint can be implemented in the usual fashion: the z-component needs to be extracted from the lumped solution vector $\tilde{\boldsymbol{X}}$ at each time step, to form a component-

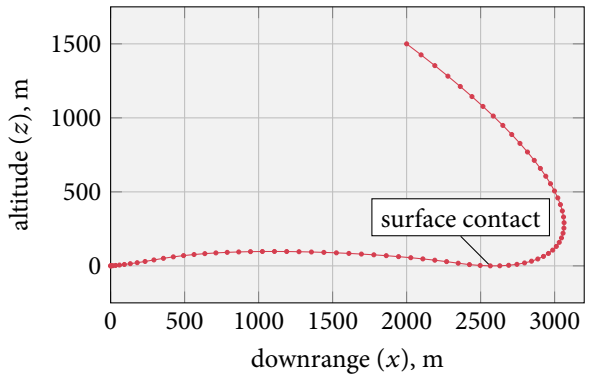


FIGURE 7.2 Trajectory with no-subsurface-flight constraint for the same boundary conditions as FT-1, Figure 7.1. The flight is not subterranean, but touches the surface at 36 s. Time-of-flight is 75 s. See also [Açıkmeşe and Ploen 2007, Figure 6].

wise vector inequality constraint. The matrix to extract r_z from the solution vector of a particular time step $\boldsymbol{x}(k)$ is:

$$E_{r_z,k} = \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & 1 & & & & \\ & & & 0 & & & \\ & & & & 0 & & \\ & & & & & 0 & \\ & & & & & & 0 \end{bmatrix} \tag{7.2}$$

To extract r_z at all states from $\tilde{\boldsymbol{X}}$, this matrix is stacked as before:

$$E_{r_z} = \begin{bmatrix} E_{r_z,k} & 0_{7\times7} & \cdots & 0_{7\times7} \\ 0_{7\times7} & E_{r_z,k} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0_{7\times7} & \cdots & \cdots & E_{r_z,k} \end{bmatrix}, \quad E_{r_z} \in \mathbb{R}^{(7\times N)\times(7\times N)} \tag{7.3}$$

The constraint can thus be written as:

$$E_{r_z}\tilde{\boldsymbol{X}} \geq 0 \tag{7.4}$$

7.1.3 Results

The trajectory resulting from applying the constraint to the same scenario as before is shown in Figure 7.2. The problem parameters are presented in Table 6.3. Clearly, the sub-surface segment has been eliminated. Compared to Figure 7.1, the trajectory now reaches out further downrange, which is explained by an increased thrust-vector component in the z-direction, leading to a slower deceleration in x-direction. This in turn leads to a longer optimal flight duration, which is 75 s compared to the previous 72 s. The longer t_{go} comes at the price of 1.1 % extra propellant than for the unconstrained trajectory (389.9 kg vs. 394.3 kg).

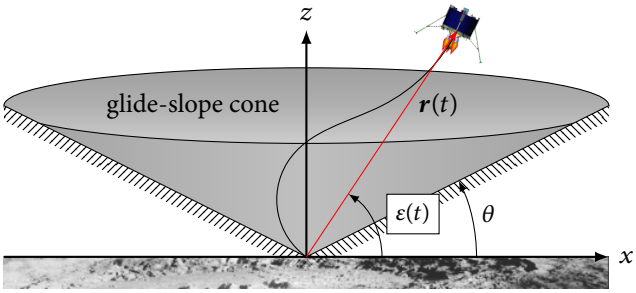


FIGURE 7.3 Illustration of the glide-slope constraint: the lander's elevation must be large than the glide-slope at all times..

Unfortunately, the result is also not flawless: 36 s into the flight, at 2565 m downrange, the vehicle touches the surface. While this fulfills the constraint, this is intolerable from a practical standpoint and calls for another constraint. This will be treated in the following section.

7.2 GLIDE-SLOPE CONSTRAINT

In the previous section it was shown how the constraint $r_z(t) \geq 0$ could avoid subterranean flight. However, this did not resolve the entire problem: surface contact did still occur at one point (Figure 7.2). A *glide-slope constraint* can be employed to finally resolve this issue.

7.2.1 Need statement

To explain what exactly a glide-slope constraint is, consider Figure 7.3. The lander is closing in on the landing site and has the position vector $\boldsymbol{r}(t)$. In context of a planetary landing, a glide-slope is a cone defined by the glide-slope angle θ, that has its vertex at the origin of the reference frame. In this case, the cone extends from the landing site frame $\mathcal{L}$ upwards. A glide-slope constraint ensures that the spacecraft always stays within this cone. In other words, the spacecraft's elevation angle with respect to the x-y plane at the landing site must always be larger than the glide-slope angle:

$$\varepsilon(t) \geq \theta, \quad \forall\, t \in (t_0, t_f) \tag{7.5}$$

Naturally, with such a constraint the lander can never touch the surface except at the landing site.

The glide-slope constraint can also be made use of for other applications. In the context of a GNC system with HDA in the loop, the glide-slope constraint can also be used as a means to avoid hazards. Consider a scenario where a boulder lies close to the targeted landing site, as shown in Figure 7.4. In such cases, the lander must avoid contact with the boulder at all costs, and pass over it at a safe distance to avoid plume interactions. A safe approach that avoids this can be flown by defining a glide-slope that passes over the rock.

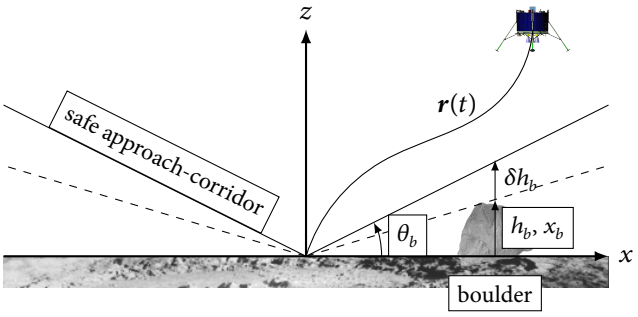


FIGURE 7.4 In context of an HDA framework, the glide-slope constraint can be used to establish a safe approach-corridor for avoiding boulders.

The advisable glide slope for boulder avoidance can be easily derived. In the two-dimensional case, consider that the boulder b lies at the coordinate $x = x_b$ in the $\mathcal{L}$-frame, and is h_b meters tall. These information can be obtained from the HD instrument output, such as LIDAR. Then it is advisable to let the glide-slope pass over the boulder by an additional safety distance δh_b. The glide-slope angle for a safe landing, that defines a safe approach-corridor, is then in three dimensions:

$$\theta_b = \arctan\left(\frac{h_b + \delta h_b}{\sqrt{x_b^2 + y_b^2}}\right) \qquad [7.6]$$

Of course, any piloting function must take into account that the landing site is sufficiently remote from any boulders. If tall boulders were close to the target, the safe glide-slope angle could quickly be too large and render a landing infeasible. However, such discussions remain beyond the scope of this report, which does not integrate the guidance algorithm with a full HDA system.

7.2.2 Implementation

The derivation and implementation of the glide-slope constraint follows the same principle as for all other constraints until now. First, a mathematical description is derived for continuous time in the form of a second-order cone-constraint of the form (cf. Equation [5.18]):

$$\|A_i \boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^{\mathsf{T}} \boldsymbol{x} + d_i, \quad i = 1, \ldots, m$$

This is then put into a stacked matrix-form to apply in discrete time.

To start with the continuous-time formulation, consider Figure 7.5. This sketch shows the three-dimensional geometry of the approach cone defined by the angle θ. The elevation of the lander at any given time is given by $\varepsilon(t)$. However, for the SOCP formulation it is more beneficial to use the co-elevation, as will become clear in a moment. It is defined by:

$$\tilde{\varepsilon}(t) = \frac{\pi}{2} - \varepsilon(t) \qquad [7.7]$$

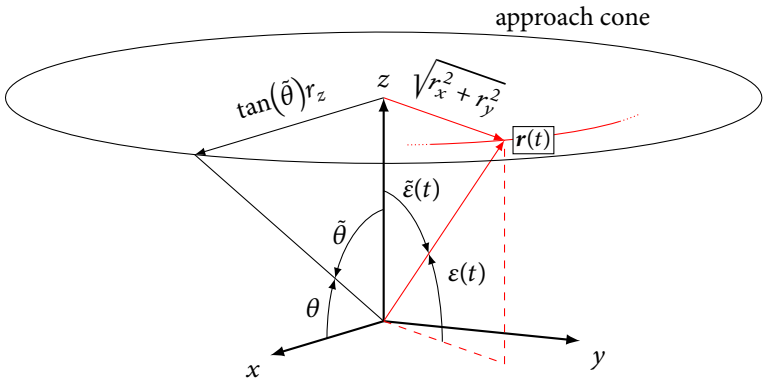


FIGURE 7.5 Geometry of the glide-slope constraint: the position vector of the lander must at all times remain within an approach cone, defined by the glide-slop angle.

The co-elevation at time t is:

$$\tan\tilde{\varepsilon} = \frac{\sqrt{r_x^2 + r_y^2}}{r_z} \tag{7.8}$$

Re-arranging, this equation reads:

$$\tan(\tilde{\varepsilon})r_z = \sqrt{r_x^2 + r_y^2} \tag{7.9}$$

Note that this is a parametrization for a cone of a given angle ε. This can be used to express the constraint. If the angle $\tilde{\varepsilon}$ is replaced by the co-angle of the glide-slope, one can write:

$$\sqrt{r_x^2(t) + r_y^2(t)} \leq \tan\left(\tilde{\theta}\right)r_z(t), \quad \forall\, t \in (t_0, t_f) \tag{7.10}$$

which evidently is a second-order cone constraint (cf. Equation [5.18]).

Although not directly expressed in angles anymore, this constraint allows for an intuitive interpretation. The right-hand side of the equation, $\tan(\tilde{\theta})r_z(t)$, is the dashed line extending from the tip of the z-axis to the local circle-segment in Figure 7.5. This line is the local radius of the cone. The left-hand side, $\sqrt{r_x^2(t) + r_y^2(t)}$, is the red arrow extending from the z-axis to the tip of the position vector. The constraint enforces that this dashed red line must always *be smaller than the local cone radius*, which implies that the position vector must lie *inside* of the cone.

The discretization works in the standard form of selecting the respective elements from the stacked solution (state) vector $\tilde{\boldsymbol{X}}$. For the sake of brevity, the individual selection matrices will not be written out again. See Equation [7.2] for an example. The SOCP constraint in discrete time reads:

$$\boldsymbol{e}_{r_z,\tilde{\theta}}^{\mathsf{T}} E_{k,\tilde{\boldsymbol{X}}} \tilde{\boldsymbol{X}} \leq \left\| E_{x,y} E_{k,\tilde{\boldsymbol{X}}} \tilde{\boldsymbol{X}} \right\|, \quad k = 1, 2, \ldots, N \tag{7.11}$$

$$\boldsymbol{e}_{r_z,\tilde{\theta}}^{\mathsf{T}} = \begin{pmatrix} 0 & 0 & \tan\tilde{\theta} & 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.12}$$

where $\boldsymbol{e}_{r_z,\tilde{\theta}}$ is the vector that multiplies the r_z component by the co-glide-slope angle, $E_{k,\boldsymbol{X}}$ is the matrix that selects the state vector of the current time step out of the stacked solution vector, and E_{xy} is the matrix that extracts the r_x and r_y components from that state vector. The optimization problem derived from this constraint is shown in Problem 7.

▶ PROBLEM 7 Convex approach-phase guidance-problem with glide-slope constraint. Based on Problem 6.

$$
\begin{aligned}
&\text{minimize } \boldsymbol{e}_{N\sigma\Delta t}^{\mathsf{T}}\boldsymbol{U} \\
&\text{subject to} \\
&\|E_{\boldsymbol{\tau}}E_{k,\boldsymbol{U}}\boldsymbol{U}\| \leq \boldsymbol{e}_{\sigma}^{\mathsf{T}}E_{k,\boldsymbol{U}}\boldsymbol{U}, \quad k = 1, 2 \ldots, N \\
&\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right) + \tfrac{1}{2}\left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right)^2\right] \leq E_{\sigma}\boldsymbol{U} \leq \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{2N}\right)\right] \\
&\boldsymbol{z}_{2N} \leq E_z\tilde{\boldsymbol{X}} \leq \boldsymbol{z}_{1N} \\
&\boldsymbol{e}_{r_z,\tilde{\theta}}^{\mathsf{T}}E_{k,\tilde{\boldsymbol{X}}}\tilde{\boldsymbol{X}} \leq \left\|E_{x,y}E_{k,\tilde{\boldsymbol{X}}}\tilde{\boldsymbol{X}}\right\|, \quad k = 1, 2, \ldots, N \\
&\boldsymbol{e}_{r_z,\tilde{\theta}}^{\mathsf{T}} = \begin{bmatrix} 0 & 0 & \tan\tilde{\theta} & 0 & 0 & 0 & 0 \end{bmatrix} \\
&E_{\boldsymbol{x}(N)}\boldsymbol{X} = \begin{bmatrix} \boldsymbol{r}_f^{\mathsf{T}} & \dot{\boldsymbol{r}}_f^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}
\end{aligned}
$$

7.2.3 Results

A plot of the trajectory with glide-slope constraint is shown in Figure 7.6. This is the outcome of a simulation for the same initial conditions as in Figure 7.2, but with an additional arbitrary glide-slope angle of 4° (see also Table 6.3). As indicated by the line, the trajectory stays within the approach cone, and does not touch the surface anymore. The flight path now extends even more downrange, for the same reason as explained in Section 7.1.3: the thrust force component opposing the downrange velocity is now necessarily even smaller than before. The flight duration has increased by another 6 s to 81 s. Propellant usage has gone up by 7.6 kg to 401.6 kg (1.85 %, see Section 7.1.3). In conclusion, the glide-slope constraint has been successfully implemented and can be used to avoid surface contact under specific initial conditions. It also has the potential to avoid boulders in an HDA framework. However, at least in this case, a slight propellant penalty cannot be avoided.

On a final note, one may now wonder why a no-sub-surface flight constraint would still be necessary when applying a glide-slope constraint. In practice, the algorithm should remain as lean as possible to spare resources. It certainly makes sense to disregard the no-subsurface-flight constraint throughout the major part of the trajectory. However, observations have shown that slight sub-surface flight might actually occur sometimes near the very end of the flight. It could therefore make sense to keep applying the constraint throughout the last few time steps.

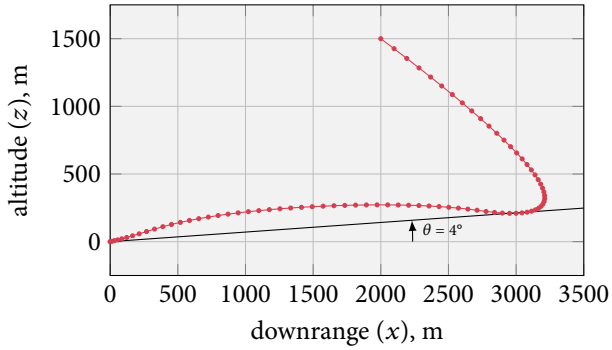


FIGURE 7.6 Trajectory with glide-slope constraint for the same boundary conditions as FT-1, Figure 7.1. The flight remains within the cone. Time-of-flight is 81 s. See also [Açıkmeşe and Ploen 2007, Figure 8].

7.3 THRUST-POINTING CONSTRAINT

Two path constraints have been imposed in the preceding sections. However, to yield the best possible physically feasible trajectory, further control constraints can also be required. A basic thrust-pointing control-constraint will be derived in this section. The ground work for this has been laid in [Açıkmeşe and Ploen 2007; Açıkmeşe et al. 2013], on which much of the theoretical part of this section is based.

7.3.1 Need statement

The most important need for a thrust-pointing constraint is to maintain the physical feasibility of a trajectory with respect to the lander hardware. To give a rationale for this, it must first be understood that a thrust-pointing constraint is, in fact, identical to an attitude constraint. For all intents and purposes, considering only the three *translational* degrees of freedom (DOFS) suffices for the treatment of guidance in this report. This means, in turn, that the vehicle is modelled as a point mass, where the rotational motion may be considered to be completely decoupled from the translational dynamics. As explained before (see Section 3.1.2), "[this] is a common assumption" [Açıkmeşe et al. 2013, p. 2106]. Under these circumstances, the attitude of the vehicle can be derived from the commanded thrust vector. Assuming that the propulsive plant of the lander is rigidly attached to it, a guidance command to thrust into a certain direction directly translates to pointing the engine in this same direction. In a six-DOF scenario, this would amount to an input from guidance to the attitude-control system, which would realize the attitude for the resulting thrust vector (Chapter 4). Thrust-pointing and attitude constraints are thus equivalent.

The two main needs for the constraint are depicted in Figure 7.7. From a vehicle point-of-view, upside-down flights must be avoided at all times. Such an attitude would be dangerous from several points of view, and could cause, for example, excessive propellant sloshing or problems with the engine feed-system. In the context of HDA and vision-based navigation, it is also necessary to maintain a line-of-sight with the landing region, such that the area around it remains in the FOV of the landing instruments. If that was not the case, the HDA system could not detect hazards, and the navigation

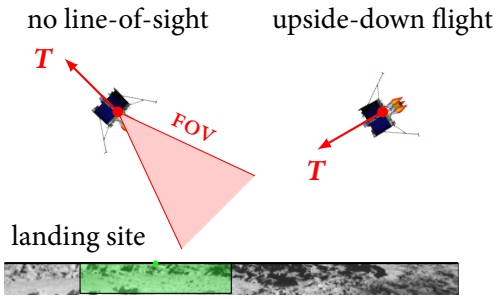
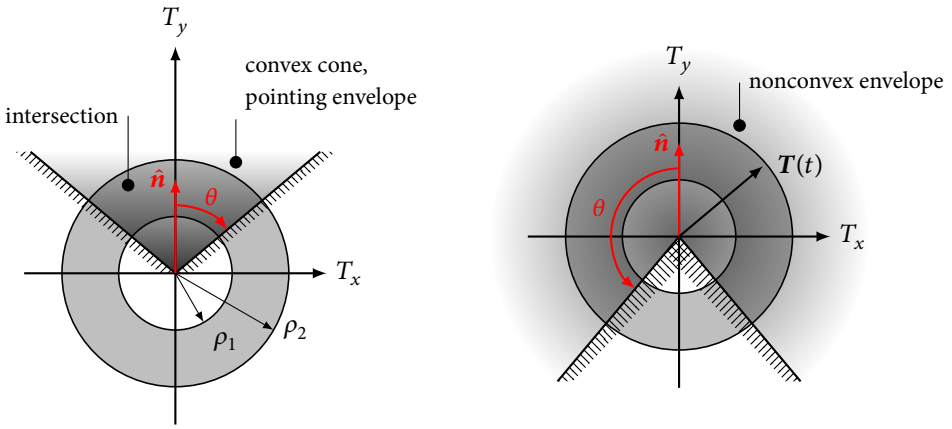


FIGURE 7.7 Two problems avoidable with attitude constraints.



(a) convex cone, $\theta \in (0, \pi/2)$ **(b)** nonconvex case, $\theta \in)\pi/2, \pi($

FIGURE 7.8 Mathematical formulation of the thrust-pointing constraint. See also [Açıkmeşe et al. 2013].

system would not see features for relative navigation. There is thus a need for an attitude constraint to maintain a pitch above the local horizon, and to keep a line-of-sight with the landing region. Here, only the basics have been discussed. The more complex constraint for vision-based applications is discussed in Section 7.4.

7.3.2 Implementation

Within the convex programming framework, the constraint can be derived by starting from the thrust-bound control-constraint. The fundamental idea of the thrust-pointing constraint is to first define a direction vector in inertial space. This indicates the general direction into which the thrust vector may point. Secondly, an angle θ is defined, that determines how much the thrust vector $\boldsymbol{T}$ may deviate from the direction vector $\hat{\boldsymbol{n}} \in \mathbb{R}^3$, $\|\hat{\boldsymbol{n}}\| = 1$. This is depicted for the planar case in Figure 7.8(a), where the thrust-pointing envelope defined by $\hat{\boldsymbol{n}}$ and θ has been superimposed over the thrust-bound constraint. Clearly, for $\theta \leq \pi/2$ this constraint defines a convex cone, which is required for convex programming. Combined with the thrust bounds, the intersection of the cone and the annulus in the picture defines the set of admissible thrust vectors.

A major problem arises if $\theta > \pi/2$, as shown in Figure 7.8(b). In this case, the pointing envelope is clearly nonconvex, and the constraint would thus not be suited for convex guidance anymore. However, this nonconvexity can be avoided in a similar fashion as

before, which will become clear from the mathematical treatment of the problem that is explained next.

The derivation of the constraint in continuous time is straight forward. Consider the case that the thrust vector is aligned with one of the boundaries in Figure 7.8(a). Then from the vector dot-product it is clear that:

$$\hat{\boldsymbol{n}}^\mathsf{T}\boldsymbol{T} = \|\boldsymbol{T}\|\cos\theta \qquad [7.13]$$

To allow for play in the thrust vector, this is written as an inequality constraint:

$$\hat{\boldsymbol{n}}^\mathsf{T}\boldsymbol{T} \geq \|\boldsymbol{T}\|\cos\theta \qquad [7.14]$$

With $\boldsymbol{T} \in \mathbb{R}^3$ and the Euclidean norm of the thrust, this is a second-order convex cone-constraint, or ice-cream cone.

To resolve the nonconvexity for $\theta > \pi/2$, recall that it was shown for the lossless convexification of the thrust-bound constraint that the norm of the thrust must be equal to the scalar slack variable Γ for all optimal trajectories (cf. Section 6.1):

$$\|\boldsymbol{T}(t)^*\| = \Gamma(t)^*$$

Substituting this back into Equation [7.14], one obtains:

$$\hat{\boldsymbol{n}}^\mathsf{T}\boldsymbol{T} \geq \Gamma\cos\theta \qquad [7.15]$$

which is a convex constraint for $\theta \in \mathbf{(}0, \pi\mathbf{)}$. A detailed treatment of this convexification is beyond the scope of this report, and the interested reader is invited to study the full proof in [Açıkmeşe et al. 2013] for details.

To transcribe the constraint to discrete time, the standard approach using selection matrices and stacked vectors is employed. Skipping the lengthy documentation of the selection matrices, the discretized constraint reads:

$$\hat{\boldsymbol{n}}^\mathsf{T}E_{\boldsymbol{r}}E_{k,U}\boldsymbol{U} \geq \boldsymbol{e}_\sigma{}^\mathsf{T}E_{k,\boldsymbol{U}}\boldsymbol{U}\cos(\theta), \quad k = 1, 2, \ldots, N \qquad [7.16]$$

which is the final formulation.

7.3.3 Results

To demonstrate this constraint, consider again the standard reference scenario that has been applied to the previous constraints as well (cf. Table 6.3). On top of the additional glide-slope constraint of 4° and the no-subsurface-flight constraint, the maximum thrust-pointing constraint is now enforced with a time-invariant vertical direction-vector and cone angle:

$$\hat{\boldsymbol{n}} = [0 \quad 0 \quad 1], \quad \theta_{\max} = 35° \qquad [7.17]$$

For simplicity, this formulation implies that $\hat{\boldsymbol{n}}$ and θ are time invariant. However, these variables may also change for each time step, if desired.

The resulting trajectory and relevant guidance output is plotted in Figure 7.9. As shown by the arrows in Figure 7.9(a), which represent the thrust vector each two seconds, and the plot of the pitch angle Figure 7.9(d), the attitude is successfully constrained to a maximum deviation of 35° from the vertical. The attitude information has been derived from the thrust components, for which the graphs are shown in Figure 7.9(b). Corresponding to the constrained pitch, it can be seen that the planar components are constant and max out for the first 24 s. The components then change when the lander starts flying towards the landing site, which is also well reflected in the attitude plot Figure 7.9(d). The bang-bang thrust-profile, shown in Figure 7.9(c), remains unaffected, which indicates that the optimality of the trajectory is retained despite the newly introduced constraint.

An essential verification step is to show that the convex relaxation still holds (see [Açıkmeşe et al. 2013] for a proof). This can be checked by comparing the scalar relaxation-variable σ to the norm of the thrust-command variable $\boldsymbol{\tau}$, which must be identical $\|\boldsymbol{\tau}\| = \sigma$. To this end, the difference of these two quantities is shown in Figure 7.9(e). Given the minimal order of magnitude of the error, ranging around 10^{-5}, it is safe to say that the convex relaxation remains lossless, and thus still holds. The solver output for the convex program that defines guidance is still valid, and remains optimal.

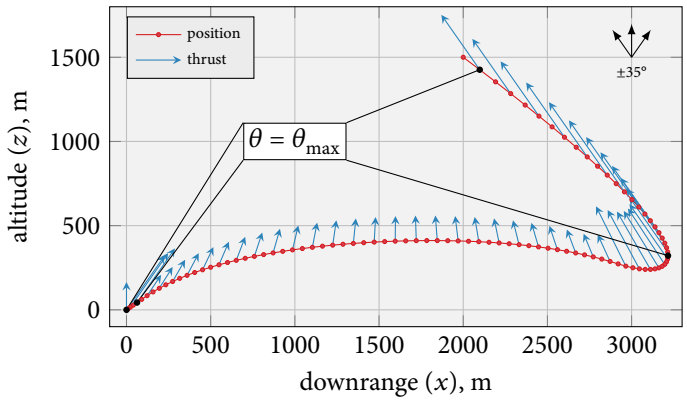
7.3.4 Limitations of the Lossless Relaxation

Although it was shown in the last section that the convex relaxation holds for the test case, see Figure 7.9(e), it must be stressed that this is *not* the case for *all* imaginable scenarios. In fact, Açıkmeşe and Ploen [2007] proof that the convex relaxation can fail, and does not necessarily hold anymore with the attitude constraint. If the relaxation fails, this will result in a physically meaningless solution, where the slack variable does not equal the norm of the thrust anymore, $\Gamma(t) \neq \|\boldsymbol{T}(t)\|$.

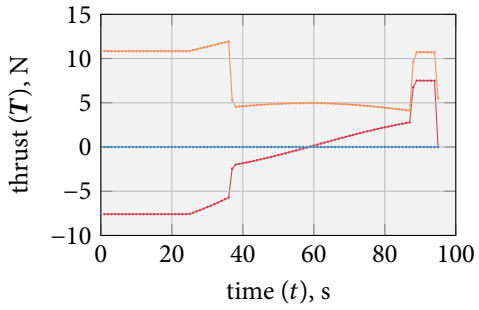
A test case where the relaxation fails is the previously successfully solved vertical drop case, see Section 6.6.4.1. As demonstrated in that section, the unconstrained optimal solution is to let the vehicle fall freely at first, and then cancel the velocity by thrusting at maximum throttle such that it just reaches zero speed at the surface. The optimal solution changes if the problem is constrained. If the lower thrust-bound is nonzero, the spacecraft will first thrust towards the landing site at minimum thrust, then turn around instantaneously and burn a bit longer at minimum thrust, and ultimately null all speed by thrusting maximally. This emulates the unconstrained case as well as possible by reducing the time-of-flight, where shorter time-of-flights generally mean less propellant usage.

A guidance solution for this scenario, without attitude constraint, is shown in Figure 7.10. The initial conditions are 5 km altitude at zero speed, and the vehicle is to soft-land right below it. All other vehicle parameters and conditions remain identical to test case FT-1, see Table 6.3. The plots perfectly mimic the expected trajectory just explained.

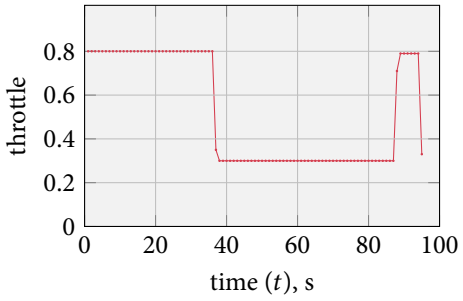
If an attitude constraint is now applied that does not allow the lander to thrust upside-down anymore, the situation changes dramatically. Due to the non-zero thrust bound, the optimal guidance solution will be very unintuitive, almost bizarre. Consider the case where the nominal pointing-vector is vertical, $\hat{\boldsymbol{n}} = \hat{\boldsymbol{k}}$, and the attitude is constrained to remain above the local horizontal, $\theta = 90°$. Then the optimal scenario



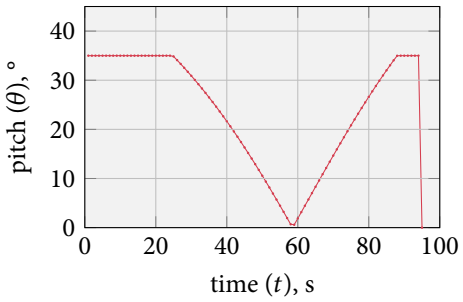
(a) Trajectory with thrust vector in two second ticks



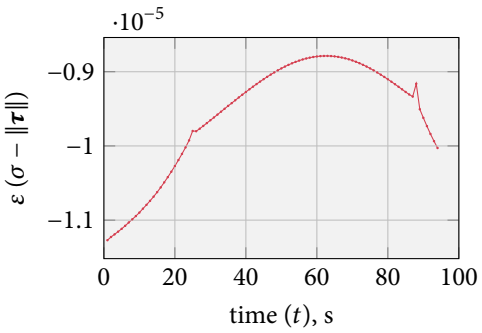
(b) Thrust components



(c) Throttle level

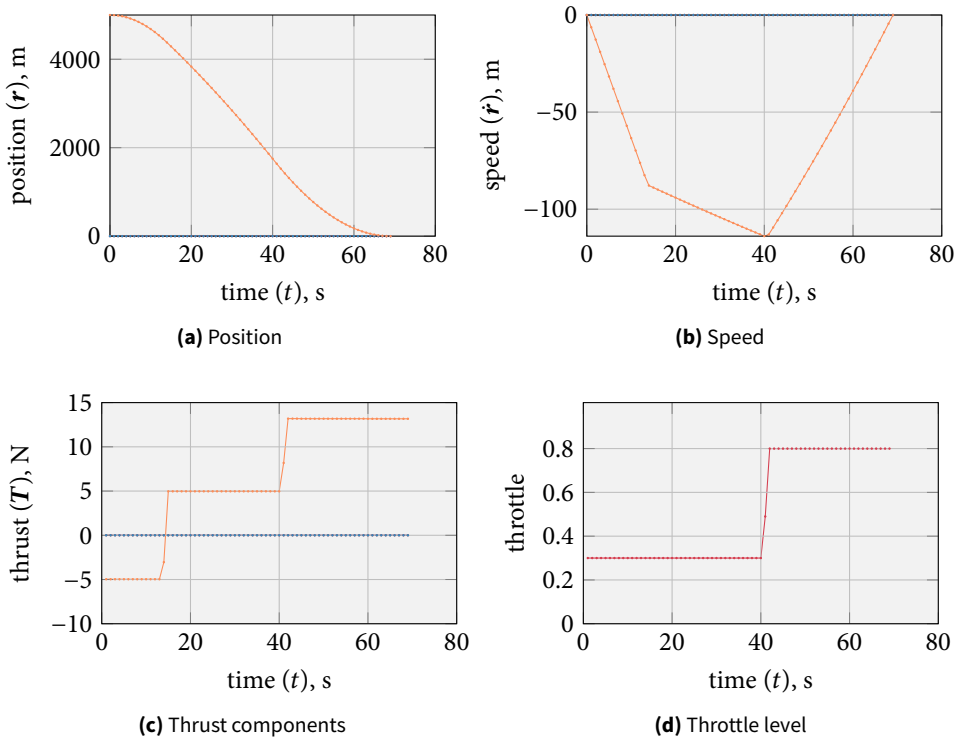


(d) Attitude



(e) Relaxation error

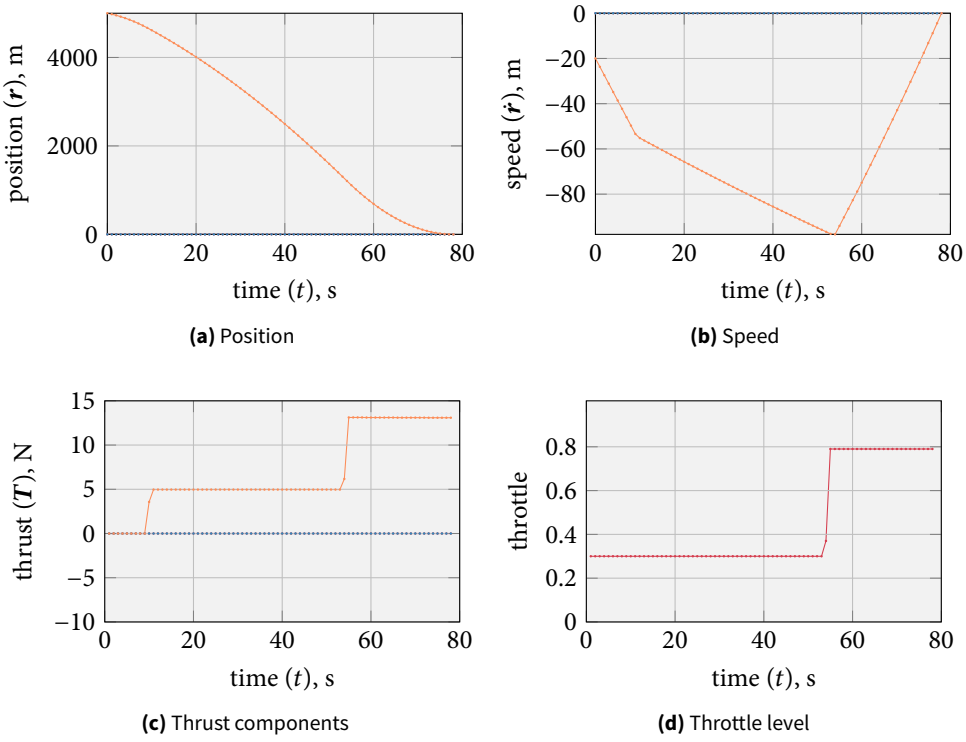
FIGURE 7.9 Guidance output for the same initial and boundary conditions as before (FT-1). The thrust-pointing constraint is successfully enforced at 35°. Time-of-flight is 95 s.



(a) Position **(b)** Speed

(c) Thrust components **(d)** Throttle level

FIGURE 7.10 Unconstrained vertical-drop test-case with nonzero lower thrust bound. The convex relaxation holds, and the command is a typical min-max profile. Time-of-flight is 69 s.



(a) Position **(b)** Speed

(c) Thrust components **(d)** Throttle level

FIGURE 7.11 Vertical-drop test-case with nonzero lower thrust bound, constrained by an attitude cone with vertical direction vector and 90° pointing angle. The constraint is not realized, and the convex relaxation fails. Time-of-flight is 78 s.

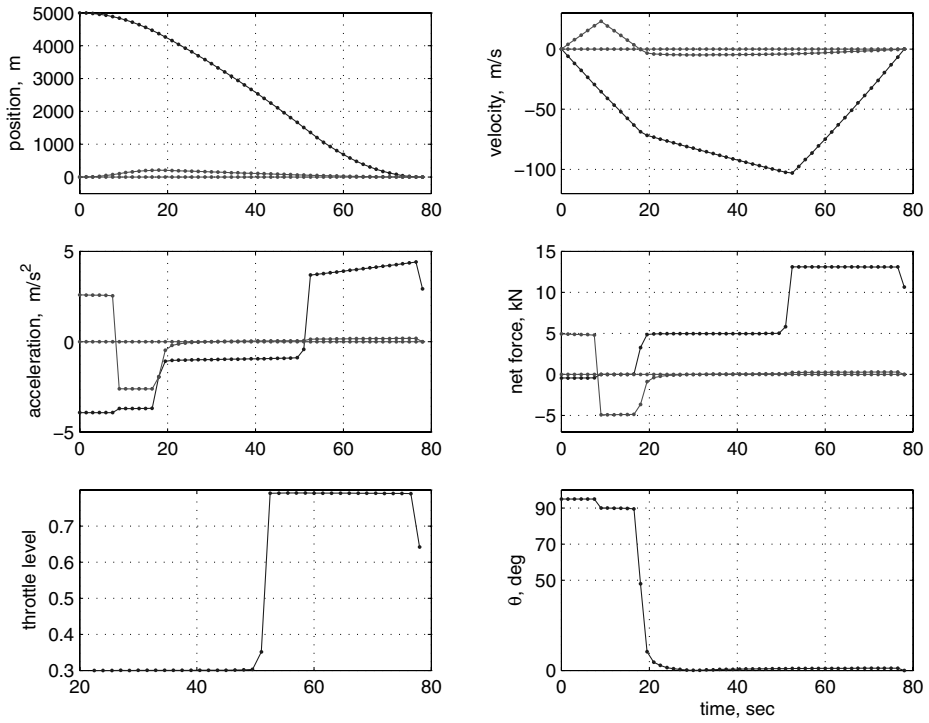


FIGURE 7.12 Reference data taken from [Açıkmeşe and Ploen 2007, p. 1365]. The thrust-pointing constraint works in this extreme scenario, by working around the convex relaxation.

will be to first thrust along the local horizontal at minimum throttle, *away from the landing site*. This is because in that case, the spacecraft can pick up as much vertical speed as possible to reduce the time-of-flight, will still thrusting at minimum throttle. Next, this horizontal speed is eliminated again by thrusting in the opposite direction. At some point, the attitude will jump to vertical again, ultimately nulling all speed at maximum thrust.

The guidance command for this constrained vertical-drop scenario is shown in Figure 7.11, only now with an initial altitude rate of −20 m/s to be consistent with the reference data in [Açıkmeşe and Ploen 2007, p. 1356]. The speed plot in Figure 7.11(b) immediately shows that no horizontal speed is ever gained. This is confirmed by the graphs for the thrust components in Figure 7.11(c), which only depicts a nonzero vertical component. Most importantly, though, comparing the thrust component to the throttle level in subfigure (d) shows that the convex relaxation *does no longer hold*. For the first nine seconds of the flight, there is no thrust, yet a throttle level of 30 % is commanded. This is contradictory, and means that $\Gamma(t) \neq \|\boldsymbol{T}(t)\|$. In conclusion, convex guidance fails in this very extreme scenario.

As a matter of fact, though, methods exist to resolve this violation of the relaxation, and make it lossless again. Açıkmeşe and Ploen [2007, p. 1363] propose a heuristic, that relaxes the pointing constraint only when "it cannot be simultaneously satisfied with the thrust magnitude constraint". The results from that paper are shown in Figure 7.12. The graphs show that the lander indeed first thrusts horizontally, and then switches to a vertical attitude. A small horizontal speed remains after this maneuver, which could be explained by small errors made by the heuristic.

Another approach to the lossless relaxation is proposed in the more recent paper [Açıkmeşe et al. 2013]. This considers the guidance law with a system matrix that corrects for the rotational motion of the central body. A theorem is proven that states that the lossless relaxation holds if the rotation vector of the central body is slightly altered, depending on the nominal pointing vector. However, no results for the test case discussed above are shown in this paper, so the validity of the results cannot be finally verified.

Although the problem can most likely be resolved with these two methods, one may ask whether the complications of implementing are worth the effort at this point. The constraint vertical-drop scenario poses extreme initial conditions, and stresses the algorithm to its limits. It is, however, also unrealistic. On a positive note, the algorithm was already shown to work even for the extreme test case in the previous section, where the spacecraft flies away from the landing site at high speed initially. If the lossless relaxation works even under such conditions, it is ever the more likely that it will also work under the much less extreme condtions of a nominal – or slightly prerturbed – Moon landing approach. For this reason, no effort will be spent on altering the method, and this work will be left for future projects.

To summarize, a basic thrust-pointing constraint has been derived and demonstrated in this section. However, further refinements are desirable, and will be made in the following section.

7.4 SENSOR-OPTIMAL THRUST-POINTING

The previous section introduced a basic methodology to constrain the attitude of a spacecraft by defining a cone of admissible thrust-vectors around a nominal direction-vector. However, the direction vector was assumed to be time-invariant for the sake of simplicity. In this section, the work of Açıkmeşe and Ploen [2007] is extended and adapted to landing scenarios that use vision-based technologies. This will be done by applying a different thrust-pointing constraint at each point in time.

7.4.1 Need statement

It has been pointed out that the attitude constraint can be used to avoid upside-down flight and to optimize the trajectory for HD and navigation sensors. The former can already be achieved by specifying a constant vertical direction vector $\hat{\boldsymbol{n}} = \hat{\boldsymbol{k}}$ and a maximum pointing angle of $\theta \leq \pi/2$. However, the basic constraint formulation with a constant direction-vector has limited applications when the goal is to tailor the trajectory for vision-based technologies. This is because the attitude of the lander naturally changes throughout its flight. While the attitude will be near horizontal at approach gate, it is desired to be vertical at touchdown. No single attitude constraint can thus ever lead to major improvements in the visibility of the landing site. Consequently there is a need to extend and improve the constraint. A smart way to accommodate successive, sensor-optimal thrust-pointing constraints is proposed in this section.

(a) Sensor installation points

(b) Pointing envelope

FIGURE 7.13 Relation between spacecraft, a sensor's FOV, and the attitude constraint.

7.4.2 Implementation

The development of a sensor-optimal thrust-pointing constraint entails several sub-problems. A discussion of the problem's geometry, the nominal direction vector, and the maximum pointing angle is the topic of this section. The derivation will be done step-by-step as follows:

1. First, the geometry of the sensor in relation to the spacecraft and the attitude (determined by the thrust) is defined.
2. Next, the optimal flight-geometry is defined.
3. Third, the optimal flight-geometry and the spacecraft geometry are combined to derive the optimal viewing constraint conditions.
4. Based on this the nominal direction-vector is derived.
5. Finally, it is shown how the viewing-angle can be calculated given a thrust vector.

7.4.2.1 *Sensor-vehicle geometry of the constraint*

It is essential to understand how the attitude constraint is related to any sensor and the spacecraft itself. As explained before, the thrust-pointing constraint directly translates to an attitude constraint, because the lander's engine is assumed to be rigidly fixed to the vehicle. The same argument holds for any instrument: if a sensor is rigidly fixed to the lander, the direction into which it points, and the resulting FOV, are directly determined by the attitude. Consequently, any sensor's FOV is indirectly determined by the thrust-vector command of guidance. Rather than constraining the FOV directly, it is thus necessary to constrain the thrust vector instead.

A typical way that a sensor, such as a camera, would be installed on a lander is shown in Figure 7.13(a) in blue. The sensor is rigidly attached to side of the spacecraft and points downwards under an angle with respect to the vertical. This angle can be optimized as to yield a maximal landing-site visibility throughout the approach. The optimal installation angle, and also the displacement distance from the axis of symmetry of the lander, are functions of the instruments that are used, as well as how the approach trajectory is tuned.

The assumed FOV in this research is shown in Figure 7.13(a) in red. The displacement from the lander's axis of symmetry is ignored. This is a valid assumption, considering

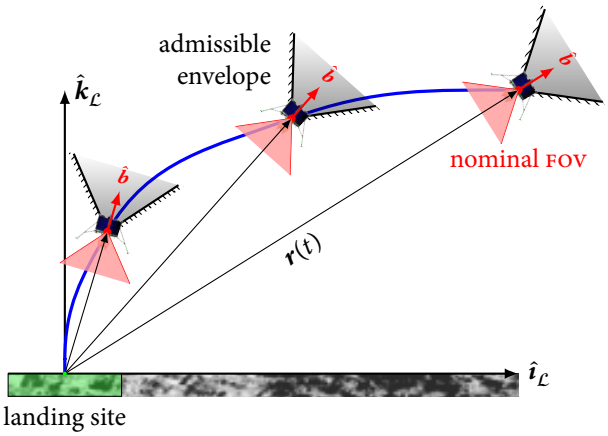


FIGURE 7.14 Sensor-optimal approach phase. The boresight vector is determined by the position vector.

that the lander is hundreds of meters remote from the landing site while surveying it for hazards and navigation features. In high-fidelity simulations later on this offset should be taken into account. The angular offset from the thrust-vector/axis of symmetry is called the *sensor angle* and is denoted by γ, which is measured from the boresight vector $\hat{\boldsymbol{b}}$ to the unit thrust-vector $\hat{\boldsymbol{T}}$. The importance of taking into account the sensor angle cannot be overstated. It very strongly influences the feasibility of the viewing constraint.

The relation between the thrust-pointing constraint and the FOV of a sensor in this set-up is shown in Figure 7.13(b). The pointing envelope for the sensors is a mirrored, but actually slightly larger pointing cone. From the figure it is clear that for a given maximum pointing angle θ, the effective FOV actually is:

$$\text{FOV}_{\text{eff}} = 2\theta + \text{FOV} \qquad [7.18]$$

7.4.2.2 *Optimal flight geometry*

The most important requirement for any landing sensor during this phase is that the landing site must remain in its FOV. If the line-of-sight to the landing region is lost, relative navigation and thus precision landing is hindered, and hazards cannot be detected. To keep the spacecraft pointed at the landing site, the method illustrated in Figure 7.14 is proposed, where the sensor angle γ has been assumed to be 0° for simplicity of the drawing. At each point in time, the boresight vector is chosen to be collinear with the radius vector at that point:

$$\hat{\boldsymbol{b}}(t) = \hat{\boldsymbol{r}}(t) = \frac{\boldsymbol{r}(t)}{\|\boldsymbol{r}(t)\|} \qquad [7.19]$$

In this configuration, the sensors always look directly at the landing site and maintain it in the center of their FOV.

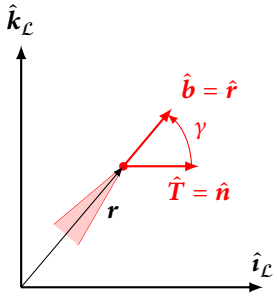


FIGURE 7.15 Relation between the optimal pointing direction and the spacecraft geometry. The boresight needs to be aligned with the position vector. The nominal pointing direction is offset by the sensor angle, which defines the direction vector for the attitude constraint.

7.4.2.3 *Combining flight and spacecraft geometry*

The spacecraft and optimal-approach geometries have been presented in Figure 7.13 and Figure 7.14 respectively. These two geometries have been combined in Figure 7.15. This drawing makes clear that to maintain a suitable boresight vector, the attitude constraint actually has to define a nominal direction vector that points elsewhere.

Not shown in this drawing is the allowable pointing cone around $\hat{\boldsymbol{n}}$, that was given by the constraint:

$$\hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{T}} \geq \Gamma \cos\theta \tag{7.20}$$

From this it can be realized that if the landing site shall remain in the FOV at all times, the cone must be defined such that it is not larger than half the field of view:

$$\theta = \frac{\text{FOV}}{2} \tag{7.21}$$

In most cases, simply keeping the landing site in the FOV is not enough. Rather, it should remain in the center of the FOV, aligned with the boresight. This can be achieved by constraining the pointing cone to a small tolerance value:

$$\theta \ll \frac{\text{FOV}}{2} \tag{7.22}$$

For the analysis presented in Chapter 9 the tolerance has been set to 5°. Note that directly *fixing* an attitude, which would correspond to $\theta = 0°$, is not feasible with convex guidance.

7.4.2.4 *Method for finding the optimal pointing direction-vector*

In the previous section it was shown that the boresight vector and the nominal pointing vector obviously do not coincide (Figure 7.15). Given the position vector, it is therefore necessary to compute the thrust-pointing direction at each point in time. A method for this is presented in this section.

The three-dimensional geometry for this problem is drawn in Figure 7.16. All vectors are defined in the landing-site frame $\mathcal{L}$. The procedure for computing $\hat{\boldsymbol{n}}$ is as follows:

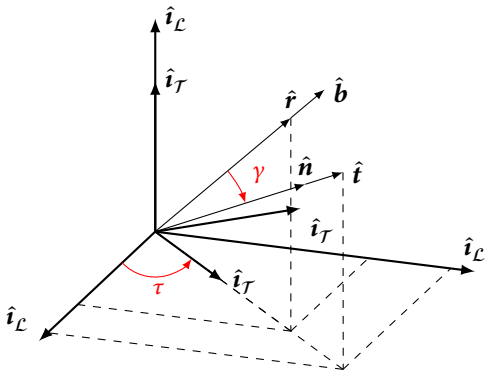


FIGURE 7.16 Three-dimensional relation between the nominal boresight-vector and the corresponding thrust vector that is to-be-commanded.

1. Define an intermediate frame $\mathcal{T}$ in which the x-y-plane is coplanar with the position vector $\boldsymbol{r}$. The vector $\hat{\boldsymbol{n}}$ will be defined in this plane as well, to avoid off-centering the FOV through a crossrange component.
2. Transform the position vector into the intermediate frame.
3. Rotate the unit direction-vector by the sensor angle γ to obtain the pointing direction in the intermediate frame.
4. Transform this vector back into the landing-site frame and feed it to guidance.

These steps are derived mathematically now. First, the direction vector is expressed in the intermediate frame by transforming it using an elementary rotation matrix (see Section 3.3):

$$\hat{\boldsymbol{r}}^{\mathcal{T}} = \mathsf{R}_z(\tau)\hat{\boldsymbol{r}}^{\mathcal{L}} \tag{7.23}$$

where the crossrange angle τ is defined as:

$$\tau = \arcsin \frac{r_x}{\sqrt{r_x^2 + r_y^2}} \tag{7.24}$$

Second, this vector is *rotated (not transformed)* downwards by the angle γ. A vector rotation is performed by carrying out exactly *the opposite* of a transformation, and thus one may write:

$$\hat{\boldsymbol{n}}^{\mathcal{T}} = \mathsf{R}_y(-\gamma)\hat{\boldsymbol{r}}^{\mathcal{T}} \tag{7.25}$$

Third, the direction vector is projected back onto the axes of the $\mathcal{L}$ frame:

$$\hat{\boldsymbol{n}}^{\mathcal{L}} = \mathsf{R}_z(-\tau)\hat{\boldsymbol{r}}^{\mathcal{T}} \tag{7.26}$$

The overall equation to express the direction vector in the landing-site frame is therefore given as:

$$\hat{\boldsymbol{n}} = \mathsf{R}_z(-\tau)\mathsf{R}_y(-\gamma)\mathsf{R}_z(\tau)\hat{\boldsymbol{r}} \tag{7.27}$$

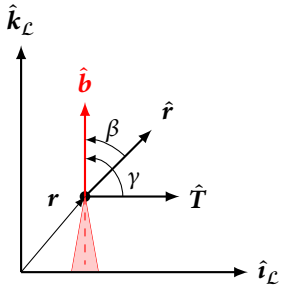


FIGURE 7.17 Viewing angle geometry given the local position and thrust vector.

This procedure is to be carried out at each time-step to apply the attitude constraint.

7.4.2.5 *Method for finding the landing-site viewing-angle*

To verify the performance of the viewing constraint it is critical to compute the viewing angle at each time step. This problem is the inverse of finding the pointing direction in the previous section: it is now assumed that the guidance command has already been computed, and the actual attitude is know. From this it must then be computed under which angle the landing site is seen.

The planar geometry of this situation is sketched in Figure 7.17. Evidently, the formula for the viewing angle is:

$$\beta = \arccos(\hat{\boldsymbol{b}} \cdot \hat{\boldsymbol{r}}) \tag{7.28}$$

Three cases of landing-site visibility can be deduced from this equation:

1. $\beta = 0°$: direct line-of-sight.
2. $\beta \leq \text{FOV}/2$: landing site visible.
3. $\beta > \text{FOV}/2$: landing site invisible.

To find $\hat{\boldsymbol{b}}$, the transformation is the same as in the previous section, only now with the thrust as reference and reversing the vector rotation:

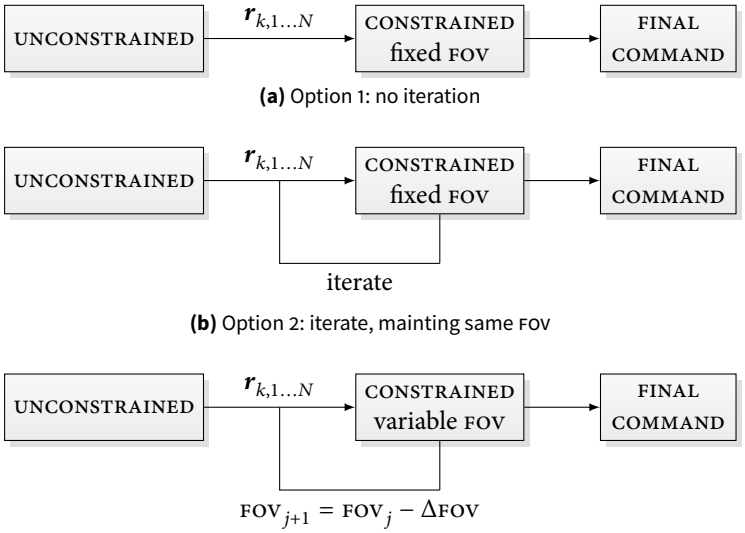
$$\hat{\boldsymbol{b}} = \mathrm{R}_z(-\tau)\mathrm{R}_y(\gamma)\mathrm{R}_z(\tau)\hat{\boldsymbol{T}} \tag{7.29}$$

where τ is given in this case as:

$$\tau = \arctan\frac{T_y}{T_x} \tag{7.30}$$

7.4.2.6 *An iterative approach to applying the constraint*

At this point all geometries have been defined. However, one may note that the reasoning until this point has indeed been circular! To define the constraint for the thrust (the control variable), the position vector at each point in time needs to be known. However, in the optimization problem, the position vector is actually defined *by the thrust*



(a) Option 1: no iteration

(b) Option 2: iterate, mainting same FOV

(c) Option 3: iterate, decreasing FOV step-wise

FIGURE 7.18 Architectural options for enforcing the viewing-angle constraint.

itself. There is thus a "chicken and egg problem" regarding the direction vector and the thrust vector.

This problem becomes apparent when setting the sensor angle to $\gamma = 0°$ for the sake of argument. This would correspond to a direct alignment of the thrust vector with the position vector. One may then set $\hat{\boldsymbol{n}} = \hat{\boldsymbol{r}}$ in the constraint (Equation [7.20]), which yields:

$$\boldsymbol{r}^\mathsf{T}\boldsymbol{T} \geq \|\boldsymbol{r}\|\Gamma\cos\theta \tag{7.31}$$

It is stressed that this shortcut *is not* possible, because this is *not* a second-order cone-constraint as defined in Section 5.5 anymore. This is because the variables and the solution $\boldsymbol{r}$ are mixed in this equation.

It is proposed to circumvent this problem *through iteration*. The idea is to first solve the unconstrained problem, and feed forward the time-history of the position vector to the constrained optimization problem. The time-of-flight is kept constant during the iterations. Three different options for this have been considered, as shown in Figure 7.18.

In Option 1, see (a), the final command would be determined in one step. Test runs have shown that this option is not viable. If, for example, the pointing cone is chosen as small as $\theta = 5°$ to enforce good visibility throughout the flight (which comes close to centering the FOV on the landing site), then the difference between the unconstrained state history and the constrained output would be too large, resulting in an infeasible problem.

In the second option, the pointing cone would now still be constrained by a constant cone-angle θ, but now it should be converged to the final solution through several iterations. This option is flawed in the sense that if the first iteration fails, all other iterations will be doomed as well. However, it was found that even if the first iteration succeeds, all

subsequent runs will deteriorate in performance (!) and ultimately become infeasible. Therefore, Option 2 is not suitable either.

The approach in Option 3 is more gradual. First, the unconstrained problem is solved. However, in the next step, the cone angle is taken to be very lax at first. It is then gradually decreased per iteration, so that the solution can *converge* to the final trajectory. This option indeed works in practice, as demonstrated in the sequel. The following heuristic has been found to work very well through simple experimentation, with $M = 5$:

$$\theta(0) = M \cdot 6° + \theta_{\text{requirement}} \tag{7.32}$$

$$\theta(j+1) = \theta(j) - 5°, \quad j = 0, \dots, M \tag{7.33}$$

One limitation that is immediately apparent from this is that the optimization problem has to be solved repeatedly. This compromises some of the speed gained by formulating it as an SOCP. It is doubtful whether applying such a computationally intensive constraint would still allow for optimizing the time-of-flight t_{go} concurrently. However, this is not necessarily an issue, as it is reported that the guidance algorithm developed by ASTRIUM for the ESA LL also fixes the t_{go} during the approach phase [Diedrich 2011].

7.4.3 Algorithm consolidation

Having laid out the geometry and the general idea for applying the constraint through iteration, it can now be formulated as a mathematical optimization problem. To derive the new attitude constraint, consider again the previously discretized general constraint Equation [7.16]:

$$\hat{\boldsymbol{n}}^{\mathsf{T}} E_{\boldsymbol{\tau}} E_{k,\boldsymbol{U}} \boldsymbol{U} \geq \boldsymbol{e}_\sigma{}^{\mathsf{T}} E_{k,\boldsymbol{U}} \boldsymbol{U} \cos(\theta), \quad k = 1, 2, \dots, N$$

The state history of the previous run will be given by $\tilde{\boldsymbol{X}}^-$, and is not a problem variable. One can thus write for the state history:

$$\hat{\boldsymbol{r}}(k) = \frac{E_{k,\tilde{\boldsymbol{X}}} \tilde{\boldsymbol{X}}^-}{\| E_{k,\tilde{\boldsymbol{X}}} \tilde{\boldsymbol{X}}^- \|}, \quad k = 1, 2, \dots, N \tag{7.34}$$

Combining this with the formula relating the position vector to the nominal pointing direction, Equation [7.27], one can thus write for the constraint:

$$\left[\mathsf{R}_z(-\tau_k) \mathsf{R}_y(-\gamma) \mathsf{R}_z(\tau_k) \left(\frac{E_{k,\tilde{\boldsymbol{X}}} \tilde{\boldsymbol{X}}^-}{\| E_{k,\tilde{\boldsymbol{X}}} \tilde{\boldsymbol{X}}^- \|} \right) \right]^{\mathsf{T}} E_{\boldsymbol{\tau}} E_{k,\boldsymbol{U}} \boldsymbol{U} \geq \boldsymbol{e}_\sigma{}^{\mathsf{T}} E_{k,\boldsymbol{U}} \boldsymbol{U} \cos(\theta) \tag{7.35}$$

which is still a valid SOCP constraint, and as before $k = 1, 2, \dots, N$. The crossrange angle is computed using Equation [7.24] at each time step.

The two optimization problems to be solved successively are now formulated. They are applied with the glide slope constraint (see Equation [7.11]).

▶ PROBLEM 8 Convex approach-phase guidance-problem with glide-slope constraint and attitude constraint. Based on Problem 7.

$$
\begin{aligned}
&\text{minimize } \boldsymbol{e}_{N\sigma\Delta t}^{\mathsf{T}}\boldsymbol{U}\\
&\text{subject to}\\
&\|E_{\boldsymbol{\tau}}E_{k,\boldsymbol{U}}\boldsymbol{U}\| \le \boldsymbol{e}_{\sigma}^{\mathsf{T}}E_{k,\boldsymbol{U}}\boldsymbol{U}, \quad k = 1, 2 \dots, N\\
&\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right) + \tfrac{1}{2}\left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right)^2\right] \le E_{\sigma}\boldsymbol{U} \le \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{2N}\right)\right]\\
&\boldsymbol{z}_{2N} \le E_z\tilde{\boldsymbol{X}} \le \boldsymbol{z}_{1N}\\
&\boldsymbol{e}_{r_z,\tilde{\theta}}^{\mathsf{T}}E_{k,\tilde{\boldsymbol{X}}}\tilde{\boldsymbol{X}} \le \left\|E_{x,y}E_{k,\tilde{\boldsymbol{X}}}\tilde{\boldsymbol{X}}\right\|, \quad k = 1, 2, \dots, N\\
&\boldsymbol{e}_{r_z,\tilde{\theta}}^{\mathsf{T}} = \begin{pmatrix}0 & 0 & \tan\tilde{\theta} & 0 & 0 & 0 & 0\end{pmatrix}\\
&\left[\mathsf{R}_z(-\tau_k)\mathsf{R}_y(-\gamma)\mathsf{R}_z(\tau_k)\left(\frac{E_{k,\tilde{\boldsymbol{X}}}\tilde{\boldsymbol{X}}^{-}}{\|E_{k,\tilde{\boldsymbol{X}}}\tilde{\boldsymbol{X}}^{-}\|}\right)\right]^{\mathsf{T}}E_{\boldsymbol{\tau}}E_{k,\boldsymbol{U}}\boldsymbol{U} \ge \boldsymbol{e}_{\sigma}{}^{\mathsf{T}}E_{k,\boldsymbol{U}}\boldsymbol{U}\cos(\theta)\\
&\qquad k = 1, 2, \dots N\\
&E_{\boldsymbol{x}(N)}\boldsymbol{X} = \begin{pmatrix}\boldsymbol{r}_f^{\mathsf{T}} & \dot{\boldsymbol{r}}_f^{\mathsf{T}}\end{pmatrix}^{\mathsf{T}}
\end{aligned}
$$

This algorithm is now summarized in Figure 7.19. First the unconstrained problem is solved. If feasible, the constrained problem is repeatedly solved, where the pointing cone is continuously narrowed down as j increases (Equation [7.32]). Moreover, the state history is fed back into this problem to approximate spherical gravity. This is necessary even in the approach phase, as otherwise errors in the order of meters will occur. Convergence of the gravity-field model is rapid and will be as fast as the convergence of the viewing angle. Once the iterations are completed and the problem was still feasible, a final verification step is done. In this step, the guidance output is post-checked after the run has been completed to see whether the constraint is truly met. This is necessary because the state-history fed to the constraint from the previous iteration is not necessarily exactly identical to the state history of the current run. It could therefore happen that the direction vectors at the temporal nodes are slightly off, and while the constraint is enforced, in reality it is not met. This constitutes the need for the additional check. If this should have been passed, guidance outputs the final command. The algorithm is now put into action in the next section, presenting some results.

7.4.4 Results

To demonstrate the feasibility of the constraint, a realistic test case with challenging conditions has been selected. This has been chosen from the most difficult trajectories that have been identified among the results in Chapter 9. The scenario is based on the ESA LL baseline, which defines the following *phases* for applying the viewing-angle constraint (see Chapter 9, Section 4.5.1 for details) [Kerr et al. 2013]:

$r_z > 150\,\text{m}$: $\theta = 5°$ From the beginning of the approach phase up to an altitude of 150 m the landing site shall remain close to the boresight. The pointing cone is therefore kept small.

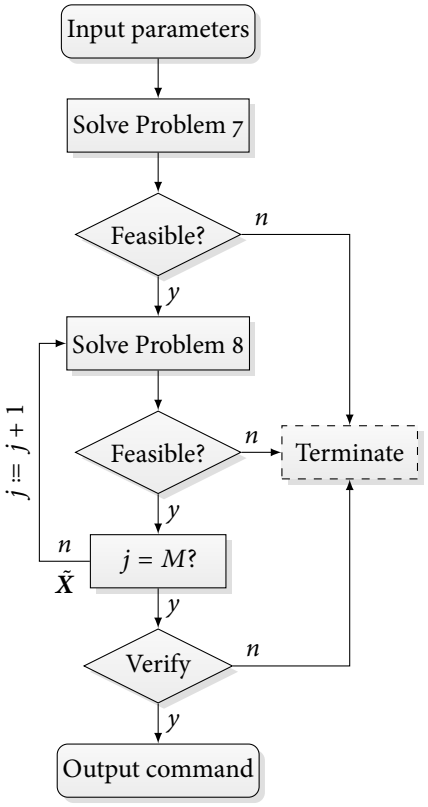


FIGURE 7.19 Algorithmic procedure for applying the viewing angle constraint.

TABLE 7.1 Input to the viewing-angle constraint test.

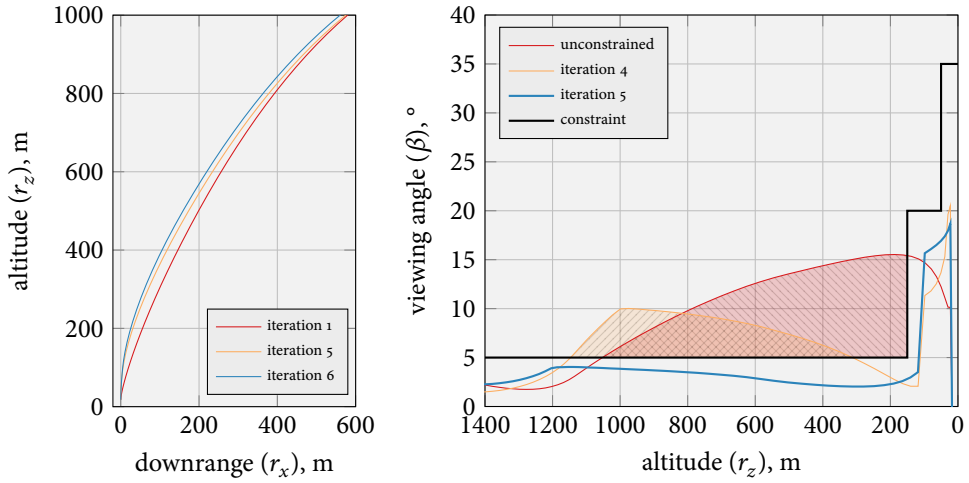
r_{x_0}, m	r_{z_0}, m	v_{x_0}, m	v_{z_0}, m	r_{z_f}, m	v_{z_f}, m	m_0, kg	T_h, N	T_l, N	t_{go}, s	γ, °
1678	1438	−82	−20	15	−1.5	987	3629	437	59	20

150 m > r_z > 50 m: $\theta = 20°$ In this leg the landing site does not need to be in the center of the FOV anymore, but the LIDAR needs to maintain a visual of the landing site somewhere in the FOV. As the LIDAR's FOV is 40°, the maximum tolerable viewing angle is therefore 20°.

50 m > r_z > 15 m: $\theta = 35°$ In the final approach-phase segment, the LIDAR may lose a visual, but the camera (FOV = 70°) must maintain it within its FOV.

This scenario has been put into the algorithm shown in Figure 7.19 in combination with the parameters in Table 7.1. Note the "Terminate" block, which flags the problem as infeasible if no solution can be found. The results are plotted in Figure 7.20. These plots show the evolution of the trajectory throughout the iterations, where only three selected graphs are shown to keep the figure readable. Figure 7.20(a) shows that at the end of the flight, the trajectory is noticeably changed.

More striking results can be read off from Figure 7.20(b). It can be seen that the unconstrained solution exhibits a smooth viewing-angle profile, that violates the first constraint phase. After four iterations, the magnitude of the error has been signficantly reduced already. However, in the first phase there is still some overshoot. This is to be expected, because the tolerable viewing angle has not converged to the allowed 5° yet,



(a) Trajectory evolution

(b) Viewing-angle constraint

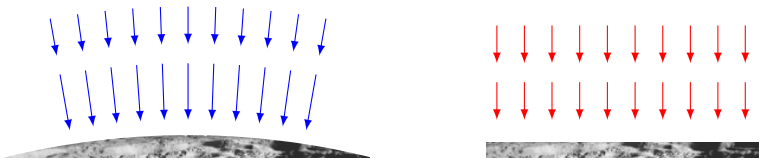
FIGURE 7.20 Demonstration of the viewing-angle constraint. The unconstrained solution grossly violates the constraints, while this can be avoided after five iterations.

but rather is at 10° at this stage. The latter can be seen in the plot, as there is a peak at 10° after which the angle decreases again. Finally, the fifth and final iteration fully meets the constraint in all legs. Note that even though the trajectory changes slightly, the fuel penalty is actually minimal, rising only by 0.4 % to 48.33 kg from 48.14 kg.

One interesting observation is that at the point where a larger viewing angle is permissible ($r_z < 150$ m), the result shoots up. This is because the algorithm uses the final flight phase to alleviate the constrained control-solution from the first part of the flight. Finally, the plot shows that the viewing angle does not shoot up at 150 m exactly, but rather a bit later. This is because of the discretization: there exists no temporal node at exactly 150 m, and the constraint is thus relaxed only at the first node after this.

7.5 SPHERICAL GRAVITY

Up until now the gravity model used in the guidance algorithm has been the simplest possible—a flat-Moon, constant-magnitude model. This choice was made to be in line with the literature, in which the same model was applied [Açıkmeşe and Ploen 2007]. The use of this simple LTI model allowed to focus on the overall guidance-algorithm development and made its verification easy, because it could be compared to reference data. However, the flat-Moon assumption fails if convex guidance is applied in practical uses. Especially in the main braking phase, leading from PDI up to AG, a significant part of the curvature of the Moon is covered. Even for the approach phase, starting at only about two kilometers downrange, the resulting errors will already be in the order of meters. Such applications should resort to spherical-gravity models, which are applied to convex guidance in this section. The method discussed in this section has first been proposed by Liu and Lu [2013] for rendez-vouz operations. This section presents the first application to landing guidance.



(a) spherical gravity, spherical Moon

(b) constant gravity, flat Moon

FIGURE 7.21 Differences between gravity-field models.

TABLE 7.2 Conditions at AG and PDI. The faster speed, larger altitude range during the maneuver, and longer distance downrange render a constant gravity model unsuitable.

(a) comparison (ESA LL baseline)

CONDITIONS	AG	PDI
speed, m/s	100	1700
down range, km	1.6	500
% curvature	0.015	4.58

(b) errors

	r, FLAT	r, SPHERICAL	ERROR
downrange (x), m	1265.0	1266.2	1.2
altitude (z), m	1625.0	1624.5	−0.5

7.5.1 Need

To understand the need of spherical gravity models for the main braking phase, an introduction to the errors made by constant gravity models is required. On a conceptual level, the error is illustrated in Figure 7.21. Already for the limited circle segment in the drawing, the curvature of the spherical gravity-field is noticeable. On top of the curvature, there will also be a discrepancy in the magnitude as the altitude changes.

To quantify this error, first consider the different initial conditions at PDI and AG, shown in Table 7.2(a) (see also Section 4.5.1). At AG, the spacecraft only covers a curvature of 0.015 % of the Moon, which has been computed as the fraction of the length of the equator that is covered. This gives an indication for how much the spherical gravity-field changes in direction throughout the flight, which is minimal in this case. At 117 m/s, the speed is also limited. Thus little distance is covered overall, while the direction of the spherical gravity vector would also only change slowly. At PDI the situation is different. Starting at about 500 km downrange, almost 5 % of the Moon's curvature needs to be covered. At the same time, the direction of the gravity field changes quickly due to the high speed. The resulting error is thus expected to be significant.

Nevertheless, the errors introduced by assuming a flat Moon can also be significant even for the approach phase. This is documented in Table 7.2, which shows the error made when computing the Cartesian components of the AG when ignoring the curvature of the Moon. The second column shows the downrange and altitude components for the flat Moon. In the third column these have been converted to be measured along the curvature of the Moon. At > 1 m, the downrange error is significant, remembering that the landing precision shall be in the order of meters.

To back these claims with numbers, the following test case is devised. A 200 km LLO is simulated in the Lunar-inertial $\mathcal{I}$-frame, once with a spherical gravity-field model and once with a constant gravity-field model. The simulation is run with a simple RK4

TABLE 7.3 Input to the test cases for demonstrating the difference between flat and spherical gravity-fields. See Figure 7.22. Unspecified state-vector elements are zero.

$\mu_{\leftmoon}$, m^3/s^2	$x_{I,0}$, m	$\dot{y}_{I,0}$, m/s	I_{sp}, s	g_0, m/s^2	T/W	Δt, s
$4.9028 \cdot 10^{12}$	$1937.1 \cdot 10^3$	1591	300	9.806 65	1.75	0.1

fixed step-size integrator until the spherical gravity-field model reaches 500 km in x position (Section 8.3).[1] To have a fair comparison of both models, the simulation is stopped for the flat-Moon model after the same time. A second test case is run according to the same conditions, but now with a thrust vector pointing opposite to the velocity vector at a constant thrust-to-weight ratio of $T/W = 1.75$ (essentially a gravity turn). This is done to show the influence of powered flight on the error. The simulation parameters are documented in Table 7.3.

The results of this investigation are plotted in Figure 7.22. The first plot shows the curved trajectory over 500 km for the cases with and without thrust. The curvature is clearly visible. However, on this scale the difference between spherical and constant gravity models is hardly discernible. For this reason, the evolution of the errors is plotted in Figure 7.22(b) for the position and (c) for the velocity. They have been computed as follows: [2]

$$\varepsilon_{\boldsymbol{r}} = \left| \|\boldsymbol{r}_{\text{cst}}\| - \|\boldsymbol{r}_{\text{sph}}\| \right|, \quad \varepsilon_{\dot{\boldsymbol{r}}} = \left| \|\dot{\boldsymbol{r}}_{\text{cst}}\| - \|\dot{\boldsymbol{r}}_{\text{sph}}\| \right| \tag{7.36}$$

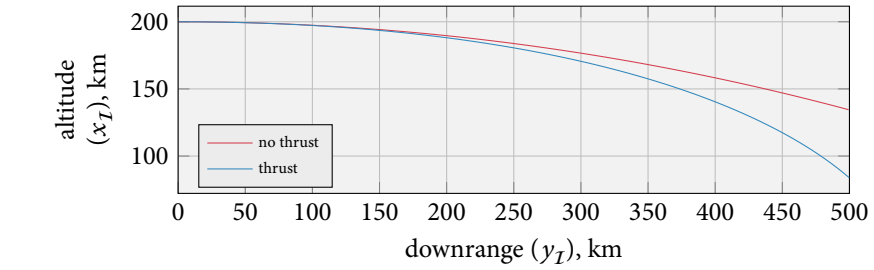
Most importantly, the plots show that the position error is in the order of 3000 m for the thrusting case, and still 1000 m without. As the altitude at AG is merely about 1600 m, such an error is unacceptable. In the worst case, the vehicle would end up crashing into the surface due to guidance inaccuracies. Several other important observations are:

- The error grows *faster* if thrust is applied. This is because the gravity-turn-like maneuver rotates the trajectory gradually into a vertical orientation, where the difference in the models starts to play a bigger role. This also explains that the error is almost the same for the thrusting and non-thrusting case for the first 200 s and then starts to diverge.
- The maximum error is larger if thrust is applied for the reason above, and because of the increased time-of-flight.
- The velocity errors for both thrust cases are closer to each other than the position errors. This can be explained by the fact that the position is the time integral of the velocity and thus should grow faster.

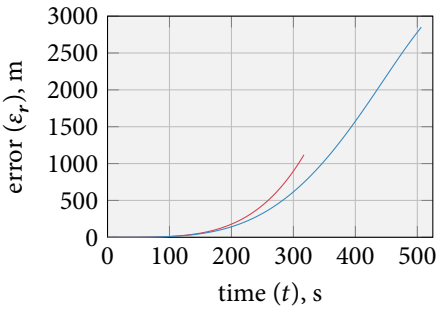
In conclusion, a spherical gravity-field model surely is required for main-braking-phase guidance-algorithms. Such a model will be applied to convex guidance in the following section.

1 Measured linearly along the coordinate axis here, not curved.

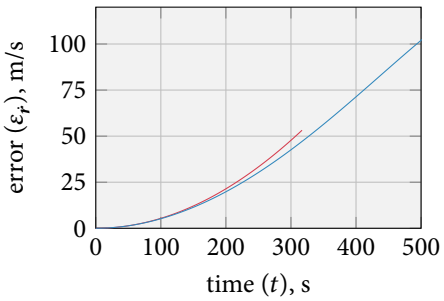
2 For the case without thrust the error could be computed analytically by solving the Kepler equations and the simple EOMs of the constant gravity case. The case with thrust is more difficult to treat. For consistency and simplicity, only numerical solutions are discussed here.



(a) Trajectory in spherical gravity-field



(b) Position error



(c) Velocity error

FIGURE 7.22 Error made by assuming a constant, vertical gravity vector with respect to a spherical gravity model.

7.5.2 Implementation

Following the need for an improved gravitational model, this section discusses its actual implementation. This will be achieved by revising the process equation as an linear time-varying system (LTV). It is then solved and discretized.

7.5.2.1 *Revised process equation*

The derivation of the spherical-gravity model for convex guidance starts with the definition of a new linear system-model for the dynamics. In the vertical gravity case, the LTI was given by (see Equation [6.39], Page 71):

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} \dot{\boldsymbol{r}}(t) \\ \ddot{\boldsymbol{r}}(t) \\ \dot{z}(t) \end{pmatrix} = \underbrace{\begin{bmatrix} 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A_c} \underbrace{\begin{pmatrix} \boldsymbol{r}(t) \\ \dot{\boldsymbol{r}}(t) \\ z(t) \end{pmatrix}}_{\boldsymbol{x}(t)} + \underbrace{\begin{bmatrix} 0 & 0 \\ \mathrm{I} & 0 \\ 0 & -\alpha \end{bmatrix}}_{B_c} \underbrace{\begin{pmatrix} \boldsymbol{\tau}(t) \\ \sigma(t) \end{pmatrix}}_{\boldsymbol{u}(t)} + \underbrace{\begin{bmatrix} 0 & 0 \\ \mathrm{I} & 0 \\ 0 & 0 \end{bmatrix}}_{B_g} \underbrace{\begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix}}_{\boldsymbol{g}_c} \tag{7.37}$$

where the dimensionality indices have been omitted for convenience. In this case, the gravity was modelled as a constant input to the system. This was done because the constant model is not state-dependent, but merely reads:

$$\boldsymbol{g}(t) = \boldsymbol{g} = -\frac{\mu}{R_{\mathfrak{C}}^2}\hat{\boldsymbol{k}}_{\mathcal{L}} \tag{7.38}$$

where $R_{\mathbb{C}}$ is the mean radius of the central body, in this case the Moon. However, the spherical Newtonian gravity-field model *is* state dependent, and can be written as (see Section 3.5.1):

$$\boldsymbol{g}(t) = -\frac{\mu}{\|\boldsymbol{r}(t)\|^3}\boldsymbol{r}(t) = -\frac{\mu}{r(t)^3}\boldsymbol{r}(t) \tag{7.39}$$

To include this in the dynamical model, the system matrix needs to be reformulated. The process equation then becomes:

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} \dot{\boldsymbol{r}}(t) \\ \ddot{\boldsymbol{r}}(t) \\ \dot{z}(t) \end{pmatrix} = \underbrace{\begin{bmatrix} 0 & \mathsf{I} & 0 \\ -\mu/r(t)^3\mathsf{I} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A_c(r(t))} \underbrace{\begin{pmatrix} \boldsymbol{r}(t) \\ \dot{\boldsymbol{r}}(t) \\ z(t) \end{pmatrix}}_{\boldsymbol{x}(t)} + \underbrace{\begin{bmatrix} 0 & 0 \\ \mathsf{I} & 0 \\ 0 & -\alpha \end{bmatrix}}_{B_c} \underbrace{\begin{pmatrix} \boldsymbol{\tau}(t) \\ \sigma(t) \end{pmatrix}}_{\boldsymbol{u}(t)} \tag{7.40}$$

This equation reveals why a spherical gravity model was not implemented sooner: the state-dependency on $r(t)^3$ makes the system matrix nonlinear and time-dependent. It is therefore no longer directly applicable to convex guidance, which requires a *linear*, (but not necessarily time-invariant) system.

It is important to realize that the frame of reference of the state propagation is now the inertial frame $\mathcal{I}$, as compared to the landing-site frame $\mathcal{L}$ that was used in the LTI formulation. The radius vector in Equation [7.40] is thus measured from the Moon's COM up to the spacecraft's COM. This makes the application of some constraints, such as the glide-slope constraint, impossible in this formulation, because they rely on the state vector in the landing-site frame. A way to work around this is proposed in Section 7.5.6.

The classical approach to resolve the nonlinearity in the new process equation would be to linearize $A_c(t)$ about a nominal trajectory. A smarter and more accurate way that can retain nonlinear dynamics has been proposed by Liu and Lu [2013] and will also be applied here. This method successively approximates the spherical solution with a linear time-varying system (LTV). The general idea is to solve the dynamic system by iteration, and with every step obtain a time-history for the radius vector $\boldsymbol{r}(t)$. At each successive step, the time-history of the previously found $\boldsymbol{r}(t)$ is substituted into the system matrix A_c, thereby making it a linear time-varying matrix. This is a similar concept as for the viewing-angle constraint the previous section, where the state-history of the previous step was also reused. The problem is to be solved for $t \in (t_0, t_f)$. This time interval is not changed during the entire process. In this way, the full nonlinear dynamics are modelled. Liu and Lu [2013] prove that this approach leads to convergence, as has also already been observed for the attitude constraint.

In mathematical terms, the algorithm can be described as follows. First consider the homogeneous dynamical model by itself, without any optimization problem:

1. Set $j = 1$ for the first iteration.

2. Define the dynamical-system model based on the relevant time-history of the radius vector:

$$\begin{aligned} r^{(j-1)}(t) &= \begin{cases} r^{(0)} = r_0 = \text{const.} & \text{if } j = 1 \\ r^{(j-1)}(t) & \text{for all other } j > 1 \end{cases} \\ \dot{\boldsymbol{x}}^{(j)}(t) &= A_c\left(r^{(j-1)}(t)\right)\boldsymbol{x}^{(j)}(t) + B_c\boldsymbol{u}^{(j)}(t), \quad \boldsymbol{x}^{(k)}(0) = \boldsymbol{x}_0 \end{aligned} \tag{7.41}$$

where $r^{(j-1)}(t) = \left\|\boldsymbol{r}^{(j-1)}(t)\right\|$ is already known from the previous iteration.
NB: the initial conditions remain the same for each iteration.

3. Solve the linear LTV to obtain the state history:

$$\boldsymbol{x}^{(j)}(t) = \begin{bmatrix}\boldsymbol{r}^{\mathsf{T}(j)}(t) & \dot{\boldsymbol{r}}^{\mathsf{T}(j)}(t) & z^{(j)}(t)\end{bmatrix}^{\mathsf{T}} \qquad [7.42]$$

The solution method is explained below, and is very similar to the solution of LTIS described in Section 6.3.2.

4. Check for convergence by comparing the current to the previous solution:

$$\sup_{t_0 \le t \le t_f} \left\|\boldsymbol{x}^{(j)}(t) - \boldsymbol{x}^{(j-1)}(t)\right\| \le \varepsilon, \quad j > 1 \qquad [7.43]$$

where ε is a prescribed tolerance. To save computational effort, one could also base this evaluation on the total error, although this could lead to outliers or too many iterations:

$$\left\|\boldsymbol{x}^{(j)}(t) - \boldsymbol{x}^{(j-1)}(t)\right\| \le \varepsilon, \quad j > 1 \qquad [7.44]$$

Based on either outcome, determine the next step:

a) If the inequality holds true terminate the loop. The final solution has been found as $\boldsymbol{x}(t) = \boldsymbol{x}^{(j)}(t)$.
b) If the condition is not satisfied, return to Step 1, increase the index $j := j+1$, and repeat the process.
c) If a specified maximum number of iterations has been reached, terminate and raise a warning flag.

This basic algorithm can be extended by a control input in a straight-forward manner. However, this then requires the solution of an optimization problem at each step. Before this is discussed, the following section first treats the solution of the LTV, which is slightly different than the previously solved LTI (see Section 6.3.5).

7.5.2.2 *Linear time-variant system solution and discretization*

The solution to an LTV was already given for the continuous-time case in Equation [6.41] (Page 71):

$$\boldsymbol{x}(t, t_0, \boldsymbol{x}_0) = \Phi(t, t_0)\boldsymbol{x}_0 + \int_{t_0}^{t} \Phi(t, s) B_c(s) \boldsymbol{u}(s)\, \mathrm{d}s$$

Contrary to the LTI case, this equation does *not* have a simplified solution and needs to be solved as is. Fortunately, this solution becomes easier if the discrete-time case is considered. Because only this case is relevant for solving the guidance problem at this point, it will immediately be detailed next.

Linear time-varying discrete-time systems were introduced in Equation [6.50] as:

$$\boldsymbol{x}(k+1) = A_d(k)\boldsymbol{x}(k) + B_d(k)\boldsymbol{u}(k) \qquad [7.45]$$

The system matrix will be written in a short-hand from now on, omitting the explicit dependence on $r^{(j-1)}(k)$:

$$A_k = A_d(k) = A_d\left(r^{(j-1)}(k)\right) \tag{7.46}$$

At this point the developments start digressing from the LTI case. Where the state-transition matrix was previously given as $\Phi_d(k, k_0) = A_d^{k-k_0}$, this simplification now does not hold anymore. This is because A_d changes at each time step, and can thus not be just raised to a power anymore. Antsaklis and Michel [2006, p. 64] derive the state-space system solution for the LTV case, and show that the state-transition matrix then reads:

$$\Phi_d(k, k_0) = \begin{cases} \prod_{i=k_0}^{k-1} A_d(i), \; i > k_0, & k \neq k_0 \\ \mathrm{I}, & k = k_0 \end{cases} \tag{7.47}$$

Consequently, the state-transition matrix of any time step depends on all time steps before it. It therefore needs to be re-evaluated with every solution step. The individual evaluations only depend on the known system matrix $A_d(i)$, though, and are mathematically easy to perform.

With the given state-transition matrix, the total solution of the LTV can be written as [Antsaklis and Michel 2006, p. 64]:

$$\boldsymbol{x}(k) = \Phi_d(k, k_0)\boldsymbol{x}_0 + \sum_{i=k_0}^{k-1} \Phi_d(k, i+1)B_d(i)\boldsymbol{u}(i) \tag{7.48}$$

For future reference and to ease the computer implementation of this equation, it can be very useful to write out the first few terms of this equation as an example.

☑ EXAMPLE 7.1 Series expansion of the solution of LTVs.

Develop the series expansion of the LTV solution given in Equation [7.48] up to $k = 3$, given $k_0 = 0$.

$$\boldsymbol{x}(3) = \Phi_d(3,0)\boldsymbol{x}(0) + \sum_{i=0}^{2} \Phi_d(3, i+1)B_d(i)\boldsymbol{u}(i) \tag{7.49}$$

$$= \Phi_d(3,0)\boldsymbol{x}(0) + \Phi_d(3,1)B_d(0)\boldsymbol{u}(0) + \Phi_d(3,2)B_d(1)\boldsymbol{u}(1) \cdots \\ \cdots + \Phi_d(3,3)B_d(2)\boldsymbol{u}(2) \tag{7.50}$$

$$= A_d(2)A_d(1)A_d(0)\boldsymbol{x}(0) + A_d(2)A_d(1)B_d(0)\boldsymbol{u}(0) \cdots \\ \cdots + A_d(2)B_d(1)\boldsymbol{u}(1) + B_d(2)\boldsymbol{u}(2) \tag{7.51}$$

Just as for the LTI case, the system now needs to be stacked based on the summation in Equation [7.48] above, to allow for a simultaneous solution of the control variables (see Section 6.4.1). The resulting process equation is:

$$\underbrace{\begin{pmatrix} \boldsymbol{x}(0) \\ \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix}}_{\boldsymbol{X}} = \underbrace{\begin{bmatrix} I & & & & \\ A_0 & & & & \\ A_1A_0 & & & & \\ \ddots & & & & \\ A_{N-1}A_{N-2}\cdots A_0 & & & & \end{bmatrix}}_{S_x} \underbrace{\begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \vdots \\ \boldsymbol{x}_0 \end{pmatrix}}_{\boldsymbol{x}_0} \cdots$$

$$\cdots + \underbrace{\begin{bmatrix} 0 & & & \\ B_0 & & & \\ A_1B_0 & B_1 & & \\ A_2A_1B_0 & A_2B_1 & B_2 & \ddots \\ \vdots & \vdots & \vdots & \\ A_{N-1}A_{N-2}\cdots A_1B_0 & \cdots & \cdots & B_{N-1} \end{bmatrix}}_{S_u} \underbrace{\begin{pmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix}}_{\boldsymbol{U}} \qquad [7.52]$$

where the individual matrices have been abbreviated as $A_d(k) = A_k$ for the sake of readability. Note that unlike for the LTI case, the whole system dynamics including gravity are now covered by the stacked system matrix S_x. Treating gravity arguably unintuitively as input is not required anymore, which comes at the price of a significantly more complex matrix. Note that because these matrices contain lots of matrix products and are of significant size, Section 8.4 describes an efficient algorithm to generate them.

The last step in the discretization of the dynamical model is the actual derivation of the discretized matrices A_d and B_d. This can be done using the series expansion previously discussed in Section 6.3.4. The difference is now that the matrices are time-varying, and thus the discrete matrices $A_d(k)$ and $B_d(k)$ now need to be discretized at each time step. The relevant equations are then (cf. Equations [6.55] to [6.57], Page 73):

$$\Psi(\Delta t, k) = \sum_{j=0}^{\infty} \frac{\Delta t^j}{(j+1)!} A_c(k)^j \qquad [7.53]$$

$$A_d(k) = \mathsf{I} + \Delta t A_c(k) \Psi(\Delta t, k) \qquad [7.54]$$

$$B_d(k) = \Delta t \Psi(\Delta t, k) B_c(k) \qquad [7.55]$$

As the common denominator in these equations is the discretization matrix Ψ, it will be computed next. The series can be expanded as:

$$\Psi(\Delta t, k) = \mathsf{I}_{7\times 7} + \frac{\Delta t}{2!} A_c(k) + \frac{\Delta t^2}{3!} A_c(k)^2 + \frac{\Delta t^3}{4!} A_c(k)^3 + \mathcal{O}\big(A_c(k)^4\big) \qquad [7.56]$$

TABLE 7.4 Discretizaton error made by truncating the discretization matrix.

BENCHMARK	DIFFERENCES (ε_r), m		
r, $\mathcal{O}(A_c^3)$, km	$\mathcal{O}(A_c^2)$	$\mathcal{O}(A_c^1)$	$\mathcal{O}(A_c^0)$
1937.0993	$3.6787 \cdot 10^{-8}$	$3.6787 \cdot 10^{-8}$	0.6533

The system matrix is given as:

$$A_c(k) = A_c\left(r(t_k)\right)\begin{bmatrix} 0 & \mathsf{I} & 0 \\ -\mu/r(t)^3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad [7.57]$$

Substituting this into Equation [7.56] yields the summation:

$$\Psi(\Delta k) = \mathsf{I}_{7\times7} + \frac{\Delta t}{2}\begin{bmatrix} 0 & -\mu/r(t_k)^3 & 0 \\ -\mu/r(t_k)^3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{\Delta t^2}{6}\begin{bmatrix} -\mu/r(t_k)^3 & 0 & 0 \\ 0 & -\mu/r(t_k)^3 & 0 \\ 0 & 0 & 0 \end{bmatrix}\cdots$$
$$\cdots + \frac{\Delta t^3}{24}\begin{bmatrix} 0 & -\mu/r(t_k)^3 & 0 \\ \left(-\mu/r(t_k)^3\right)^2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad [7.58]$$

The discretized system and input matrices are now simply computed by substitution of Ψ into Equations [7.54] and [7.55]. Although an analytical solution of A_d and B_d would be possible as well, tests have shown that the speed gain in code would be negligible. This step is therefore deemed unnecessary and skipped, as also no further insight would be gained.

The necessary order of this series is not entirely straight-forward to establish. This is because the acceleration is actually hard to extract. In fact, Ψ is just the discretization matrix, and the system matrix still needs to be computed using Equation [7.54]. However, this also does not directly contain an acceleration term, but is multiplied by the state vector. It is therefore much easier to evaulate series convergence based on how the position vector changes with the order.

One may test this as follows. Consider a time-step of 1 s and a 200 km LLO around the Moon. Then the full summation shown in Equation [7.58] is taken as a benchmark, against which the position error computed with the series truncated at the lower orders is compared up to zeroth order (the identity matrix). The results of this are shown in Table 7.4. As can be seen, the error made by the zeroth order is considerable: 65.3 cm in one step of 1 s. However, if one adds only the first order term, the error nearly vanishes and is only in the order of 10^{-8} m. This is negligible for the integration times of maximally 500 s in this study. Because the error of the next higher order is of the same magnitude, it can be concluded that the result is already consistent just with the first term, and adding extra terms has no added benefit. For this reason, the series will be truncated after the first order in computer implementations, which saves floating-point operations and reduces rounding errors.

7.5.3 Spherical moon, constant magnitude model

Arguably one of the weak and inelegant points of basic convex guidance formulation is its constant, flat-moon gravity-field model. Even though such a model might be justified for short time-of-flights near the surface of the planet, such as during the approach phase, having to use a separate constant input is a somewhat awkward formulation.

A simple improvement for this is proposed here. Rather than taking the constant input, a *spherical moon, constant magnitude* gravity model is proposed. Although this does not cover the change in gravitational magnitude for a change in altitude, this will at least model the field's curvature.

The equations for this are readily available: one can use the discrete input-matrix that has just been derived. However, rather than iterating and finding the corresponding value for $r(k)$ at each point, the *initial value* r_0 is substituted for *all* points on the interval. This way, gravity is included in the system matrix again and is of spherical nature. The initial value is preferred over the central body's mean radius – which would usually be taken – because the acceleration will be more accurate at the start where the speed is larger, and the propagation of initial errors will thus be avoided.

To express this idea in a formula, one can write the resulting process equation as (cf. Equation [6.125], Page 94):

$$\begin{pmatrix} \boldsymbol{x}(0) \\ \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix} = \begin{bmatrix} A_0^0 & & & & \\ & A_0^1 & & & \\ & & A_0^2 & & \\ & & & \ddots & \\ & & & & A_0^N \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \vdots \\ \boldsymbol{x}_0 \end{pmatrix} \cdots$$
$$\cdots + \begin{bmatrix} 0 & & & \\ B_d & & & \\ A_0 B_d & \ddots & & \\ \vdots & \ddots & \ddots & \\ A_0^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix} \begin{pmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix} \quad [7.59]$$

where the system matrix is given by:

$$A_0 = A_d(0) = A_d(r_0) \quad [7.60]$$

A comparison of this method to the simpler approach is shown in Section 7.5.5.

Note that a major problem with this method is that it depends on the radius vector in the inertial frame, $\mathcal{I}$. This vector points from the Moon's COM to the lander. This formulation is unfortunately incompatible with all constraints derived in the previous section. This is because these constraints need to be expressed in the landing-site frame, $\mathcal{L}$. For this reason, yet another gravity-field model is described in Section 7.5.6.

7.5.4 Optimization algorithm

In this final step, the spherical-gravity model will be integrated into the guidance algorithm. This is a straight-forward procedure, that is also illustrated in Figure 7.23.

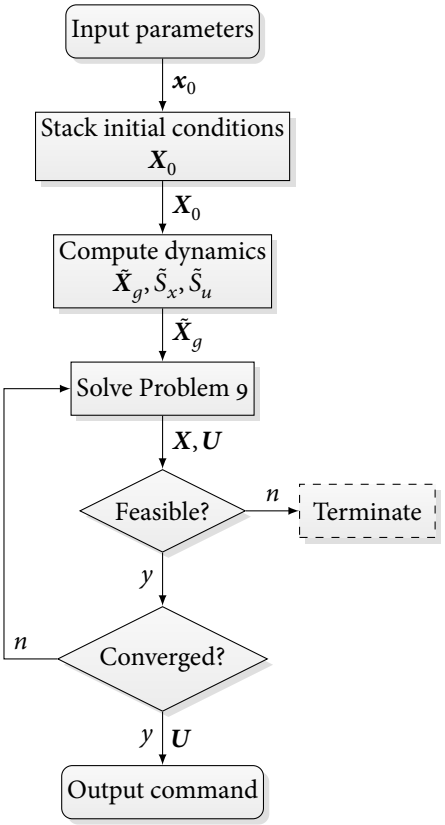


FIGURE 7.23 Algorithmic procedure for guidance with spherical gravity.

Working again in discrete-time and stacked matrices again, the first step is to stack the initial conditions:

$$\boldsymbol{X}_0 = \left[\boldsymbol{r}_0^\mathsf{T} \quad 0_{1\times 4} \quad \overset{\times(N+1)}{\cdots}\right]^\mathsf{T} \tag{7.61}$$

in which the initial position-vector is stacked $N+1$ times. This is now used to compute the reduced, stacked gravity-solution, as well as the reduced input matrix (see Page 77):

$$\tilde{\boldsymbol{X}}_g = \tilde{S}_x \boldsymbol{X}_0 + \tilde{S}_u \boldsymbol{U} \tag{7.62}$$

This equation is substituted into the optimization problem, which is then solved in the same fashion as before. The iteration procedure detailed in Section 7.5.2.1 is then applied to this problem. The specific problem applied for the main-braking phase, where spherical gravity is used, is shown below. Note that if this problem is infeasible, the algorithm is terminated (see Figure 7.23).

TABLE 7.5 Orbital parameters for the two test cases considered for evaluating propagation using the stacked system matrix.

ORBITAL PARAMETER	CIRCULAR	ELLIPTICAL
perilune (r_p), km	200	200
apolune (r_a), km	200	1860
eccentricity (e)	0	0.3
speed at perilune (v_p), km/s	1.591	1.814
speed at apolune (v_a), km/s	1.591	0.977
orbital period (T), min	127.5	217.7
time step (Δt), s	30	30

▶ PROBLEM 9 Convex main-braking guidance problem solved through iteration.

$$
\begin{aligned}
&\text{minimize } \boldsymbol{e}_{N\sigma\Delta t}^{\mathsf{T}}\boldsymbol{U}\\
&\text{subject to}\\
&\|E_{\boldsymbol{\tau}}E_{k,U}\boldsymbol{U}\| \le \boldsymbol{e}_{\sigma}^{\mathsf{T}}E_{k,U}\boldsymbol{U}, \quad k = 1, 2 \ldots, N\\
&\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}}_g - \boldsymbol{z}_{1N}\right) + \tfrac{1}{2}\left(E_z\tilde{\boldsymbol{X}}_g - \boldsymbol{z}_{1N}\right)^2\right] \le E_{\sigma}\boldsymbol{U} \le \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}}_g - \boldsymbol{z}_{2N}\right)\right]\\
&\boldsymbol{z}_{2N} \le E_z\tilde{\boldsymbol{X}}_g \le \boldsymbol{z}_{1N}\\
&E_{\boldsymbol{x}(N)}\boldsymbol{X}_g = \begin{bmatrix}\boldsymbol{r}_f^{\mathsf{T}} & \dot{\boldsymbol{r}}_f^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}
\end{aligned}
$$

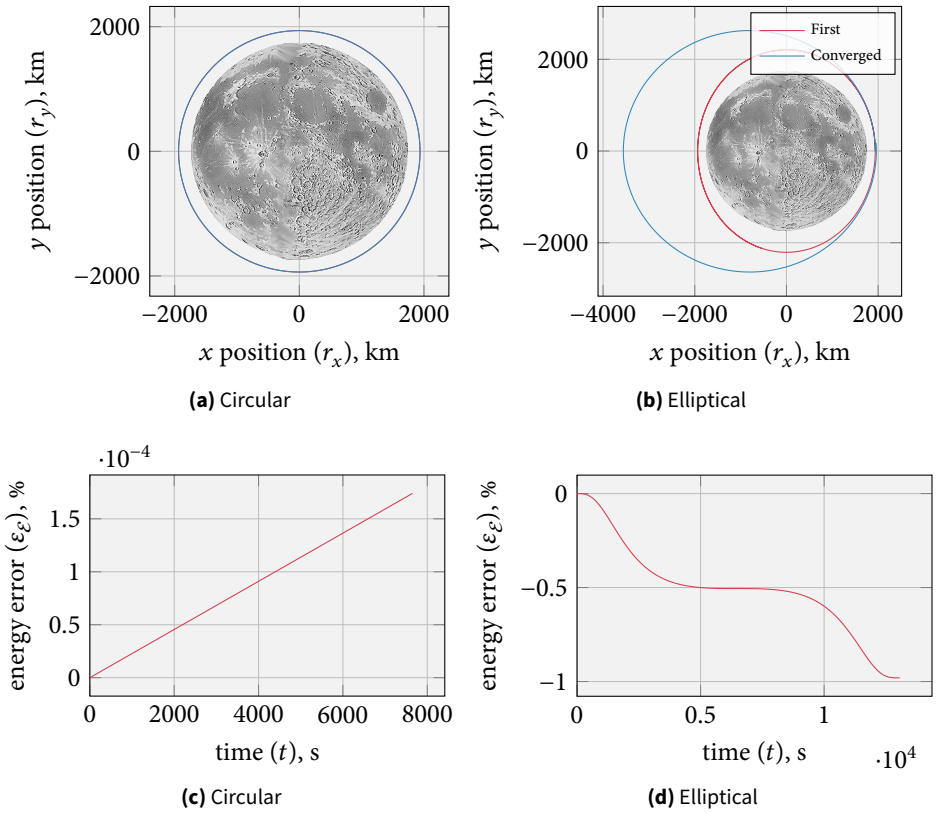
7.5.5 Results

Having introduced the algorithms for modelling spherical gravity, the method is put to test in this section. First, results for the full spherical dynamics are presented, and second the limited model of constant magnitude. To avoid duplication, guidance solutions *are not* discussed in this section, as the dynamical model is, in fact, the main result. The guidance command and resulting trajectory for a relevant flight phase, the main braking phase, are presented in Figure 7.23 on scenario design.

7.5.5.1 *Spherical gravity dynamics*

To stress the dynamical model, the first two tests shall model a circular and an elliptical orbit. It is expected that the elliptical orbit-solution exhibits minimal errors, because the magnitude of the position vector remains constant and is equal to the initial condition. For the elliptical orbit this is different, because $r(t)$ changes in magnitude throughout the orbit. The orbital parameters used for this simulation are listed in Table 7.5.

The propagated orbits are plotted in Figure 7.24. Considering the circular orbit first, the quality of the results is very promising. The algorithm converges up to numerical precision in twenty steps. On the scale of Figure 7.24(a) no errors are discernible. To



(a) Circular **(b)** Elliptical

(c) Circular **(d)** Elliptical

FIGURE 7.24 Evolution of the state propagation using the stacked system matrix for a 200 km LLO and an elliptical orbit with the same perilune and an eccentricity of 0.3. Integration step size 20 s on an orbital period of 127.5 min and 217.7 min.

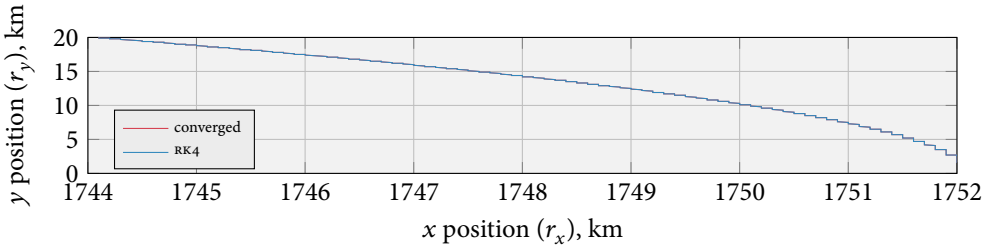
have a more accurate measure of the error, the orbital energy – which should remain constant – is evaluated. Its formula reads [Vallado 2007]:

$$\mathcal{E} = \frac{v^2}{2} - \frac{\mu}{r} \qquad [7.63]$$

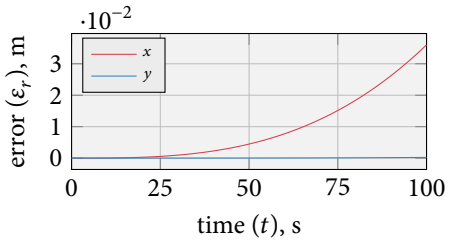
The error of the orbital energy is plotted in percent in Figure 7.24(c). This error is so small that it reaches only $1.74 \cdot 10^{-4}$ % at the end of the flight. This is an excellent result for an algorithm that is tailored to short flight phases and most certainly not for orbit propagation.

A more challenging test case is presented by the elliptical orbit shown in Figure 7.24(b) and (d). Plots are shown for the first iteration and the final iteration in (b). Clearly, the initial iteration does not deliver a correct result yet. This is because the position state-history is wrong, and a constant $r(t)$ is substituted into the dynamical system for a speed that is higher than circular velocity. However, after 23 iterations the algorithm converges to a good solution. The error in orbital energy is shown in (d). It is now much more significant than for the circular test case, reaching up to 1 % and growing in between perilune and apolune.

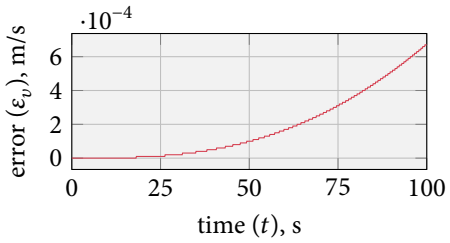
It is stressed again that the simulation of an elliptical orbit only serves the purpose of example. A relevant scenario is presented in Figure 7.25, which shows the propagation



(a) Propagated trajectory



(b) Position errors



(c) Speed error

FIGURE 7.25 Trajectory and errors for a relevant reference scenario. Initial altitude 15 km, initial speed 200 m/s. Propagated using the stacked system matrix for 100 s at a step size of 0.1 s. Errors compare the converged matrix to the RK4 solution.

TABLE 7.6 Total errors for the relevant scenario after integrating at a step-size of 0.1 s. Convergence is rapid

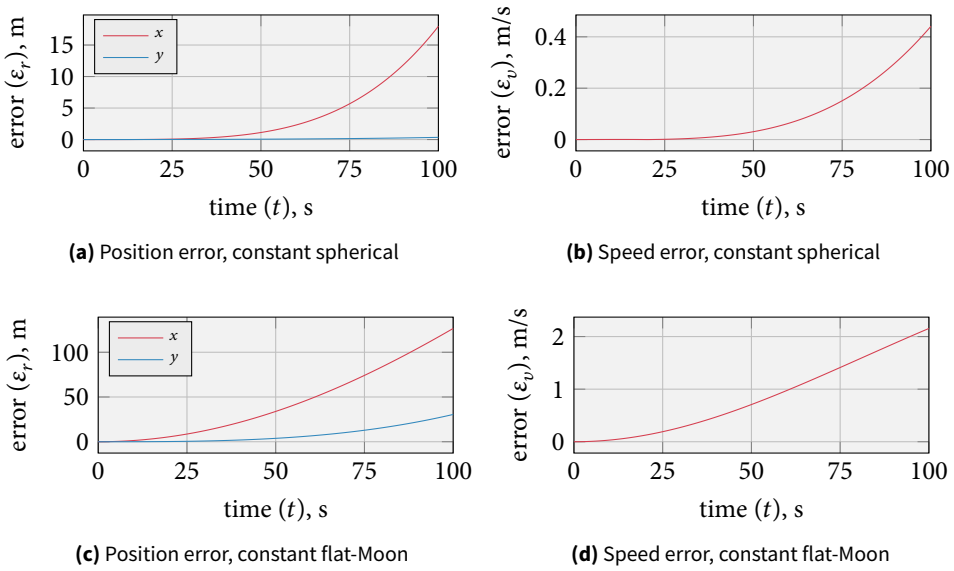
CASE	ε_{r_f}, m	ε_{v_f}, m/s
converged spherical	$3.602 \cdot 10^{-2}$	$6.744 \cdot 10^{-4}$
constant spherical	$1.797 \cdot 10^{1}$	$4.457 \cdot 10^{-1}$
constant flat moon	$1.302 \cdot 10^{2}$	2.155

of a trajectory at initially 15 km altitude and a speed of 200 m/s. It is propagated at a step size of 0.1 s for 100 s. For reference and to compute the errors, the trajectory has also been propagated with an RK4 integrator at a step size of 0.01 s, which results in negligible errors for this set-up.

The resulting errors for this scenario are minimal, as is also stated in Table 7.6. In fact, for the converged spherical model the position error grows to only about 4 cm and 0.7 mm/s for the speed. This means that in practice the errors due to the dynamical model may essentially be ignored.

7.5.5.2 *Constant spherical gravity dynamics*

The spherical Moon, constant magnitude model that has been developed in Section 7.5.3 has been applied to the relevant scenario defined in the previous section, of which the results are shown in Figure 7.26(a) and (b). For reference, the LTI dynamics have also been run for these conditions, for which the output is given in Figure 7.26(c) and (d). Applying these models to the orbital test cases makes little sense, as there is no way that they can converge to the correct solution.



(a) Position error, constant spherical

(b) Speed error, constant spherical

(c) Position error, constant flat-Moon

(d) Speed error, constant flat-Moon

FIGURE 7.26 Errors for the relevant scenario. Initial altitude 15 km, initial speed 200 m/s. Propagated using the stacked system matrix for 100 s at a step size of 0.1 s for the constant-magnitude, flat-Moon model and the constant magnitude, spherical Moon model.

The plots immediately show that the errors are unacceptably big now. For the constant spherical model, the final magnitude of the position error is 18 m and 0.5 m for the speed. This is *four* orders of magnitude more than for the converged model (Table 7.6). The situation gets even worse for the LTI model, where the error is in the hundreds of meters for the position vector, being *five* orders of magnitude worse.

7.5.5.3 *Error analysis*

To give some more substance to the analysis of the previous sections, the two spherical Moon models with constant and variable magnitude have been run at different step sizes for the reference scenario. These results are shown in Figure 7.27.

Considering the position error first, a startling observation can be made: for the spherical Moon, constant magnitude model the error (nearly) does not change with the step size. However, this is actually expected, and has already been discussed in Section 6.6.1.2, where this phenomenon was observed for the LTI model. For the LTI dynamics, it was shown that the system model perfectly matches the analytical solution for motion in the constant, flat-Moon gravity-field. The situation is similar here: the dynamics are modelled well at the temporal nodes for the spherical Moon, constant magnitude model—only now the underlying model is fundamentally wrong, because it is compared to a spherical Moon, variable magnitude model. The discrepancy at the end occurs for the runs at 20 s, 50 s, and 100 s step sizes on a 100 s simulation. The final points for the constant and converged models are identical, because they both jump directly from the initial conditions to the final state.

For the spherical Moon, variable-magnitude gravity-model the error is log-linear with respect to the time step. The number of iterations does not vary strongly. Only for the runs with extremely large step sizes – 50 s and 100 s – there are only two and one

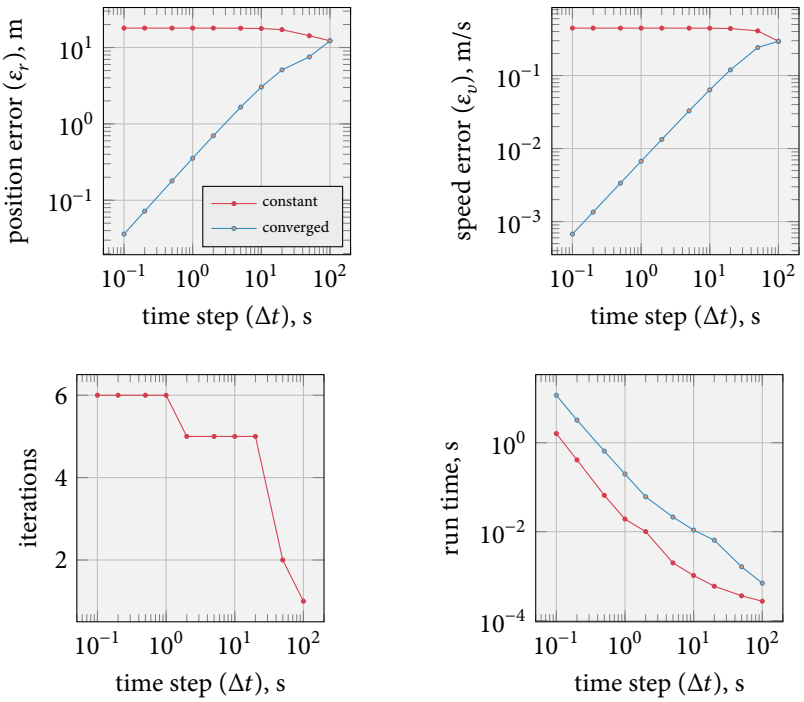


FIGURE 7.27 Performance of the propagation of the relevant scenario of 200 m/s at 15 km altitude for 100 s as a function of step size.

iterations to converge, respectively. This is because there are significantly less temporal nodes. Overall it can be said that, because there are merely six iterations, the algorithm converges quickly. Finally, it can be concluded that the run time also decreases with the step size, but is not exactly log-linear. This is because the number of iterations changes for different step sizes. In fact, the offset between both curves can be directly traced to the required iterations.

7.5.5.4 *Guidance solution*

As stated in the introduction, a relevant guidance solution is presented at length in Section 9.4.2. To avoid duplication, this is therefore not documented here.

7.5.6 Alternative implementation

Late in the process of thesis writing a major problem with regard to the gravity model has been identified. It turned out that it is *not* possible to assume constant gravity for the approach phase, because the errors caused by this assumption are in the order of meters and violate the requirement. There is thus a need for a spherical gravity-field model, even for the approach phase.

Unfortunately, the model in this section is not compatible with the approach-phase guidance law developed in the previous sections. This is because the glide-slope and the viewing-angle constraints can only be formulated in the landing-site frame $\mathcal{L}$. However, the integration of the spherical equations of motion needs to happen in the Moon-centric $\mathcal{I}$ frame. Hence, there are conflicting requirements.

Fortunately, driven by this need a new method was developed and verified in the late stages of this research. This is explained in the following. For this, first consider the LTI dynamics-model again:

$$\dot{\boldsymbol{x}}(t) = \begin{pmatrix} \dot{\boldsymbol{r}}(t) \\ \ddot{\boldsymbol{r}}(t) \\ \dot{z}(t) \end{pmatrix} = \begin{bmatrix} 0 & \mathsf{I} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{r}(t) \\ \dot{\boldsymbol{r}}(t) \\ z(t) \end{pmatrix} + \begin{bmatrix} 0 & 0 \\ \mathsf{I} & 0 \\ 0 & -\alpha \end{bmatrix} \begin{pmatrix} \boldsymbol{\tau}(t) \\ \sigma(t) \end{pmatrix} + \begin{bmatrix} 0 & 0 \\ \mathsf{I} & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix}$$

In this equation it has been assumed up to this point that the gravity vector is of constant direction and magnitude. However, this is actually by no means necessary. The gravity vector may indeed change with time and the system will still remain linear and time-invariant, as the matrices remain constant. For this reason, it is proposed to apply an iterative procedure, in which the gravity vector is recomputed at each time step and then substituted into the system equation.

This process is actually much simpler than the involved procedure with a linear time-varying system. The algorithm is:

1. Set $j = 1$ for the first iteration.
2. Determine the relevant time-history of the radius vector in the $\mathcal{L}$ frame:

$$r^{(j-1)}(t) = \begin{cases} r^{(0)} = r_0 = \text{const.} & \text{if } j = 1 \\ r^{(j-1)}(t) & \text{for all other } j > 1 \end{cases} \qquad [7.64]$$

   NB: the initial conditions remain the same for each iteration.

3. Convert the radius vector for each time step to the $\mathcal{I}$ frame. Omitting the superscript for the iteration step, the transformation can be written as (see Section 3.3.3.1):

$$\boldsymbol{r}^{\mathcal{I}} = \boldsymbol{t}_{\mathcal{I}}^{\mathcal{L}} + \mathsf{R}_{\mathcal{L}}^{\mathcal{I}} \boldsymbol{r}^{\mathcal{L}} \qquad [7.65]$$

4. Compute the local gravity vector:

$$\boldsymbol{g}^{\mathcal{I}}(t) = -\frac{\mu_{\text{☾}}}{\left(r^{\mathcal{I}}\right)^3} \boldsymbol{r}^{\mathcal{I}} \qquad [7.66]$$

5. Convert the gravity vector back into the $\mathcal{I}$ frame:

$$\boldsymbol{g}^{\mathcal{L}}(t) = \mathsf{R}_{\mathcal{I}}^{\mathcal{L}} \boldsymbol{g}^{\mathcal{I}}(t) \qquad [7.67]$$

6. Substitute this back into the system equation and obtain the solution:

$$\dot{\boldsymbol{x}}(t) = A_c \boldsymbol{x}(t) + B_c \boldsymbol{u}(t) + B_c \begin{pmatrix} \boldsymbol{g}^{\mathcal{L}}(t) \\ 0 \end{pmatrix}$$

7. Check for convergence by comparing the current to the previous solution:

$$\sup_{t_0 \le t \le t_f} \left\| \boldsymbol{x}^{(j)}(t) - \boldsymbol{x}^{(j-1)}(t) \right\| \le \varepsilon, \quad j > 1 \qquad [7.68]$$

where ε is a prescribed tolerance. Based on the outcome, determine the next step:

a) If the inequality holds true terminate the loop. The final solution has been found as $\boldsymbol{x}(t) = \boldsymbol{x}^{(j)}(t)$.

b) If the condition is not satisfied, return to Step 1, increase the index $j := j+1$, and repeat the process.

c) If a specified maximum number of iterations has been reached, terminate and raise a warning flag.

It has been determined that the with this approach the errors in the approach phase could be fully eliminated (Section 9.4.4). Initial tests indicate that the solution obtained in this way can yield identical results as the method involving the LTV. It is also expected that it will require far less floating-point operations. Therefore, it is encouraged to investigate using this model as a replacement for the LTV approach in future projects.

7.6 FURTHER ENHANCEMENT OPTIONS

This chapter set out to extend and improve the guidance algorithm based on convex guidance that has been derived in Chapter 6 and was originally proposed by [Açıkmeşe and Ploen 2007]. First, a no-subsurface-flight constraint and a glide-slope constraint from the literature have been applied. Second, a novel viewing-angle constraint has been implemented, alongside several different options for gravity-field models.

However, the possibilities of extending convex guidance do not stop at this point. Several more enhancements have been proposed in the literature and some new ideas came up during the course of this research. These are very briefly discussed in this section, as inspiration for future research.

Attitude-rate constraint

Açıkmeşe and Ploen [2005] propose a method to constrain the attitude-rate of the spacecraft. To avoid involving rotational dynamics into the problem this constraint is based on the attitude-command control-history, derived from the thrust vector. Whether this constraint is directly applicable to the SOCP used in this report needs to be investigated, as the relevant publication formulates the landing problem as a semidefinite program.

Chebychev polynomials as basis functions

It is reported in [Açıkmeşe and Ploen 2007, p. 1360] that instead of piecewise constant basis functions for the controls, one can also use Chebychev polynomials. The authors claim that:

> *Then, the description of the controls can be achieved with fewer number of coefficients $M \ll N$. This leads to significant reductions in the dimension of the resulting finite-dimensional SOCP, and in the execution times of the algorithm. In that case the preceding matrices will change but the resulting form of the optimization problem will be very close.*

This promising option should definitely be studied in the future.

Rotational speed of the central body

In practical applications the rotation of the central body needs to be taken into account, because guidance is defined in a co-rotating frame. One of the issues that can occur is inter glide-slope subsurface flight in between temporal nodes. Açıkmeşe et al. [2008] demonstrate that this can be avoided by taking into account the rotation of the central body in the system matrix A_d. It is recommended to investigate this, to see whether this is also compatible with the spherical gravity-model proposed in this thesis.

Modelling a variable specific impulse

The mass consumption during the flight is modelled by the constant α in this report:

$$\alpha = (g_0 I_{\text{sp}})^{-1} \tag{7.69}$$

This is assumed constant. As described in Section 3.5.2, though, the specific impulse varies in reality, for example, because clusters of different engines are used or because the efficiency varies with the throttle level.

One idea how this impulse could be taken into account builds on the iterative nature of solving the spherical-gravity guidance-problem. Because the optimization problem is solved multiple times, not only the state history will be available, but also the control history. This could be used to calculate the actual I_{sp} and substitute this into the equation above for each time step. An important benefit of this is that possible physical constraint violations could be avoided.

Improving the thrust-constraint linearization

In a similar manner as the previous recommendation, the thrust-constraint linearization could be improved by accurately defining a nominal trajectory. In the iterations for converging to a spherical-gravity solution, not only a position history is determined, but rather a full state history—including the mass $z(t)$. This could be substituted into the constraint equation (see Page 67):

$$\mu_1(t)\left[1 - [z(t) - z_1(t)] + \tfrac{1}{2}[z(t) - z_1(t)]^2\right] \leq \sigma \leq \mu_2(t)[1 - (z(t) - z_2(t)] \tag{7.70}$$

As a result the constraint should be more accurately tracked.

7.7 SUMMARY: A CONSOLIDATED GUIDANCE METHOD

To conclude this chapter, the developments of the previous sections shall pass review. Using convex optimization theory, a guidance algorithm has been developed that can solve the landing TPBVP efficiently and up to specified precision and global optimality. Two different subproblems have been treated: the main-braking problem (Problem 9) and the approach-phase problem (Problem 7). For each of these, different constraints and dynamical models are applied.

Common features

Both subproblems share the following properties.

- Global propellant optimality.
- System dynamics including variable mass.
- Thrust-bound constraint.
- Mass-feasibility constraint.
- Specifiable discretization step-size.

Approach phase

The approach-phase algorithm features the following:

- No sub-surface flight constraint.
- Choice of three gravity models:
  1. Flat Moon, constant magnitude;
  2. Spherical Moon, constant magnitude; and
  3. Spherical Moon, variable magnitude.
- Glide-slope constraint.
- Boulder-avoidance constraint through glide-slope.
- Attitude constraint for:
  - Avoiding upside-down flight; and
  - Constraining the landing-site viewing-angle.

Main-braking phase

In addition to the common features, the main-braking algorithm models:

- Full spherical gravity model.

Overall, a highly constrained algorithm that has the potential of guiding a Lunar lander in real-time with global optimality has been developed. Even though two subproblems need to be solved, the algorithms actually have a lot in common. This is illustrated in Figure 7.28, which shows how guidance is embedded in the GNC system. Both algorithms work with the same convex-optimization framework. This means, for

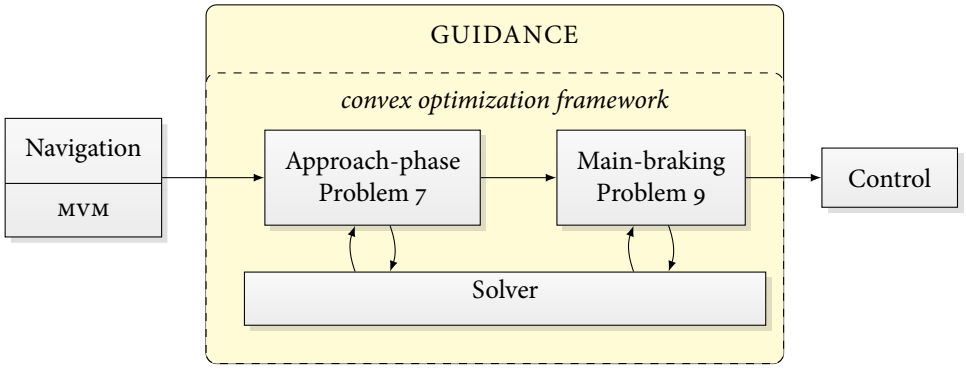
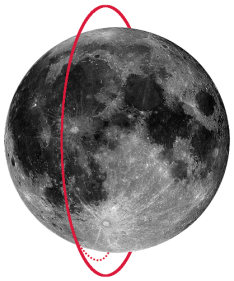


FIGURE 7.28 The developed guidance algorithms the context of the GNC architecture.

example, that they share the same solver. If the algorithm were to be implemented onboard, this would mean that the verification and validation efforts can be significantly reduced. Moreover, being rooted in the same theory means that the analysis of the algorithm is greatly eased through commonalities. Overall, this consolidated algorithm leads to a reduction of the complexity. For reference, a safe-landing scenario designed by NGC AEROSPACE LTD relied on four different algorithms, which has now been consolidated into one [Philippe 2014].

Chapter 8

CONTINUOUS-TIME SIMULATION-ENVIRONMENT



> "The purpose of computing is insight, not numbers."
>
> *Richard W. Hamming*

In science, the verification of a hypothesis calls for an experimental setup. As hardware tests of guidance algorithms are neither feasible nor necessary at this point of the development, a simulator can be considered an appropriate "experimental setup" for this research. Just as such a setup is to be carefully planned and documented for a laboratory experiment, this is also true for a software package. Thus, to ensure a streamlined development effort, this chapter outlines the basic design of the simulator.

The chapter is structured as follows. The first section outlines the top-level software architecture and the computational flow within the guidance sub-function. Tools required for solving numerical optimization problems are then discussed in Section 8.2. The propagation of the lander's state in continuous time is detailed in Section 8.3. To close the chapter, some performance considerations are the subject of Section 8.4.

8.1 SOFTWARE ARCHITECTURE

Even though the simulation environment developed for this research is comparably basic, the implementation deserves a brief introduction. The architecture is thus described in this section.

8.1.1 Top level

The software architecture is defined on the highest level in Figure 8.1, which indicates the required in- and outputs. The following will describe the purpose and functionality of the individual blocks in the flow chart.

INPUTS At the beginning of a run, the user needs to enter a number of *inputs* to set up the simulation. The basic necessities are listed in the figure. It would be excessive to list every single input here. The most important variables are:

- Initial conditions: the state of the vehicle at the first epoch.
- Vehicle parameters: mass properties, propulsive features.
- Simulation parameters: type of integrator, time step.
- Algorithm parameters: time step, desired level of convergence.

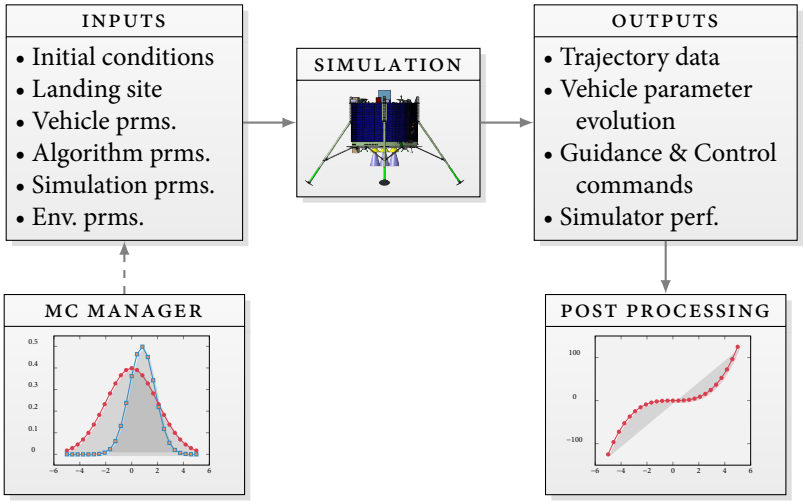


FIGURE 8.1 Top-level software architecture.

- Environment parameters: specification of gravity-field model and its parameters.

SIMULATION This block represents the simulation core that computes the desired outputs from the input. This block is discussed in more detail in the next section.

OUTPUTS The simulation will yield several outputs that are required for the analysis of the algorithms. These are user specified. Some data that is essential is:

- Trajectory data (translational position, velocity, and acceleration) as a function of time.
- Evolution of the vehicle parameters, such as the lander's mass, as a function of time.
- The guidance commands themselves as a function of time.
- Performance of the simulator, for example the estimated accuracy of the integrator, the runtime, *etc.*

The output is handled in different manners depending on the application. For post processing it is saved in text files, whereas for reproducing results it is stored in a binary format (the `.mat` files generated by MATLAB).

POST PROCESSING In a separate step, the saved output can be post-processed. This brings the abstract numerical data into a human-readable format. Post processing is done using the LaTeX package `pgfplots`.

(MC MANAGER) Optionally, the simulation can be controlled by an MC manager. This will execute the simulation a specified number of times. The MC manager also applies some user-defined stochastic variations to the input variables.

The most important block – the simulation – is detailed in the next section.

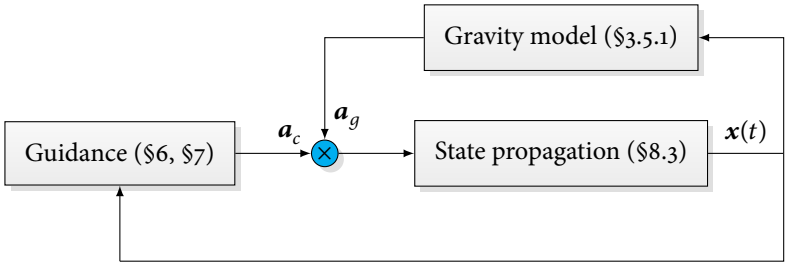


FIGURE 8.2 Basic simulation flow chart.

8.1.2 Simulation design

At the core of the software is the simulation itself. This block is further broken down in Figure 8.2. The very basic setup of the simulation reflects in the simplicity of the chart. Three main functions are needed: i) the guidance algorithm itself; ii) a gravity model; and iii) a state propagator. The guidance and gravity blocks each output an acceleration vector, that is fed (alongside the initial conditions) to the state-propagation block. This block integrates the equations of motion and computes the state at the next time step, which is then forwarded in a loop to the other blocks.

The most important design decision that needs to be made from a software point-of-view is the computational framework in which the tools are implemented. In [Gerth 2013] several such options have been identified. Ultimately, MATLAB has been chosen for the following reasons:

- In this early prototyping stage the ease of use and debugging capabilities of MATLAB are beneficial. They allow for quick implementation and modification of algorithms.
- Availability and familiarity were important factors schedule-wise, as the author did not need to delve into a new programming language first.
- The entry descent and landing (EDL) simulator the Entry and Guided Landing Environment (EAGLE) that has been developed under ESA contract is also implemented in MATLAB/SIMULINK, so porting the algorithm to this in the future is easy.
- Convex-optimization solvers and modelling languages (see Section 8.2) are readily available in MATLAB, whereas modelling in C++, for instance, is many times more complicated.
- At this point there is no "need for speed". The thesis focuses on algorithm development, not mission analysis. For future more practical uses it would be beneficial to have a faster C++ implementation, though.

Owing to its importance, the guidance block is further broken down in Figure 8.3. This chart shows the steps that need to be taken internally to produce a guidance output. Also indicated are the software tools needed to accomplish each step.

COMPUTE DYNAMICS Based on the initial conditions, the dynamical system needs to be initialized ($\tilde{S}_x$, $\tilde{S}_u$, $\tilde{\boldsymbol{X}}_g$). This does not need any tools other than MATLAB.

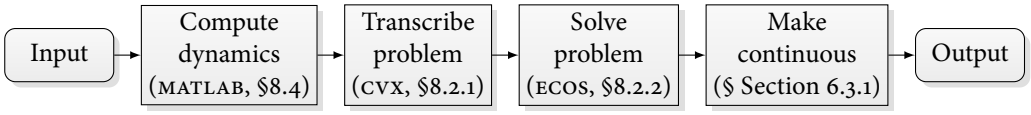


FIGURE 8.3 Computational flow chart of the guidance algorithm.

TRANSCRIBE PROBLEM The optimization problem that has previously been formulated mathematically now needs to be converted to a machine-readable format. There are tools which greatly ease this, and the particular choice – CVX – is introduced in Section 8.2.1.

SOLVE PROBLEM The solution to the transcribed problem needs to be found by a solver. The Embedded Conic Optimization Solver (ECOS) selected for this purpose is discussed in Section 8.2.2.

MAKE CONTINUOUS Finally, the generated discrete-time output needs to be converted to a continuous-time command. This has been touched upon in Section 6.3.1.

Throughout the following sections these functions will be treated separately, starting with numerical optimization.

8.2 NUMERICAL OPTIMIZATION

Numerical optimization entails the solution of the problems derived in this thesis. Generally, this is considered an established technology, as convex-optimization solvers have reached a high level of maturity in the previous decade [Boyd and Vandenberghe 2004]. It would be excessive and extremely complicated to develop a dedicated solver for this purpose, while several good options are already available. The required tools are introduced in the following sections.

8.2.1 Convex-optimization modelling

The most important step in solving the guidance problem numerically is its transcription to a format that can be read by a solver. For reference, the most basic CFTOC in this thesis, Problem 6 from Page 80 is repeated below:

▶ PROBLEM 10 Convexified, discrete soft-landing CFTOC with changed control- and constraint-variables.

$$
\begin{aligned}
&\text{minimize } \boldsymbol{e}_{N\sigma\Delta t}^{\mathsf{T}}\boldsymbol{U}\\
&\text{subject to}\\
&\|E_{\boldsymbol{\tau}}E_{k,U}\boldsymbol{U}\| \leq \boldsymbol{e}_{\sigma}^{\mathsf{T}}E_{k,U}\boldsymbol{U}, \quad k = 1, 2 \ldots, N\\
&\boldsymbol{\mu}_{1N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right) + \tfrac{1}{2}\left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{1N}\right)^2\right] \leq E_{\sigma}\boldsymbol{U} \leq \boldsymbol{\mu}_{2N}\left[\mathbf{1}_N - \left(E_z\tilde{\boldsymbol{X}} - \boldsymbol{z}_{2N}\right)\right]\\
&\boldsymbol{z}_{2N} \leq E_z\tilde{\boldsymbol{X}} \leq \boldsymbol{z}_{1N}\\
&E_{\boldsymbol{x}(N)}\boldsymbol{X} = \begin{bmatrix}\boldsymbol{r}_f^{\mathsf{T}} & \dot{\boldsymbol{r}}_f^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}
\end{aligned}
$$

There are two options for transcribing the problem: doing this by hand, or using a modelling language. To show why the former option is impractical for prototyping, this is discussed first. The problem with transcribing the problem by hand is that the matrices in the optimization problem need to be completely rewritten. This is a formidably time-consuming process and hinders prototyping.

To explain this, first consider the standard form of the SOCP that is used for all derivations:

$$\text{minimize } \boldsymbol{f}^\mathsf{T}\boldsymbol{x} \tag{8.1}$$

$$\text{subject to } \|A_i\boldsymbol{x} + \boldsymbol{b}_i\| \leq \boldsymbol{c}_i^\mathsf{T}\boldsymbol{x} + d_i, \quad i = 1, 2, \ldots, m \tag{8.2}$$

$$F\boldsymbol{x} = \boldsymbol{g} \tag{8.3}$$

This is *not* the form of the problem needed by the solver. It requires the problem to written as:

$$\text{minimize } \boldsymbol{f}^\mathsf{T}\boldsymbol{x} \tag{8.4}$$

$$\text{subject to } A\boldsymbol{x} = \boldsymbol{b}, \quad G\boldsymbol{x} \preceq_K \boldsymbol{h} \tag{8.5}$$

which is called the *canonical form* [Ben-Tal and Nemirovski 2001]. In this form there are no m inequality constraint equations anymore, but instead they have been transformed into a single *generalized inequality* over the cone K, written as $\preceq_K$ and only containing matrices [Boyd and Vandenberghe 2004]. The matrices from this report would thus have to be rewritten accordingly, which is explained in [Ben-Tal and Nemirovski 2001, 79f.].

Fortunately, this step can be avoided by using third-party add-ins to MATLAB that do the work for the user. There are two options: YALMIP [Löfberg 2004] and CVX [Grant and Boyd 2014; Grant and Boyd 2008]. Both tools accomplish the same goal in slightly different manners. The latter option has been selected for this research for a multitude of reasons: i) its syntax is simpler; ii) it is being actively developed; iii) it has a large user-community that responds quickly to support requests; and iv) it appears to perform equally well or better than YALMIP.

To give an example application of CVX and to demonstrate its ease-of-use, the MATLAB implementation of the optimization problem listed above is shown below:

```
%% Solve the problem using CVX
cvx_begin quiet
    % Set the solver and its precision
    cvx_precision high % low, default, medium, high, best
    cvx_solver(solver) % ecos, sdpt3 sedumi

    % Optimization variable
    variable U(N*4); % ([tau, sigma] x N)

    % Objective function
    minimize(esigmadtvec' * U);

    subject to
    % Second-order cone constraint relating sigma to tau
    for i=0:N-1
        % Apply at individual time steps by choosing u(k) at k=1,...,N
        i4 = i*4;
        EkuCols = [i4+1 i4+2 i4+3 i4+4];
```

```
19              Eku = sparse(EkuRows, EkuCols, EkuVals, 4, N4, 4);
20
21              % Constraint relating tau to sigma (Gamma to T)
22              norm(Etau * Eku * U, 2) <= esigma' * Eku * U;
23          end
24
25          % Thrust-bound constraint, limits control magnitude
26          mu1vec .* ( onesN1 - (Ez * (xgN + SuN * U) - z0Tlovec)...
27          + halfvec .* square(Ez * (xgN + SuN * U) - z0Tlovec) ) ...
28          <= Esigma * U <= ...
29          mu2vec .* (onesN1 - (Ez * (xgN + SuN * U) - z0Thivec));
30
31          % Inequality constraint on mass (must stay in feasible range)
32          z0Thivec <= Ez * (xgN  + SuN * U) <= z0Tlovec;
33
34          % Constrain final attitude to be vertical
35          U(end-3 : end-1) == U(end) * khat;
36
37          % Final boundary condition
38          Exf * (xgN  + SuN * U) == xf;
39      cvx_end
```

The use of literal keywords as a mnemonic in this code makes the use of CVX intuitive.

One major downside of using modelling languages is that they mean a huge performance hit to the optimziation. Auto-generating the matrices is very time-consuming. This is disussed further in Section 8.4.

8.2.2 Solver selection

Next to the modelling language, a solver to actually obtain the solution to the optimization problem is needed (see Line 6 in the listing above). An extensive literature survey on solvers was part of the preparation on this thesis [Gerth 2013]. The results of this survey are summarized in Table 8.1. CVX ships by default with the open-source solvers SEDUMI and SDTP3. Configuration files for the commercial solvers GUROBI and MOSEK are also provided out-of-the-box. The modelling language is also compatible with ECOS.

The trade-off in [Gerth 2013] concluded with the recommendation of using ECOS for this research. There are several good reasons for this:

- The solver is an *Embedded Conic Optimization Solver (ECOS)*. It is thus intended by design for use in embedded systems, which spacecraft hardware may be considered as well. This leads to a very minimal and efficient design of the solver (library free, only 750 lines of Ansi-C code).
- The solver has many interfaces, including CVX, YALMIP, Python, Julia, and Ansi-C. It can also be used for future more efficient implementations.
- ECOS is being actively developed at the moment and has a growing user base with great support.
- Most importantly, it appears to be the fastest solver available and is specialised on SOCPS.

To verify the latter claim, Problem 6 has been solved ten times for 72 temporal nodes. This results in a problem with 870 variables and 429 equality constraints. ECOS manages

TABLE 8.1 Overview of SOCP solvers. Sources: [Mittelmann 2008] [Mittelmann 2012] [Mittelmann 2013] [Pólik et al. 2010]

Solver	Compatibility	License	Remark
SEDUMI	MATLAB	open source	Used by: [Açıkmeşe and Ploen 2007]
SDTP3	MATLAB	open source	Used by: [Açıkmeşe and Blackmore 2011]
MOSEK	C, MATLAB	proprietary, free academic license	Appears to be best-performing solver on market
CPLEX	C, C++, MATLAB	proprietary	Good performance, opaque licensing scheme
LOQO	MATLAB	open source	Reasonable performance, development and community seem inactive, favours sparse matrices
GUROBI	C, C++, MATLAB	proprietary, free academic license	Good performance, not mentioned in peer-reviewed papers
ECOS	C, MATLAB, Python, Julia	open source	For embedded systems, works well on desktop, fast for small problems, new and actively developed

to solve this in 1.0 s on average, whereas the other two solvers need about 1.4 s, which are thus 40 % slower. On a few thousand MC runs, this can make a significant difference. Note that for SDPT3 a large spread in the execution time has been observed even for the same problem, so it does not appear to be recommendable.

The code for ECOS is available under an open-source license at https://github.com/ifa-ethz/ecos. In the process of this research the solver has been tested extensively, as a result of which several bugs in the code could be fixed. The general robustness of the solver has been found to be flawless, as it has never failed during any simulation.

8.3 STATE PROPAGATION

The equations governing the landing trajectory of a planetary lander are of non-linear nature and were discussed in Section 3.4. Although the fundamental two-body problem possesses an analytical solution in the form of Kepler's equations, no all-encompassing solution exists for the three-dimensional case of a thrusting spacecraft with variable mass, possibly under the influence of perturbations. For the simulation of a D/L phase one is thus forced to resort to numerical routines to integrate the non-linear models.

Numerical integration methods have been studied at length in [Gerth 2013], where single-step and multi-step integrators have been studied. In the scope of this research, numerical integration is only necessary for integrating continuous-time simulations, which make up a very small fraction of the work. For this and other reasons, a simple-yet-sufficient method has been selected, that is discussed in the following.

8.3.1 Elementary problem statement

Consider a system of first-order, ordinary, non-linear differential-equations in continuous time:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, t), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{8.6}$$

where $f : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$. It is the objective of *numerical integration* to find the state trajectory that represents an approximation to the solution at discrete points in time, that is:

$$\boldsymbol{x}(t) \approx \boldsymbol{x}_i = \int_{t_0}^{t_0+nh} f(\boldsymbol{x}, \tau)\, \mathrm{d}\tau \quad t \in [t_0, t_0 + nh], \quad i = 1, \ldots, n \tag{8.7}$$

where h is the step-size and n is the number of steps that defines the interval that is to be integrated over.

8.3.2 State derivative

The essential function in numerical integration is the state derivative, which appears as $\dot{\boldsymbol{x}}$ in Equation [8.6]. This is the system model that is propagated by the integrator. Here it models the EOM as:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{r}} \\ \ddot{\boldsymbol{r}} \\ \dot{m} \end{pmatrix} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{a}_g + \boldsymbol{a}_c \\ -\|\boldsymbol{a}_c\| m / g_0 I_{\mathrm{sp}} \end{pmatrix} \tag{8.8}$$

where $\boldsymbol{x} \in \mathbb{R}^7$. For more details see Chapter 3.

8.3.3 Integration scheme

Because of its good compromise between accuracy and computational effort, the most-applied integration method in engineering is the fourth-order Runge-Kutta integrator (RK4) [Iserles 2009]. For want of its simplicity and the sufficient performance, it is also selected as integrator here. The integration scheme reads:

$$\begin{aligned} \boldsymbol{k}_1 &= f\left(\boldsymbol{x}_i, t_i\right) \\ \boldsymbol{k}_2 &= f\left(\boldsymbol{x}_i + h_i \boldsymbol{k}_1/2, t_i + h_i/2\right) \\ \boldsymbol{k}_3 &= f\left(\boldsymbol{x}_i + h_i \boldsymbol{k}_2/2, t_i + h_i/2\right) \\ \boldsymbol{k}_4 &= f\left(\boldsymbol{x}_i + h_i \boldsymbol{k}_3, t_i + h_i\right) \end{aligned} \tag{8.9}$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{h_i}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right) \tag{8.10}$$

The rationale for using the RK4 is:

- It is fast enough for this application. Few continuous-time simulations are run, and the integration times at a maximum of 650 s are short. Thus, a small step-size to reduce errors may be chosen and the simulation can still be quickly executed.

- Typical use-cases for which variable step-size or variable-order integrators yield significant performance gains, such as highly elliptical orbits with long coasting arcs, do not apply to this problem where a brief flight-leg with continuous thrust is modelled.
- The integration scheme is easy to verify and mathematically not too challenging, giving insight for debugging *etc.*
- Because it is a single-step integrator it does not rely on interpolating polynomials through previous steps. It can thus cope much better with bang-bang control solutions, which cause problems in multi-step methods [Allione et al. 1968].
- It is simple to implement and can be easily modified in a prototyping environment.

As numerical integration is essentially a well-established default-procedure in engineering and is non-critical to this research, it does not deserve any further treatment here. The interested reader is referred to [Iserles 2009; Lambert 1991] for in-depth mathematical introductions to the topic.

8.4 PERFORMANCE CONSIDERATIONS

When developing a guidance algorithm that is supposed to be suitable for on-board implementations eventually, it is essential to make it as efficient as possible *from the beginning on*. In the early stages of the project, it is most important to focus on optimizing the performance on the algorithmic side as much as possible. This will lead to a global speed increase irregardless of the implementation, and is therefore more important than the choice of the programming language. Focussing on theoretical algorithm development, this is thus one of the reasons why MATLAB is used in this thesis. A faster C++ implementation, for instance, can be developed in future work.

In the course of the thesis it quickly became clear that the dynamics model is the most important bottle neck of the algorithm. This will be shown in the following sections, and an efficient algorithm to increase performance is proposed.

8.4.1 Analysis

To analyze which parts of the algorithm contribute the most to the central processing unit (CPU) runtime, 50 runs have been profiled in MATLAB. This showed that out of the total runtime of 658 s, *90 %* of the time is spent in CVX. As stated before, the ease of prototyping thus comes at a huge price to pay: a major performance hit. However, this is a known and accepted issue, that can be completely eliminated by switching to the canonical form of the optimization problem, see Section 8.2.1.

At 9 %, the second largest portion of the run-time was spent in the dynamics model functions that generate the matrices S_x and S_u to find the solution $\bar{\boldsymbol{X}}_g$. This still a substantial fraction of the time, and this is expected to increase manifold as soon as CVX is not involved anymore. It is therefore sensible to compute the dynamics model as efficiently as possible. For this reason, an algorithm that minimizes the number of floating-point operations (flops) is presented in the sequel.

However, the runtime is not the only performance metric that should be considered. Essential is also the memory use. In this regard the dynamics model is critical as well. As the system matrices scale with the order of N^2, the model can also cause memory issues. To give an example, the stacked system matrix S_x is of dimension $[7(N+1)]^2$. If there are 500 nodes this matrix will thus require 98 MB of memory if it is stored in dense form using double precision. Because at 12.5 % for S_u and <1 % for S_x the densities of the matrices are minimal, though, there is a perfect use-case for sparse matrices. In this case the memory use of S_x can be reduced by three orders of magnitude, from 98 MB down to 0.09 MB. A sparse-matrix computation algorithm is discussed in the following as well.

8.4.2 Minimal-flop dynamics-model computation

The computation of the LTV system matrices involves a substantial number of matrix flops. The system was given in Equation [7.52] (see Page 134) as:

$$\underbrace{\begin{pmatrix} \boldsymbol{x}(0) \\ \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(N) \end{pmatrix}}_{\boldsymbol{X}} = \underbrace{\begin{bmatrix} I & & & & \\ A_0 & & & & \\ A_1A_0 & & & & \\ \vdots & \ddots & & & \\ A_{N-1}A_{N-2}\cdots A_0 & \cdots & A_1A_0 & A_0 & I \end{bmatrix}}_{S_x} \underbrace{\begin{pmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \boldsymbol{x}_0 \\ \vdots \\ \boldsymbol{x}_0 \end{pmatrix}}_{\boldsymbol{X}_0} \cdots$$

$$\cdots + \underbrace{\begin{bmatrix} 0 & & & & \\ B_0 & & & & \\ A_1B_0 & B_1 & & & \\ A_2A_1B_0 & A_2B_1 & B_2 & \ddots & \\ \vdots & \vdots & \vdots & & \\ A_{N-1}A_{N-2}\cdots A_1B_0 & \cdots & \cdots & \cdots & B_{N-1} \end{bmatrix}}_{S_u} \underbrace{\begin{pmatrix} \boldsymbol{u}(0) \\ \boldsymbol{u}(1) \\ \boldsymbol{u}(2) \\ \vdots \\ \boldsymbol{u}(N-1) \end{pmatrix}}_{\boldsymbol{U}}$$

Especially the stacked input matrix contains a large number of matrix products. The simplest approach would be to type in the matrices in the equation above as-is. This has been tried and turned out to be the least efficient approach by far.

An important observation is that there are common elements in S_u and S_x. This can be made use of when designing the computation algorithm. Moreover, one may notice that in S_u only two new elements appear per row i: B_i and A_i.

An algorithm that makes use of these facts and thus yields a speed gain of two orders of magnitude (on average) is shown in Figure 8.4.[1] The idea is to first compute a *temporary* matrix that holds the common elements. Within this matrix, only the system matrix corresponding to the time step of each row is newly calculated. This matrix is then multiplied with all other elements in the row above it, which is the minimal number of flops possible.

1 The exact speed gain depends on the size of the matrix, because it grows with $\mathcal{O}(N^2)$, and so does the number of flops.

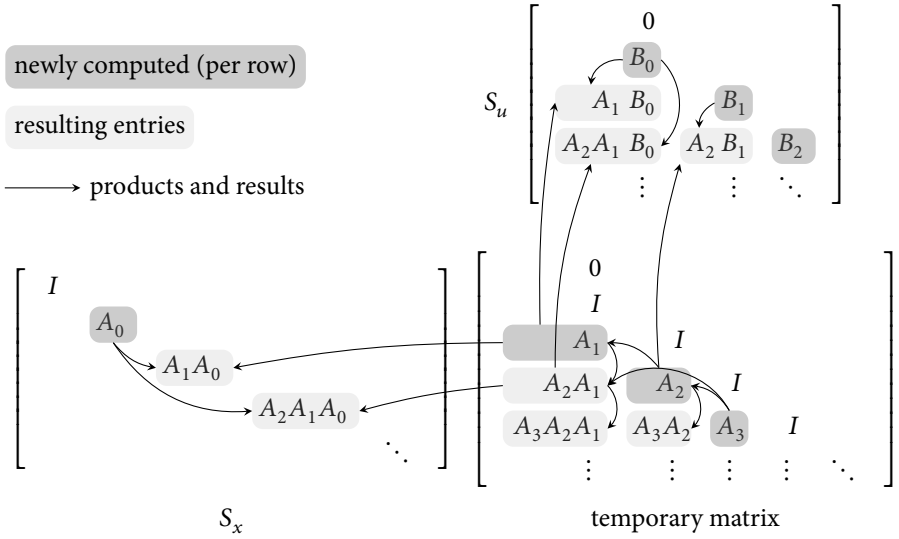


FIGURE 8.4 Algorithm for computing the stacked system and input matrices using the minimum number of multiplications. A temporary matrix holds pre-computed common elements.

To find the stacked system matrix from the temporary matrix, all cells from the first column of the temporary matrix are copied into the stacked system matrix. Each of these values is then multiplied by the system matrix for the initial conditions, A_0.

To find S_u all nonempty cells from the temporary matrix are reused. These are copied to the same column/row position as in the temporary matrix, and are then multiplied by the relevant input matrix B_i. Note that as such, the input matrix corresponding to a time-step also only needs to be computed once.

No algorithm with less flops could be thought of. The next section elaborates on how this can be implemented in a sparse fashion.

8.4.3 Sparse-matrix implementation

Apart from reducing the flops, the memory usage of the dynamics model is to be significantly reduced by harnessing sparsity properties. The easiest way to discuss this is to show relevant code snippet-by-snippet. This is deemed important, because several further hints to reduce flops become clear by looking at the code. The full code is available upon request on github.com.

First of all, to sparsely define each of the matrices three vectors are needed: one containing the rows of values, another containing the column positions, and a final one with the corresponding values. These are initialized as:

```
SxRows(1:7) = 1:1:7;
SxCols(1:7) = 1:1:7;
SxVals(1:7) = ones(7,1);
```

Note that the first entry in S_x is just the identity matrix, so it can be initialized straight away.

Now the main loop is executed, which loops from the first row to the last row. In the process of this, the system and input matrices corresponding to each row (see Fig-

ure 8.4) are computed, which is only necessary once using another function. The relevant state is extracted from the stacked state vector $\boldsymbol{X}$ that is an input.

```
for i=1:N
    rCube = sqrt(X((i-2)*7 + 8)^2 + X((i-2)*7 + 9)^2 + X((i-2)*7 + 10)^2)^3;
    [ Arow, Brow ] = discrete_input_spherical( mu, rCube, dt, g0, Isp );
```

One crucial observation is that the matrices A_d and B_d contain only five and three unique entries, respectively, that appear on diagonals of three entries, retaining only these values can thus save another threefold intermediate memory and flops. For the input matrix corresponding to the relevant row, this is:

```
    BdRowVals(BdRowCount+1) = Brow(1,1);
    BdRowVals(BdRowCount+2) = Brow(4,1);
    BdRowVals(BdRowCount+3) = Brow(7,4);
    BdRowCount = BdRowCount + 3;
```

Next it is looped over the columns, starting on the right-most nonzero column. Now the unique cell entries in the temporary matrix can be calculated. In the code below, `Arow` is the system matrix of the current row and `TmpVals` reads the previously computed products like $A_3A_2\cdots$ from the sparse value vector. In other words, all values in the temporary matrix are computed, where `Arow` are the dark grey values in Figure 8.4, and `TmpVals` are the light grey cells in Figure 8.4:

```
    for j=(i-1):-1:0
        if (i-j) ~= 1
            Aij11 = Arow(1,1) * TmpVals(SuTmpCount-10-(i-1)*13) ...
                  + Arow(1,4) * TmpVals(SuTmpCount-2-(i-1)*13);
            Aij41 = Arow(4,1) * TmpVals(SuTmpCount-10-(i-1)*13) ...
                  + Arow(4,4) * TmpVals(SuTmpCount-2-(i-1)*13);
            Aij14 = Arow(1,1) * TmpVals(SuTmpCount-5-(i-1)*13) ...
                  + Arow(1,4) * TmpVals(SuTmpCount-7-(i-1)*13);
            Aij44 = Arow(4,1) * TmpVals(SuTmpCount-5-(i-1)*13) ...
                  + Arow(4,4) * TmpVals(SuTmpCount-7-(i-1)*13);
            Aij77 = Arow(7,7) * TmpVals(SuTmpCount-6-(i-1)*13);
        else
            Aij11=1; Aij41=0; Aij44=1; Aij14=0; Aij77=1;
        end
```

Note how for the first column respectively the identity matrix is computed.

At this point the matrix S_u can be computed, which is made up of entries that are products of A_d and B_d:

```
        AijBij11 = Aij11 * BdRowVals(j*3 + 1) ...
                 + Aij14 * BdRowVals(j*3 + 2);
        AijBij41 = Aij41 * BdRowVals(j*3 + 1) ...
                 + Aij44 * BdRowVals(j*3 + 2);
        AijBij74 = Aij77 * BdRowVals(j*3 + 3);
```

Now the data is written to the vectors holding the columns, rows, and values. To further reduce flops and memory, this is written out to the full:

```
        i7 = i*7; j7 = j*7; j4 = j*4;
        SuVals(SuCount+1) = AijBij11;
        SuVals(SuCount+2) = AijBij11;
        SuVals(SuCount+3) = AijBij11;
        SuRows(SuCount+1) = 1 + i7;
        SuRows(SuCount+2) = 2 + i7;
        SuRows(SuCount+3) = 3 + i7;
        SuCols(SuCount+1) = 1 + j4;
        SuCols(SuCount+2) = 2 + j4;
        SuCols(SuCount+3) = 3 + j4;
```

etc.

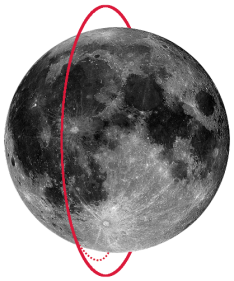
For the first column of the temporary matrix (which corresponds to $j = 0$), the values are copied into the stacked system matrix and multiplied by A_0:

```
        if j==0
            ASx11 = Aij11 * Ad0(1,1) + Aij14 * Ad0(4,1);
            ASx41 = Aij41 * Ad0(1,1) + Aij44 * Ad0(4,1);
            ASx14 = Aij11 * Ad0(1,4) + Aij14 * Ad0(4,4);
            ASx44 = Aij41 * Ad0(1,4) + Aij44 * Ad0(4,4);
        end
```

This completes the computation of all vectors, which allows for finally writing the sparse matrices. This requires the specification of: i) the row vector; ii) the column vector; iii) the values vector; iv) the total number of rows; v) the total number of columns; and vi) the total number of nonzero elements. As such, the matrices are created in MATLAB with the following command:

```
Su = sparse(SuRows, SuCols, SuVals, 7*(N+1), 4*N, N*(N-1)/2*7+N*7);
Sx = sparse(SxRows, SxCols, SxVals, 7*(N+1), 7*(N+1), 13*N+7);
```

This completes the algorithm for computing the dynamics.



Chapter 9

DESIGN AND ANALYSIS OF AN OPTIMAL SCENARIO

> "The physics of lunar landing have not changed since Apollo."
>
> *Paschall et al. [2009]*

The development of a guidance algorithm is not complete without putting it to test in a relevant scenario. However, the definition of such a scenario is less than straightforward. Ideally, boundary conditions that would allow for direct comparison to other guidance algorithms should be used. Unfortunately this is easier said than done. The problem is that guidance can hardly be considered in an isolated fashion. Guidance drives the choice of vehicle hardware and assumptions, and vice versa. A good scenario is therefore tailored after a choice for guidance. In light of this, the comparison of convex guidance to the ESA LL baseline is impossible, because a different propulsion configuration has been assumed (Section 2.2).

It was therefore agreed that rather than trying to reproduce results from the ESA LL, convex guidance should be studied in detail under consideration of broader aspects.[1] As such, in this section a limited analysis of an entire baseline scenario from PDI up to TG is designed. This is based on the LL, but not identical to its scenario. The general capabilities of the algorithm will be explored first, and a specific trajectory will then be studied in more detail to be able to judge aspects such as sensitives and robustness.

The chapter is organized in the following manner. The rationale and process for designing a scenario is explained in more detail in Section 9.1. Section 9.2 explains the design of the approach gate. The main braking phase with the powered descent initiation is treated in Section 9.3. After choosing an exemplary scenario, this is then simulated and studied in detail in Section 9.4. Some specific recommendations are given in the last part, Section 9.5.

9.1 DESIGN ELEMENTS AND RATIONALE

The development of the new Moon-landing guidance-algorithm presented in this thesis set out with three ambitious goals: i) consolidate main-braking and approach-phase guidance into a single algorithm, to reduce the GNC-system complexity; ii) derive an on-board capable algorithm that can be highly constrained; and iii) delivering a globally optimal control-solution in terms of the required propellant mass. Although a purely theoretical analysis and related results have already been presented, these claims still lack proper verification. To demonstrate the applicability of the algorithm, it has to be run in a relevant simulated environment, in other words: it has to be executed for a

1 Personal communication with Christian Philippe and Erwin Mooij.

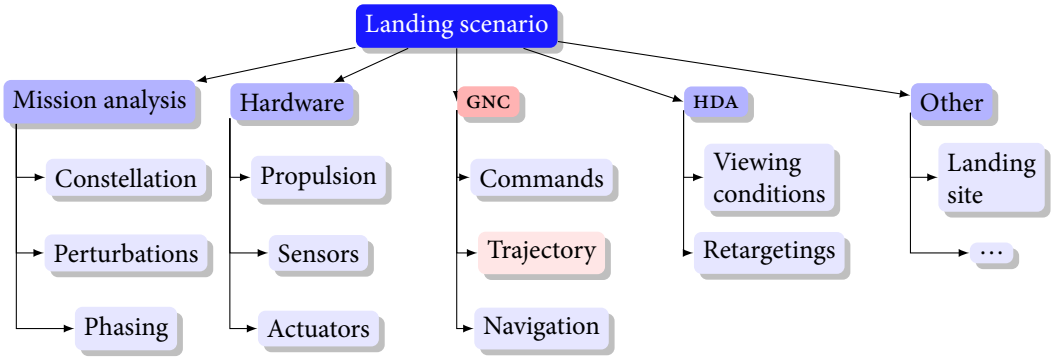


FIGURE 9.1 Some factors that need to be considered and optimized when designing a landing scenario. The problem is highly multidisciplinary. This is a simplified, inexhaustive selection of influence factors.

mission scenario. Any analysis neglecting a relevant mission scenario would give inconclusive results about guidance. Unfortunately, the consideration of a mission scenario leads to further significant complications. The following will detail why the design of a dedicated mission scenario is necessary, and explain how this will be done.

9.1.1 Need for designing a scenario

A mission scenario is made up of several elements, of which guidance is an integral part. To get an impression of what plays a role, a simplified chart is shown in Figure 9.1. Two examples forThe chapter is organized in the following manner. The rationale and process for designing a scenario is explained in more detail in Section 9.1. Section 9.2 explains the design of the approach gate. The main braking phase with the powered descent initiation is treated in Section 9.3. After choosing an exemplary scenario, this is then simulated and studied in detail in Section 9.4. Some specific recommendations are given in the last part, Section 9.5. decisive elements are the vehicle design in terms of propulsion and sensors, and the dedicated mission analysis for designing orbits and trajectories to link vehicle capabilities to physical constraints, such as constellation requirements for solar-panel illumination and communications. This makes the mission design a highly interdisciplinary process. Guidance is inherently connected to mission analysis, because it *commands* the relevant trajectories (see also Section 4.3). These trajectories result from two major variables: i) the guidance algorithm and its tuning parameters themselves; and ii) the gates (like PDI, AG, and TG) that define the boundary conditions for guidance.

Knowing that mission scenarios are highly dependent on the chosen guidance algorithm and gates, the verification of the goals formulated for this research becomes complicated. Any mission scenario that has been designed and is documented in the literature or by an agency has been designed with a specific guidance algorithm in mind. There is no landing scenario without guidance and no need for guidance without a scenario. However, *as algorithms differ, a comparison of the different guidance algorithms for the same scenario becomes an unfair and inconclusive trade-off*. To give an example, guidance algorithms are typically designed for the chosen engine configuration of a lander. They can work for fixed-thrust engines, for throttleable engines, or for a suc-

cessive engine-shutdown strategy. The mission scenario is designed according to this, and the guidance algorithm can therefore not simply be substituted into a given framework. *A re-design of the scenario is necessary* to have a fair trade-off. The performance metrics of the re-designed scenario can then be compared, given that the same amount of work went into both.

Unfortunately, this makes a fair analysis for the baseline ESA LL scenario using convex guidance impossible. The reason lies in the propulsion configuration: the LL relies on non-throttleable engines, that are to be shut-down sequentially. However, it was shown in Section 4.3 that the theoretical optimum trajectory results from bang-bang control, which is also the output from convex guidance. Such a thrust profile cannot be accommodated by the ESA LL's propulsion system. For this reason, a different propulsion configuration needs to be assumed, which is documented in Section 2.2. To present a fair analysis of convex guidance, this chapter demonstrates a possible design logic and its results.

9.1.2 Scope of the analysis

As Figure 9.1 made clear, the great variety of influence factors makes the design space for a landing scenario excessively large. A full analysis requires a team of experts, and is beyond the scope of an MSc thesis that focuses on theoretical algorithm-development. However, this does not mean that no meaningful analysis can be done. On the contrary, with a slightly limited scope and working with the baseline scenario of the ESA LL, a realistic scenario can be designed.

Key features

To make a conscious choice for the parameters to be studied, the critical features for landing scenarios must be identified. Generally, a trajectory should be analysed from at least the following perspectives:

Viewing conditions For hazard detection and vision-based navigation the viewing conditions must be right in terms of illumination and viewing angle. Were it just for the viewing conditions, the optimal trajectory would be to descent vertically to the landing site from a high altitude.

Retargeting capabilities Diversions to safe sites can be easier to accommodate on some trajectories than on others. For example, in a scenario with a high down-range speed-component, a short downrange-retargeting will be costly.

Robustness The design of the gates and time-of-flight can have an influence on the robustness. If a gate is designed to be on the edge of a feasible envelope, small state errors can lead to infeasible trajectories.

Landing precision Of course, the landing precision plays a role. This can be affected by the speed during the approach.

Propellant optimality The obvious criterion is propellant optimality. For any kilogram saved, on paper at least more than a kilogram in extra payload can be landed on the surface. For the ESA LL, a mere 6 kg of the initial 2444 kg wet mass is allocated to a payload, which is a payload mass-fraction of below 0.25 % [Diedrich 2011]. Any propellant savings will thus be a significant benefit.

Assumptions and project constraints

With regard to these points, the following constraints immediately limit the scope:

No retargetings Retargetings for safe-site diversions will be ignored for several reasons. i) No simulation environment with HDA features is available to TU Delft. ii) The additional options for choosing retargeting gates would further complicate the problem. This analysis is not feasible within the allotted time. However, the trajectory will be designed with retargetings in mind, as some choices for gates are obviously worse or better than others.

No sensor-models in-the-loop Related to the previous point, no vision-based sensors will be modelled. This means that no conclusive information of the trajectory's impact on the data quality can be made. However, these interactions have been documented in the literature [Crane and Rock 2012], and can thus still be evaluated to reasonable certainty.

No perturbations Perturbation such as gravity-field irregularities or third-body influences are ignored for several good reasons documented in Section 3.5.3. Accordingly, *no trajectory-tracking controller* is implemented either to follow the reference-trajectory.

Three-DOF dynamics The rotational DOFs are ignored completely, as is commonly done in guidance studies [Açıkmeşe and Ploen 2007]. Perfect navigation and control are also assumed, although the fundamental feasibility of the attitude control is considered.

Arbitrary landing-site Because perturbations are ignored no physical location on the Moon needs to be selected on the Moon.[2]

Scope

Having defined what *is not* modelled, the points that *are* considered are listed below:

Modelled descent leg From DOI to TG.

Trajectory type Continuous, no retargetings.

Optimization criteria Propellant mass, suitability for vision-based instruments.

All other parameters are taken from the LL specifications. The detailed design space definition is discussed throughout the following sections.

9.1.3 Requirements definition

Based on the chosen scope, the relevant requirements traced from the LL scenario are defined in Table 9.1 and Table 9.2. It must be stated that the requirements for TG in Table 9.1 are actually traced from those for TD, as no specific requirements for TG have been provided. Of these requirements TG-3, that specifies the attitude desired at TG, requires some further elaboration. It would be preferable to obtain the attitude desired

2 If perturbations were taken into account, the flight path to the landing site would encounter different disturbing forces depending on the chosen location.

TABLE 9.1 Requirements for the maximum errors (3σ) at terminal gate.

ID	REQUIREMENT	SUCCESS CRITERION	RATIONALE
TG-1	Position	The error due to guidance at TG shall not exceed the following cylindric space of dimensions: horizontal $(r_x, r_y) = 0.0 \pm 1.5$ m; vertical $r_z = -1.5 \pm 1$ m.	[Philippe 2012]
TG-2	Velocity	The horizontal ground speed shall be in the following range: horizontal $v_h = 0.0 \pm 1.0$ m/s; vertical: $v_v = 1.5 \pm 1.0$ m/s	[Soppa 2011], [Philippe 2012]
TG-3	Attitude	The maximum attitude errors shall be within a range recoverable in the leg from TG to TD.	Touchdown attitude.

TABLE 9.2 Trajectory constraints.

ID	REQUIREMENT	SUCCESS CRITERION	RATIONALE
FOV-1	Centering	The landing-site shall not drift off-center of the LIDAR FOV between AG and $r_z = 150$ m.	HDA hazard mapping
FOV-2	LIDAR	The landing-site shall remain within the camera FOV from $r_z = 150$ m to 50 m.	HDA. FOV = 40°. [Kerr et al. 2013]
FOV-3	Camera	The landing-site shall remain within the LIDAR FOV from $r_z = 50$ m to TG.	Navigation. FOV = 70°. [Kerr et al. 2013]
TC-1	Fuel	The trajectory shall be fuel-optimal.	Reduces overall system mass.
TC-2	Vision	The trajectory shall optimize vision-based sensor performance.	Improves navigation, HDA.

at TD already at TG. However, with convex guidance it is not possible to guarantee this. This is because there is no attitude-rate constraint, and specifying a final attitude will lead to a jump from the second-to-last to the final step. For this reason, it is assumed that a pitch-up maneuver is performed at TG, to correct the attitude. Analysis would need to be done on how this impacts the scenario, so that no number to maximum off-vertical attitude can be specified at this point. In general, pitch-up maneuvers are not uncommon and are also used in NASA studies [Kos et al. 2010].

The first three requirements in Table 9.2 are directly traced from the LL scenario. Figure 9.2 shows the mission baseline for the approach-phase HDA activities. As can be seen, two diversions are planned: a far-range and a close-range divert. The final divert will be executed at an altitude of 150 m at the latest. This explains requirement FOV-1, as the area of the landing-site must be permanently observed to map hazards and to navigate. Requirements FOV-2 and -3 are navigation related.

Requirement FOV-1 also needs further elaboration. Strictly speaking, the requirement defined by ASTRIUM[3] is to *fix* the landing site at the center of the FOV. Using

3 Now Airbus Defense & Space.

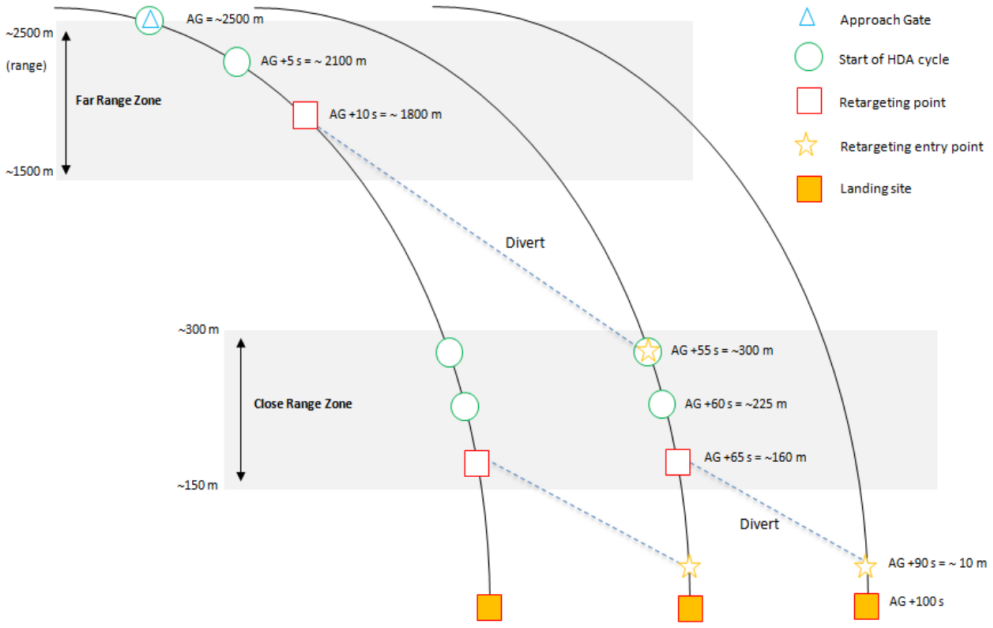


FIGURE 9.2 The ESA LL approach-phase HDA baseline scenario. Source: [Kerr et al. 2013, p. 5].

an undocumented algorithm that presumably applies NLP[4], both ASTRIUM and its subcontractor DEIMOS ENGENHARIA SA manage to achieve this requirement independently from each other. However, this is mathematically impossible using convex optimization theory (see Section 7.4). Therefore, this requirement is relaxed to a slight tolerance, which should still be acceptable. The precise value, now assumed as 5°, would need to be determined in a more detailed study.

Even though no diversions will be considered in this study, it is important to demonstrate that the algorithm would be applicable to scenarios applying vision-based technologies. For this reason, requirements satisfaction shall be demonstrated for the nominal scenario. Finally, requirements AC-1 and -2 are defined to free the trade-off from arbitration, as otherwise any trajectory satisfying the requirements would be considered equally good.

9.1.4 Definition of the scenario and design space

Having defined requirements, a scenario that can fulfill these can be proposed. Not surprisingly, this can be traced from the ESA LL again to allow for (limited) comparability. The part of the mission considered in this chapter is shown in Figure 9.3.

To be analyzed are the main braking/descent phase from PDI to AG and the approach phase from AG to TG (see also Section 4.5.1). This limited scenario can verify all three research goals:

4 During the extensive literature study, see [Gerth and Mooij 2014], no algorithms that could fulfill this requirement could be found in the literature, apart from those using classical numerical methods such as NLP.

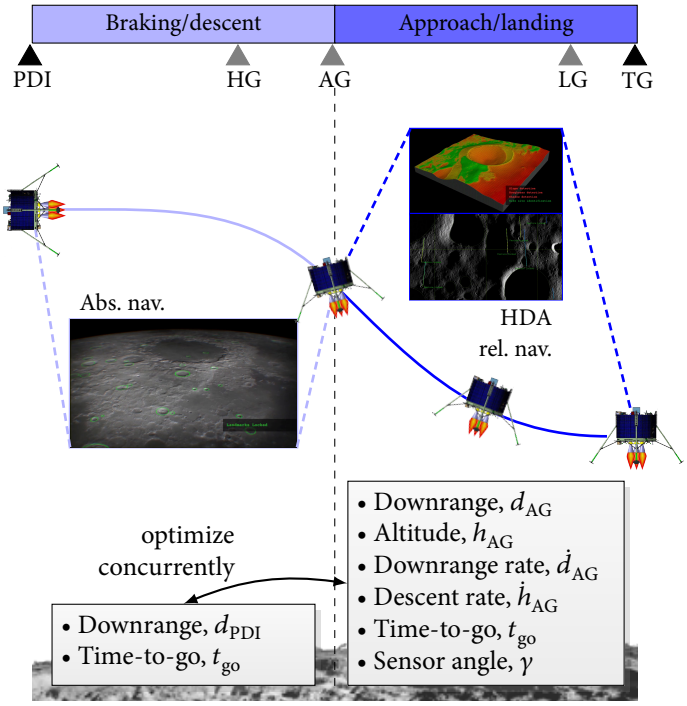


FIGURE 9.3 Elements of the to-be-designed scenario. The parameters in the boxes are the decision variables of the two phases.

Consolidation Starting with main braking and ending with the approach, the algorithm can be tested for both phases. Anything prior or after (such as DOI or TD) is ignored and irrelevant.

Constraints During the approach phase the constraint capabilities can be verified.

Optimality Adding the propellant masses of both phases will give a relevant figure of merit.

To be designed are the initial conditions at AG and at the PDI that guidance takes as input and based on which it generates a trajectory. At PDI these are the downrange and time-of-flight, see Figure 9.3. The altitude is given by the LL baseline. The speed is derived from the Hohmann transfer prior to PDI. For the approach phase, the initial position and speed, as well as the time-to-go, need to be optimized. Further, the sensor angle γ must be optimized to get good results in terms of vision-based sensors. The exact figures will be filled in throughout the following sections. To get an integrated scenario, both PDI and AG must be optimized *concurrently*. This is because the conditions at AG drive the performance during main braking, and vice versa.

It should be noted that other D/L architectures would, in fact, be possible as well, and should be considered if a design would start from scratch. For example, one could include a hovering-phase for HDA, as allegedly demonstrated by the Chinese Chang'e 3 mission on December 14, 2013 [Sun et al. 2013; Lakdawalla 2014].

9.1.5 Performance metrics

Reaching ahead and assuming for now that the variables shown in Figure 9.3 have been chosen with some kind of optimization (see Section 9.2 in the following), the resulting trajectory has to be evaluated. The result of this evaluation will be used for trade-offs and iterating the design. Two major goals influence the choice of performance metrics: the ambition to improve vision-based sensor performasnce and the need to reduce propellant mass.

The need to optimize these two criteria is also expressed in Requirements TC-1 and -2. The performance metric for the propellant mass is simply defined by summing up the used mass during main braking and approach:

$$J_p = \int_{t_{\mathrm{PDI}}}^{t_{\mathrm{AG}}} \dot{m}(t)\,\mathrm{d}t + \int_{t_{\mathrm{AG}}}^{t_{\mathrm{TG}}} \dot{m}(t)\,\mathrm{d}t \qquad [9.1]$$

A performance metric for the vision-based sensor-performance is much harder to define. Because the simulations are run without HDA in-the-loop, the direct performance of the system cannot be evaluated. However, the trajectory's suitability can still be evaluated based on general studies that have evaluated the impact of trajectories on HDA performance. Crane and Rock [2012] studied how different trajectories affect hazard estimation. They demonstrate that the strongest driver for the reliability of hazard mapping, apart from altitude (which drives imaging resolution), is the obliquity at which the sensors map the environment. This is illustrated in Figure 9.4. The angle of obliquity is defined in this thesis as the angle between the boresight vector of the sensors and the local vertical:

$$\xi(t) = \arccos\left[\hat{\boldsymbol{k}} \cdot \hat{\boldsymbol{b}}(t)\right] \qquad [9.2]$$

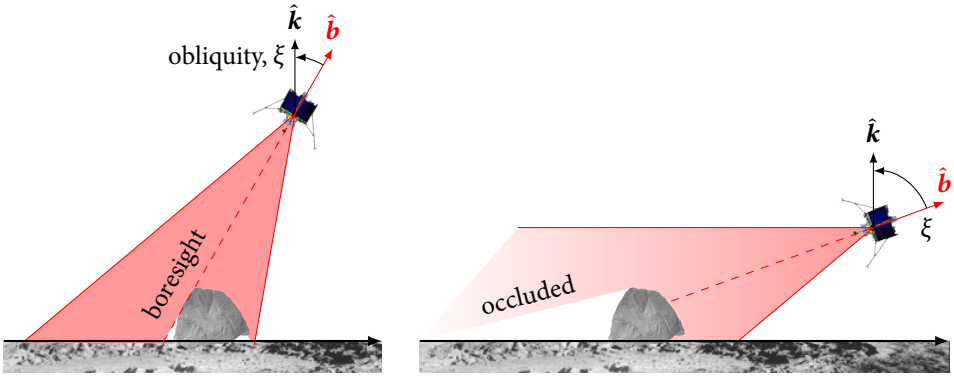
If a spacecraft were to approach a landing site straight from above, the obliquity would be $\xi = 0°$ and everything could be seen perfectly, see Figure 9.4(a). However, for large angles *occlusion* can occur, as shown in Figure 9.4(b). This means that terrain features obstruct the sensors' view, and possible hazards in the occluded area cannot be observed anymore. Moreover, the overall observable area is decreased, making navigation and hazard detection harder. It is therefore desirable to decrease the obliquity angle during the approach, and a suitable cost criterion is:

$$J_\xi = \int_{t_{\mathrm{AG}}}^{t_{\mathrm{TG}}} \xi(t)\,\mathrm{d}t \qquad [9.3]$$

One shortcoming of this metric is that it is dependent on the time-of-flight. Due to the time integration, shorter flights are favored over longer ones. A very short flight with a bad obliquity angle can be cheaper than a long flight with a low obliquity angle. On top of that, longer flights are actually preferable, because they allow for more time for hazard mapping and navigation. The cost criterion is therefore normalized as follows:

$$\bar{J}_\xi = J_\xi / t_{\mathrm{go}} \qquad [9.4]$$

This criterion essentially represents the average obliquity angle throughout the flight.



(a) Small obliquity, little occlusion **(b)** Large obliquity, heavy occlusion

FIGURE 9.4 Geometry of the obliquity angle in relation to occlusion from imperfect terrain. Occlusion due to large obliquities is a major source of performance attenuation in vision-based systems.

A second performance metric can be defined based on the viewing angle. This has been defined in Section 7.4.2.5 as the angle between the boresight and the current position vector:

$$\beta(t) = \arccos\left[\hat{\boldsymbol{r}}(t) \cdot \hat{\boldsymbol{b}}(t)\right]$$

If the viewing angle is 0°, then the boresight is aligned with the position vector and the landing site is at the center of the FOV. In a perfect world, the viewing angle would be 0° throughout the whole flight. Therefore, a suitable performance metric is:

$$J_\beta = \int_{t_{AG}}^{t_{TG}} \beta(t)\, dt \qquad [9.5]$$

As before, this is normalized:

$$\bar{J}_\beta = J_\beta / t_{go} \qquad [9.6]$$

These two performance metrics express how well a trajectory is suited for vision-based systems. In fact, both should be optimized at the same time. A joint metric should therefore be defined that expresses the total optimality of a trajectory. However, the magnitude of $\bar{J}_\xi$ and $\bar{J}_\beta$ is typically about one order of magnitude off. Assigning both parameters equal importance, a joint cost-function is therefore obtained by equalizing the means of both in the following manner:

$$J_c = \frac{E[\bar{J}_\xi]}{E[\bar{J}_\beta]} \bar{J}_\beta + \bar{J}_\xi \qquad [9.7]$$

where $E[\cdot]$ denotes the expected value (or mean, sometimes denoted as $\mu[]$). The fraction in this expression has been determined to be 7.99 based on the results shown in the sequel, and thus J_c is defined as:

$$J_c = 8J_\beta + J_\xi \tag{9.8}$$

One might interject that based on $\bar{J}_\xi$ and $\bar{J}_\beta$ the optimal trajectory would simply be a vertical descent. This is correct, but only one side of the coin. A vertical descent would disregard J_p, because a vertical descent is suboptimal. This is easy to see when considering gravity losses, that are defined in terms of Δv as:

$$\Delta v = \int_{t_0}^{t} \|\boldsymbol{g}(t) \times \hat{\boldsymbol{v}}(t)\| \, \mathrm{d}t \tag{9.9}$$

Consequently, the gravity losses are maximal in vertical flight, which conflicts the optimal conditions for $\bar{J}_\xi$ and $\bar{J}_\beta$. Indeed, in their study for HDA-optimal approach-phase trajectories, Crane and Rock [2012] do not select a vertical descent, but a side-ways approach with a landing-site flyover. The flyover causes the instruments to observe the landing site at virtually zero obliquity, significantly improving the output. This is a good trade between both worlds. The landing problem is thus a multi-objective parameter-optimization, with J_p and J_c as conflicting cost functions.

9.1.6 Summary: the design process

The previous sections have answered why a new scenario must be designed to evaluate the algorithm, have defined requirements, and have presented a baseline scenario which allowed for the definition of optimization variables and metrics. These steps are integral to the mission-scenario design-process, that is shown in Figure 9.5. This process roughly follows the basic V-model of systems engineering, though with adaptations to the problem at hand [Larson 2009]. For example, design options are not part of the study as the scenario is limited to the ESA LL baseline.

In the following sections, the design loop will be completed, starting with a look at the AG and PDI (bottom in Figure 9.5). This will lead to a trade-off for optimal guidance parameters, based on which a nominal scenario is selected. In a sensitivity analysis, the influence of the trade parameters and other variables on the solution is tested. This will lead to a refinement of the design. Finally, the robustness of the designed solution is verified using an MC analysis. Although several loops in the AG and PDI design phase have been performed, multiple full design loops are beyond the scope of this thesis. These should ideally also include re-considering other mission aspects, such as vehicle-system choices.

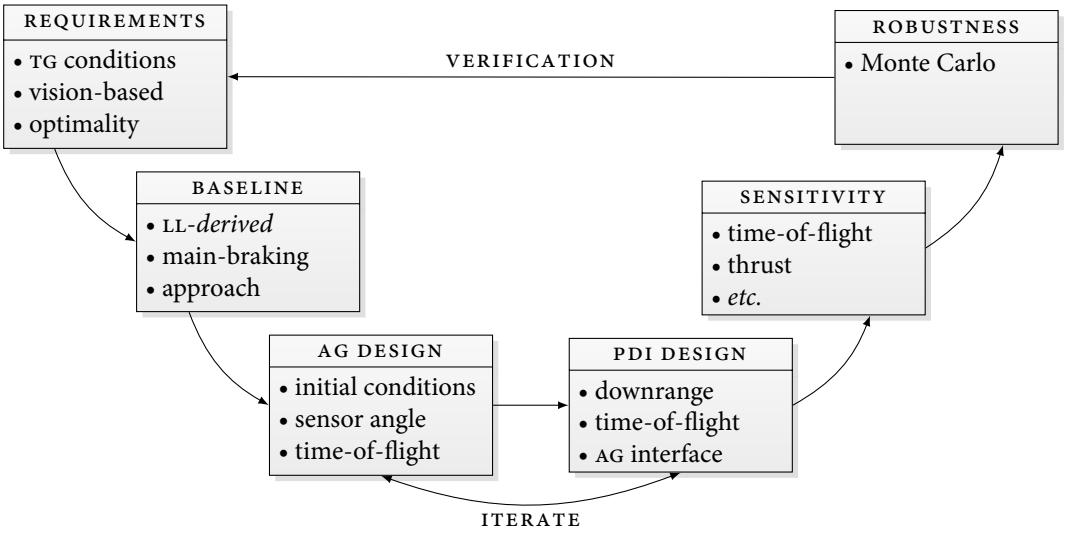


FIGURE 9.5 The design process in this chapter follows the basic V-model of systems engineering, *tailored* to the particular case. Some steps, such as design-option trades are omitted.

9.2 APPROACH-GATE DESIGN

Having defined the design process the AG can be optimized, which is the subject of this section. The results in this section are the outcome of several iterations, of which only the final one is documented. Overall, there are about seven steps to arrive at the final trajectory, which are shown in Figure 9.6.

FIGURE 9.6 Approach-gate design-flow.

1. *Definition of the search space.* Based on reference scenarios and problem insight, the search space for the optimization can be defined.
2. *Optimization.* The search space is sampled using an optimization method.
3. *Refinement.* Based on the outcome of the first optimization run, the search space can be narrowed down.
4. *Analysis.* Once a cloud of trajectories is available, the search-space is analyzed to understand the problem.
5. *Trade-off.* A candidate trajectory is selected for further analysis.
6. *Sensitivity analysis.* The candidate trajectory is studied in detail with the goal of identifying strengths and weaknesses for designing a more robust, and possibly more optimal scenario. This is essentially a local optimization.
7. *Finalization.* Ultimately, the final AG conditions are fixed.

These steps will be discussed throughout the following sections.

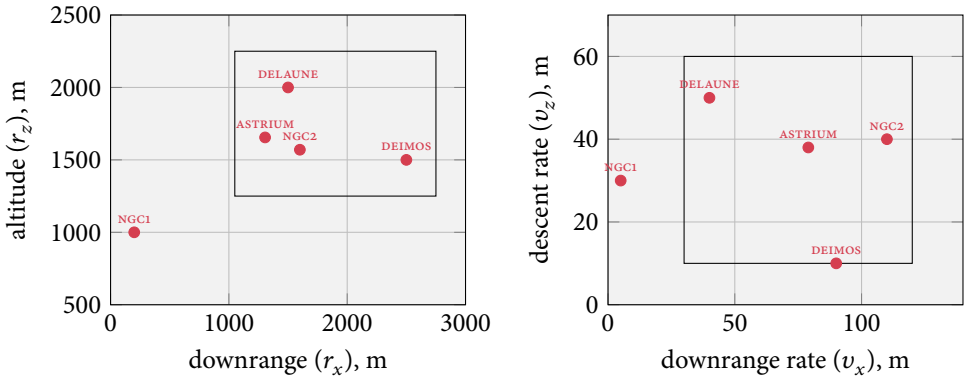
9.2.1 Definition of the search space

For the AG design six parameters need to be defined:

Initial position The initial position (downrange and altitude) in the landing-site frame, $\mathcal{L}$. The initial elevation – a critical parameter for the viewing conditions – directly follows from r_x and r_z. Nominally, there is no crossrange component.

Initial velocity The initial downrange and descent rates are design parameters as well. They determine the initial flight-path angle.

Time-to-go Strictly speaking, there one time-to-go from AG to TG that yields the propellant-optimal solution. However, this time is not known *a priori*. The time-to-go is thus also added to the search space. The true optimum can be determined through a local search later on. Moreover, the best solution in terms of propellant use is not necessarily the best solution viewing-angle wise. Some play in the time-to-go is thus in order.



(a) Initial position **(b)** Initial speed

FIGURE 9.7 Design space to be sampled for finding the optimal initial conditions. The search range is based on reference scenarios that have also been designed for vision-based approaches. For references, see Table A.1.

Sensor angle The sensor angle is an extremely sensitive parameter, that determines whether the landing region is seen during the descent or not. Trajectories that do not fulfill the viewing-angle requirements are considered infeasible. The sensor angle is therefore a critical design parameter.

Because convex guidance has not been applied to Lunar landing yet (to the best of the author's knowledge), it is hard to make a good initial choice for these parameters. Moreover, publications on the relation of the sensor angle to the trajectory also do not exist, further complicating the issue. A solution for this is to work with reference data where available and to experiment with the other parameters. To this end, relevant studies on Lunar landing have been inspected and relevant parameters have been summarized in Table A.1. To start with choosing a range of initial conditions, these have been plotted in Figure 9.7.

A quick description of the different studies shown in the plots is in order:

Astrium In-house design for the complete Phase-B1 HDA and GNC concept of the ESA LL [Diedrich 2011]. This trajectory starts at a relatively high elevation and moderate downrange rate and leads to a landing-site fly-over. Recall that Crane and Rock [2012] reported that such trajectories benefit HDA performance. The guidance law in this study is NLP-based.

NGC1 A landing scenario with HDA in the loop [Philippe 2014]. Very obviously, these AG conditions strongly differ from all others. The approach starts at exceptionally high elevation with short downrange, and essentially a straight-downwards velocity. Rather than a fly-over, the spacecraft thus approaches the landing site from the top to keep the obliquity low. This might be because Apollo guidance is used, which cannot enforce viewing constraints. Initial conditions that "naturally favor" viewing conditions must thus be chosen.

NGC2 A scenario *without* HDA [Philippe 2014]. The extremely fast downrange rate leads to a far landing-site fly-over. This might lead to a propellant-optimal scenario.

TABLE 9.3 Search space for the AG.

	r_x, m	r_z, m	v_x, m/s	v_z, m/s	γ, °	t_{go}, s
MIN	1050	1250	30	10	−10	50
MAX	2750	2250	120	60	30	90

The guidance laws are specified as powered explicit guidance (PEG) (main braking) and glide-slope guidance (approach).

Deimos Result of a GNC and HDA activity that was performed as subcontractor to ASTRIUM for the ESA LL [Kerr et al. 2013]. The scenario starts far downrange, with an almost horizontal speed. It also nominally leads to a flyover. Reportedly, this scenario uses E-Guidance.

Delaune This scenario was defined in the early stages of the ESA LL study as part of an MSc thesis, but is also cited by ASTRIUM [Delaune et al. 2010; Diedrich 2011]. It applies Apollo guidance, which cannot enforce viewing constraints. Most likely, this is why the very high elevation and low downrange rate are chosen, as this combination will lead to a very steep descent with naturally favorable viewing conditions (but likely worse mass performance).

The plot for the initial positions, Figure 9.7(a), shows that – perhaps not all to surprisingly – most studies choose rather similar starting conditions, with NGC-1 being the clear outlier. As the physics governing all guidance laws is the same, the search space is defined around the existing data points, save for NGC-1. Trajectories with such a high elevation and steep initial flight-path angle are expected to be suboptimal in terms of propellant usage. Morevoer, the search space would become too big, while only a limited amount of computing time is available.

For these reasons, the search space is defined as follows. For the initial position, ±250 m is added to the extrema of the points. For the velocity, ±10 m/s is added to the extrema. An exception is the lowest possible descent rate, which is kept at 10 m/s as a lower limit. The resulting search space is indicated by the lines in Figure 9.7.

Finally, the ranges for the sensor angle and time-to-go need to be set. There exists no experience value for the former. A small test-run campaign with a hundred trials showed that angles from −10° to 30° were all feasible. For this reason, this range is chosen. The time-to-go is allowed to range between 50 s to 90 s, where the discretization step size in the algorithm is set to 1 s. This is based on engineering judgement and the time-of-flight of the reference scenarios, see Table A.1. The chosen ranges are summarized in Table 9.3. The vehicle and environmental parameters that are used in this study are listed in Table 9.4, which are based on the ESA LL.

9.2.2 Optimization

With the search space defined according to Table 9.3, the optimum must be identified. This is particularly challenging, because the permutations for optima are explosively huge.

To get a feeling for the size of the search space, a ballpark figure for the number of search points if a mere grid-search was performed can be derived (see Section 6.5.1).

TABLE 9.4 Common input parameters in this scenario analysis.

g_0, m/s	$\mu_{\mathbb{C}}$, m^3/s^2	$R_{\mathbb{C}}$, km	m_0 (PDI), kg
9.806 65	$4.902\,801\,076 \cdot 10^{12}$	1738	1737
m_0 (AG), kg	T_h, N	T_l, N	I_{sp}, N
987	3629	417	306.7

Let all parameters in Table 9.3 be divided into integers. Then the number of trials would be:

$$N_{r_x} N_{r_z} N_{v_x} N_{v_z} N_\gamma N_{tgo} = 1.224 \cdot 10^{13} \quad [9.10]$$

However, the search does not end here. To find the propellant-optimal solution, the PDI conditions have to be identified as well. For every approach-phase solution, the optimal main-braking conditions have to be determined as well. One might arguably choose larger intervals for the searches, but the number of possible combinations will remain huge either way.

Several options come to mind for exploring the search space:

Meta-heuristics Search methods such as genetic algorithms, particle-swarm optimization, or differential evolution have been successfully applied to highly complex and large search spaces [Spall 2003]. However, they remain heuristics that cannot guarantee to find the global optimum (as all other methods but convex guidance) and must be tuned well. They also provide little problem insight.

Design of experiments Specialized theory such as the Taguchi method can be applied to reduce the number of parameter combinations and therefore to reduce the number of trials significantly [Mooij 1998]. Moreover, through tools such as the response-surface methodology, insight into the problem can be gained by determining the interactions of parameters on the outcome. One drawback is that this is only easily applicable to problems with a clear structure, that has linear or perhaps quadratic interactions. It also works better for local rather than broad searches.

Monte Carlo The easiest, but least elegant way is to simply gamble. In an MC search, randomized trials of parameter combinations are evaluated in the hope that a near-optimal combination is found.

Even though meta heuristics, for an initial global search, and design of experiments, for a local search, sound appealing, it was chosen for this thesis to simply choose a Monte Carlo approach. This is has three main reasons: i) there was not enough time to implement and verify meta heuristics; ii) a local search using design of experiments would have cost much extra time for an unexperienced designer. The purpose of this study is merely proof-of-concept at this point, and is not aiming at a full mission analysis; and iii) a Monte Carlo search is easy to implement and will give sufficient results for an initial iteration.

For the initial search of optimal points in the search space the parameters have been randomly varied according to a uniform distribution. To this end, the function

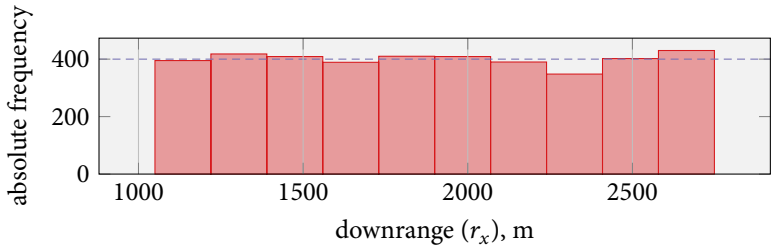


FIGURE 9.8 Basic verification of the `unidrnd()` function. Spread over ten bins, the frequency of each bin should be 400, which is approximately the case.

`unidrnd()` was used in MATLAB. 4000 iterations spread over two CPU cores were run on an INTEL I7 IVY BRIDGE processor, which took 16 hours to complete at a clock speed of 2.0 GHz. For reproducibility of the results, the random-number generator seeds were initialized with `rng(1);` and `rng(2);`. In total 201 feasible solutions were found, or 5 %. The analysis of these results is shown in the next section. The output of the random generator has been inspected visually for verification, of which a sample plot is shown in Figure 9.8.

9.2.3 Coarse search-results and pruning

This section presents the results from the first Monte Carlo search and a successive refinement iteration.

9.2.3.1 *Analysis*

The 201 solutions found in the first iteration are plotted in Figure 9.9. Although this is a busy graph, several important conclusions can be drawn from it straight away. Unfortunately, discussing all the interesting resulting trajectories in detail would be beyond the scope of this report.

First of all, it is very noticeable that trajectories are found nearly homogeneously over the entire search space. This indicates a certain robustness of the algorithm, because it can find trajectories *that satisfy the viewing requirements* for a wealth of different initial conditions.

The most propellant-optimal trajectory overall, counted from PDI up to TG, starts at the outer edge of the envelope at low elevation. The smooth shape and shallow angle of the trajectory cause it to minimize gravity losses. On the contrary, the most propellant-intensive trajectory starts at the other end of the envelope, and is almost vertical. This maximizes gravity losses. Moreover, its bumpy shape is very sub-par, and can be explained by a too long time-of-flight: in fact, the algorithm "wastes time" to satisfy the t_{go} and thus nearly hovers on the plateau for a while.

Not surprisingly, the most propellant-optimal trajectory for the approach phase only occurs for a gate close to the shortest distance from the landing site. Note that this trajectory only uses the least propellant *for the approach phase*, and not necessarily for the entire D/L from PDI. Accordingly, the least efficient trajectory is further away, and also starts at a higher speed.

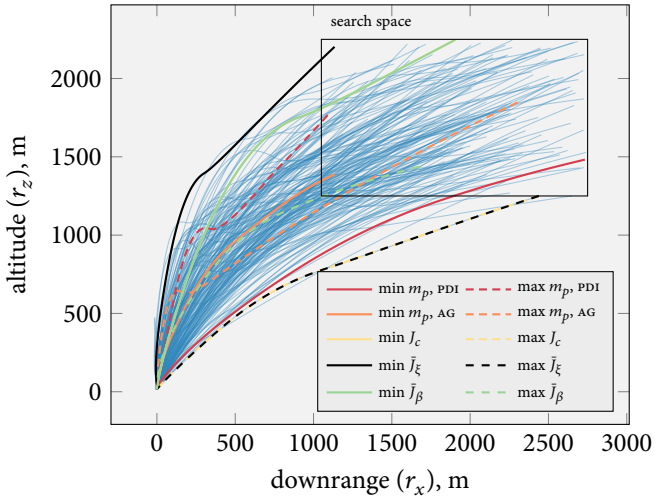


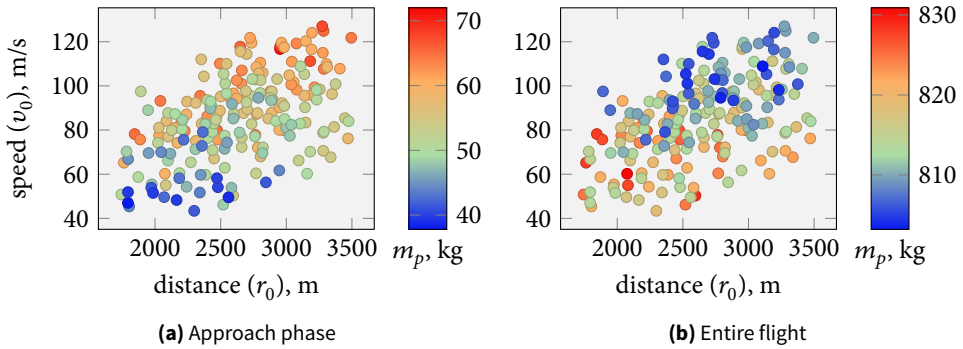
FIGURE 9.9 All feasible solutions found in the MC search. The trajectory for the best combined viewing cost is not visible because it is overlapping with the one for the minimal viewing-angle cost. The maximum cost and the maximum obliquity angle cost also overlap.

The smallest integrated obliquity-angle cost, $\bar{J}_\xi$, is found for an AG at the extreme point in the top-left corner of the search space. This comes at no surprise, because at this point the elevation ε is also maximal. This result confirms the hypothesis that a larger elevation causes a more optimal trajectory in terms of viewing conditions. To a certain extent, this also confirms the theory why the scenario NGC1 was designed at such a high initial elevation. The trajectory with the worst obliquity occurs at the point of the lowest elevation, as expected.

In terms of the viewing angle, $\bar{J}_\beta$, the best trajectory also occurs at high elevation. In this trajectory, the speed is initially very high, and the horizontal component gets cancelled first. The trajectory then steepens downwards and spends most of the time directly approaching the landing site, which explains the low cost. This trajectory is, in fact, the same as that for the combined cost, J_c. The worst viewing angle occurs at low elevation in a curved trajectory, which causes the landing site to drift in the fixed FOV of the sensors.

To summarize, the hypotheses for optimal trajectories seem to be confirmed, however, further verification of these claims is necessary. This will be done by studying the impact of the initial conditions on the cost functions.

First of all, as a sanity check the propellant cost of the approach phase only is shown as a function of the initial state in Figure 9.10(a). This figure clearly shows that the shorter the distance and the slower the speed, the less propellant will be needed. The main reason is that for these trajectories the time-of-flight is shorter, and thus the integrated propellant less. However, while this quantity is perhaps academically interesting, only the total usage from PDI to TG counts for the mission analysis. This is thus plotted in Figure 9.10(b). The behavior is now inversed: a faster initial speed and a start further away causes less propellant usage. This can be explained by the geometry of the search space: because the horizontal speed and distance range is larger, these figures contribute more to the total distance and speed. Therefore, larger numbers in



(a) Approach phase **(b)** Entire flight

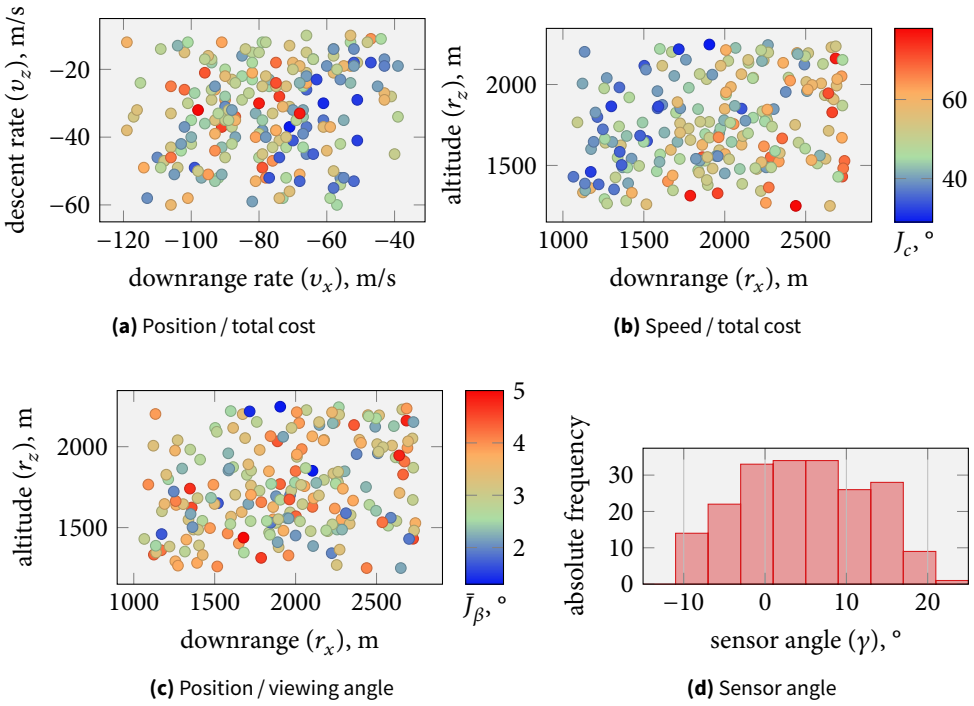
FIGURE 9.10 Propellant usage corresponding to the different gate initial-conditions.

Figure 9.10(b) generally mean shallower trajectories with less gravity losses, such as the best one shown in Figure 9.9.

To study the impact of the initial state on the vision-based performance cost, relevant graphs are shown in Figure 9.11. The graph for the initial position, Figure 9.11(a) shows that a slower downrange rate means better cost. This corresponds to steeper initial flight-path angles and thus a more direct approach to the landing site. The direct approach corresponds to a reduced obliquity angle throughout the flight. For faster downrange rates, the landing site will be approached more sideways, increasing the cost. An optimum seems to occur around 60 m/s, and no solutions for less than 39 m/s are found, although the search space was open till 30 m/s. Similar observations can be made in Figure 9.11(b) for the initial position. Gates with shorter downrange are favorable, because of the steeper initial elevation and thus a more direct descent straight from the top.

To verify this claim, only the viewing-angle component $\bar{J}_\beta$ of the total cost J_c is plotted in Figure 9.11(c). Firstly, this figure shows the positive result that trajectories with minimal average viewing angles can be found throughout the entire search space. This indicates robustness, and means that likely for any initial position good trajectories can be found viewing-angle wise (given a suitable sensor angle and velocity, of course). Secondly, a corollary from this observation is that the cost J_c is indeed driven by the obliquity. If a trajectory with optimal viewing conditions is sought, this thus means that one may solely focus on optimizing the conditions for obliquity, as it can be expected that optimal viewing-angle conditions can be found either way.

Unfortunately, one complication arises from the fact that optimal viewing-angles are feasible for all initial positions (and in fact also velocities). This is that no obvious relation for the sensor-installation angle is observable in the data. In fact, Figure 9.11(d) shows that feasible solutions are found for all possible angles γ. It would be desirable to determine a relation between the initial state and the viewing angle, so that the optimal angle can be chosen a-priori in an optimization and does not enter the solution space as an optimization variable. In fact, most of the 3800 rejected solutions were infeasible because of a wrong sensor angle. Having an analytical relation available would thus greatly decrease the search effort. It is therefore recommended to study this in the future.



(a) Position / total cost

(b) Speed / total cost

(c) Position / viewing angle

(d) Sensor angle

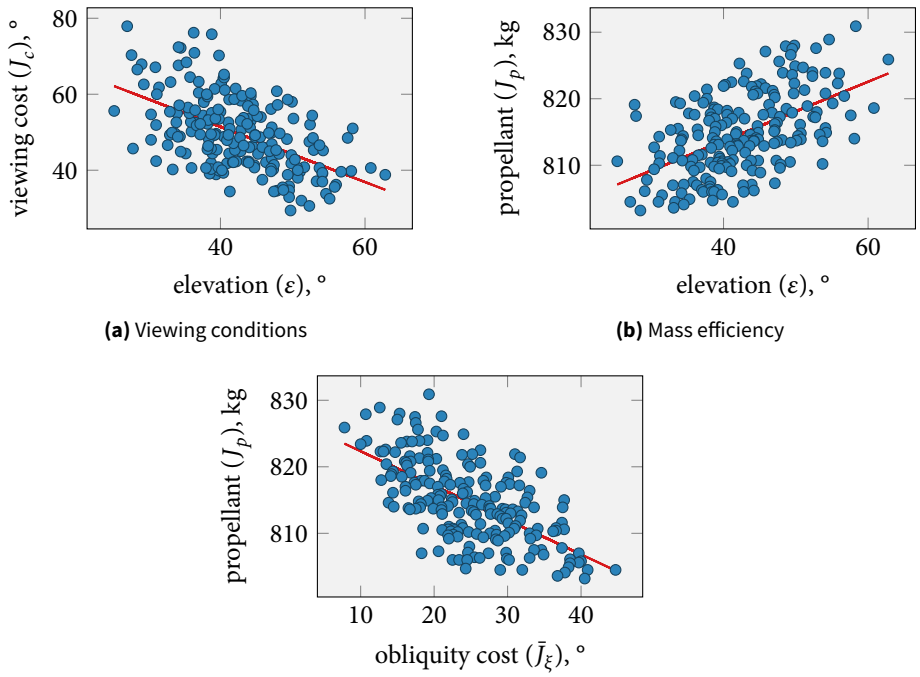
FIGURE 9.11 Impact of the initial conditions on the viewing costs.

To summarize the findings so far, a lower elevation angle and a shallow initial speed lead to the most propellant-optimal trajectories. This was demonstrated in Figure 9.10. On the other hand, trajectories with a steep elevation angle and velocity lead to better approach phases in terms of viewing conditions. This has been observed in Figure 9.11. There is thus a clear conflict of interests, as is specifically illustrated in Figure 9.12. Especially Figure 9.12(c) shows that improving the obliquity (meaning to minimize it) comes at the price of increased propellant usage. Even though the scatter in the plots is large, it is undeniable that there is an inverse between propellant-optimality and viewing conditions. Not entirely unexpected, this a fact of life that the designer of a mission scenario has to include in his trades.

The most important interactions between the performance metrics and the initial state have now been discussed. These results have been summarized in Table 9.5. Although many more plots showing interactions between parameters and their effect on the metrics could be presented, this table shall suffice for the discussion. One critical observation from this table is that, as already mentioned, the viewing angle and viewing-angle cost are completely unaffected by the other parameters. This does not exclude the possibility that there might be a nonlinear interaction, though, which has not been studied.

9.2.3.2 *Search-space refinement*

The previous section highlighted that optimal viewing conditions and optimal mass are mutually exclusive. The most mass effective trajectories are found for low elevations and near-horizontal velocities. However, the matter is slightly less clear for the



(a) Viewing conditions

(b) Mass efficiency

(c) Obliquity

FIGURE 9.12 Influence of the initial elevation of the approach phase on the two performance metrics.

TABLE 9.5 Interactions derived from the results in the search space.

	m_p, AG	J_p	$\bar{J}_\xi$	$\bar{J}_\beta$	$\bar{J}_c$
r_x	▲	▼	▲▲	•	▼▼
r_z	▲	▲	•	•	•
r	▲▲	•	▲	•	▼
v_x	▲	▼	▲▲	•	▲▲
v_z	▲	▲	▲	•	•
v	▲▲	•	▲	•	▲
t_{go}	▲▲	•	▼	•	▼
γ	•	•	•	•	•
ε	•	▲	▼▼	•	▼▼
fpa	•	▲	▼	•	▼

LEGEND:
• no observable interaction; ▲/▼ observable linear interaction; ▲▲/▼▼ strong linear interaction
▲ cost *increases* with *increasing* parameter; ▼ cost *decreases* with *increasing* parameter

TABLE 9.6 Statistical moments of the initial and refined cost distributions in the search space.

	$E[J_c]$, °	$\sigma[J_c]$, °	$E[J_p]$, kg	$\sigma[J_p]$, kg
Initial	49.65	9.64	814.7	6.0
Refined	42.17	8.54	818.3	5.7

TABLE 9.7 Refined search space for the AG.

	r_x, m	r_z, m	v_x, m/s	v_z, m/s	γ, °	t_{go}, s
MIN	1000	1500	40	10	–10	50
MAX	2000	2500	80	60	30	90

viewing conditions. Returning to Figure 9.12(a), it can be seen that the spread for J_c is significantly larger than for the propellant. This is expressed statistically in the first row in Table 9.6. The standard deviation of the viewing cost is as much as 20 % of its mean, whereas for the propellant mass this figure only reads 0.73 %. This means that when refining a search, it is easier to find more trajectories that are optimal in the sense of viewing-conditions and also perform well mass-wise than the opposite. For this reason, an additional search effort is done in the region that gives the best viewing conditions, in the hopes that relatively propellant-optimal trajectories can be found at the same time.

It was shown in Figure 9.11 that high elevations and steep flight-path angles benefit the total viewing cost J_c. The search is therefore also optimized around these parameters. Based on Figure 9.11(a), the refined downrange speed-range is chosen to be 40 m/s to 80 m/s, which contains the best solutions. The descent rate remains unaltered. Judging from Figure 9.11(b), the downrange is restricted to 1000 m to 2000 m, while the altitude is increased to 1500 m to 2500 m. This increase makes sense because the best solution is located at the top boundary of the envelope. The smaller search space should give more and better solutions in terms of viewing conditions and an – on average – worse propellant performance. All other parameters remain equal, of which an overview is given in Table 9.7.

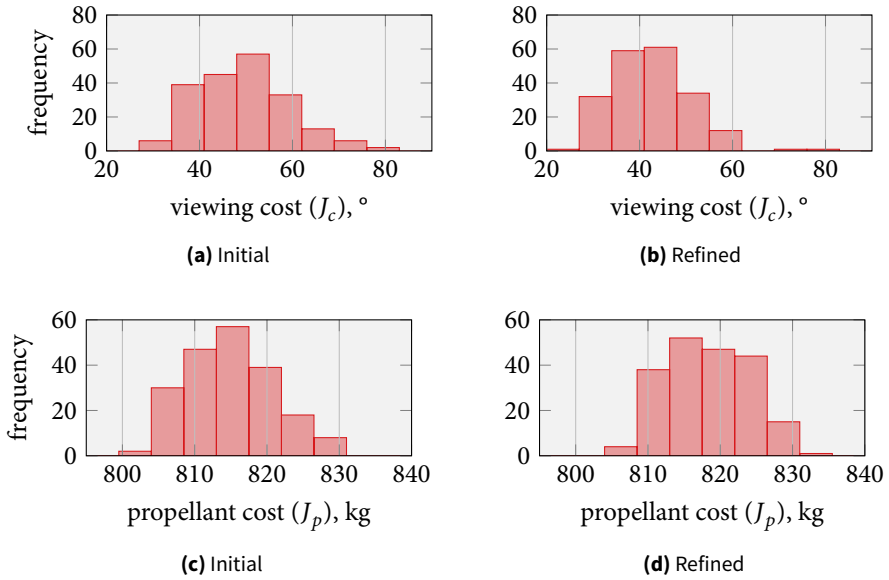
To find additional solutions, the same optimization as explained in Section 9.2.2 is executed, but now for the new variables. 2500 runs have been exectuxed, in which 201 – or 7.5 % – feasible solutions have been found. This is an increase of 50 % up from the previous run. As before, the load has been split over two cores and executed with the seeds `rng(1)` and `rng(2)`.

The results of the refinement are shown in Figure 9.13. Subfigures (a) and (b) confirm that the refined space does indeed produce better results than the initial range did. This is also confirmed mathematically in Table 9.6. Unfortunately, the propellant usage also increased as expected, which is shown in Figure 9.13 (c) and (d).

The next step is to merge the newly generated data with the previous runs and select a candidate from the overall pool. This is shown in the following section.

9.2.3.3 *Selection of an exemplary candidate*

To investigate the trade-space further and to select a candidate that will be put under closer scrutiny, the two performance criteria – viewing cost and propellant cost – can



(a) Initial **(b)** Refined

(c) Initial **(d)** Refined

FIGURE 9.13 Impact of the refined search space on the results for the viewing and propellant costs.

be plotted against each other. This is shown in Figure 9.14. This graph shows that there is a distinct family of solutions that are optimal. Each of these points is optimal in the sense that there can be no improvement in any of the cost functions without sacrificing performance in the other one. In other words, there exists no "best-of-both-worlds solution." These points are called "Pareto-optimal" and form the "Pareto front" [Spall 2003]. The end points of this front are those values with the lowest viewing cost and the cheapest propellant metric, respectively.

Having two cost functions makes a trade-off hard, because there is no single optimal value. The designer needs to select an optimal point on the Pareto front in Figure 9.14 based on another rationale. This could be, for example, the need to get the most payload for a mission as possible. If a mission was to demonstrate optimal HDA scenarios, then the designer might rather choose a point with a lower viewing cost.

Unfortunately, this complication makes it very hard to select a point for further study, because any further rationale is beyond the scope of this thesis and would enter the realms of mission design more than necessary. All Pareto-optimal points are documented in Table 9.8. To proceed either way, Point 7 in Figure 9.14 has been selected for further study for the following reasons:

- The point represents an "average" or mid-point solution, as it is located between the extreme end points. Any analysis therefore has some more general value than studying extrema.
- The viewing cost is 51 % worse than the optimum, but compared to the worst solution (157 %) this is still good (Table 9.8). This is also better than the average cost.

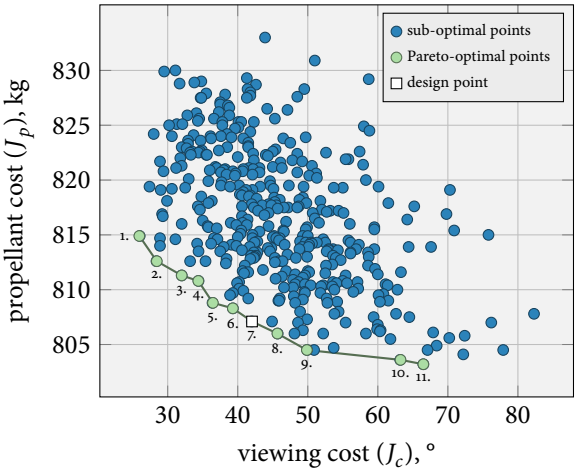


FIGURE 9.14 Pareto front of the two cost criteria for viewing conditions and propellant usage, respectively.

TABLE 9.8 Parameters and cost values of the eleven Pareto-optimal solutions. The last two columns indicate the performance with respect to the corresponding best solution.

#	r_x, m	r_z, m	v_x, m/s	v_z, m/s	t_{go} (AG), s	γ, °	J_c, °	J_p, kg	% min J_c	% min J_p
1	2732	1482	−106	−25	54	4	66.5	803.2	257	100.00
2	2334	1524	−90	−29	50	5	63.2	803.6	244	100.05
3	2108	1386	−101	−30	54	8	49.8	804.4	192	100.15
4	1836	1470	−103	−44	55	13	45.6	806.0	176	100.35
5	1708	1646	−91	−51	56	6	42.0	807.1	162	100.49
6	1845	2133	−76	−48	54	9	39.3	808.3	151	100.63
7	1240	1650	−67	−55	50	4	36.4	808.7	140	100.65
8	2104	1851	−99	−49	67	6	34.3	810.7	132	100.93
9	1206	1581	−66	−48	56	6	32.0	811.3	123	101.01
10	1225	1820	−66	−58	57	8	28.4	812.5	110	101.16
11	1395	2016	−69	−59	63	6	25.9	814.8	100	101.44

- It is hard to argue about the propellant cost, as each kilogram counts. However, at only 0.65 % more usage than the optimal solution, this cost is considered acceptable *for the sake of this analysis.*[5]
- The control solution has been initially studied and is feasible.

The trajectories corresponding to the solutions presented in Table 9.8 are plotted in Figure 9.15. Yet again, the trend from steep to shallow trajectories for more optimal viewing conditions and propellant usage, respectively, is clearly observable. In fact, some of the trajectories previously presented in Figure 9.9 on Page 179 reappear in this chart, notably the most propellant-optimal one.

Having chosen a design, a more local analysis around this point is in order.

5 If related to the payload mass of 6 kg, an increase of 5.5 kg w. r. t. the propellant-optimal solution is nearly 100 %. It is therefore stressed again that it is up to the systems engineer to do the appropriate trades in consultation with the GNC designer.

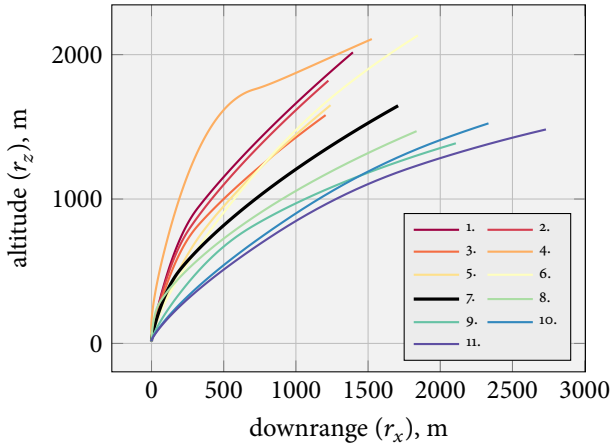


FIGURE 9.15 Trajectories corresponding to the Pareto-optimal points in Figure 9.14

9.2.4 Sensitivity analysis

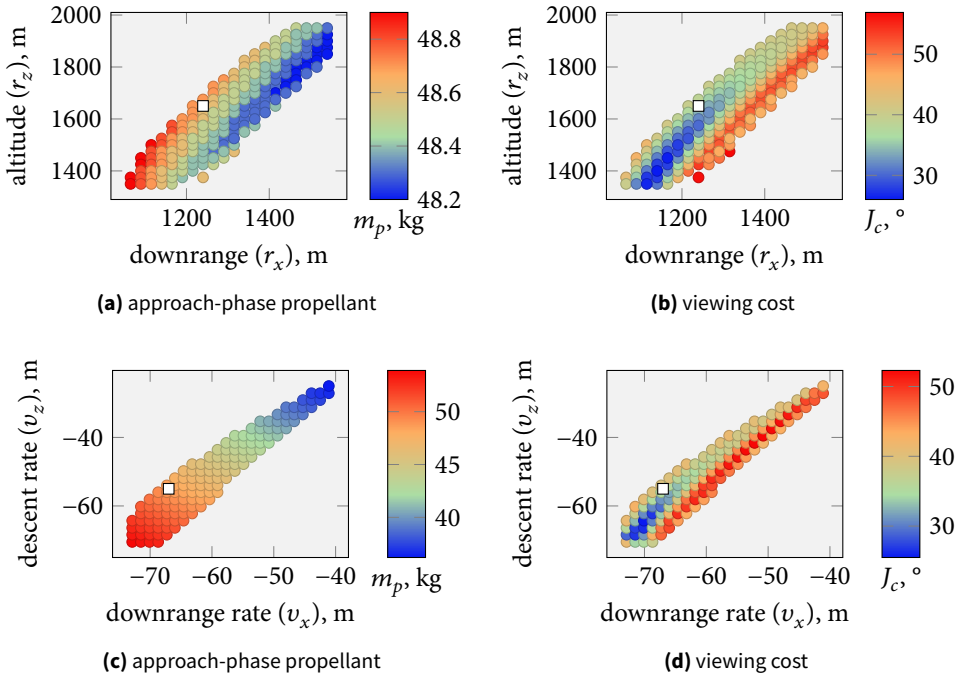
To analyze the trajectory resulting from the chosen parameters in more detail, an initial sensitivity analysis is performed. This helps in getting deeper insight into the design and allows for slightly re-designing and improving it. A second, more localized sensitivity-analysis of higher resolution is presented for the updated design point in Section 9.2.5.

Accordingly, all influential guidance and vehicle parameters will be varied over a relevant range to study the impact on the solution. The parameters and their ranges used in this analysis are shown in Table 9.9 and will each be discussed in detail in the following. The ranges have been experimentally chosen, either to span the entire feasible envelope (*e. g.*, for the sensor angle), or to span a physically sensible range (*e. g.*, for the I_{sp}). To illustrate this, consider Table 9.9. The sensor angle is varied over a range of −6° to 14°, which covers the entire feasible envelope of 2° 11°. For some other parameters, such as the position or the velocity, the range is varied over a large span (> 500 m), which still does not cover the entire feasible range. In such cases, the range is limited to a sensible value w. r. t. the design point. For example, the real dispersions in the initial position at AG are $\ll$ 500 m, so little practical insight would be gained from going far above this value.

The most important variables are the downrange, altitude, and corresponding velocity components. The change in the propellant usage during the approach phase and the viewing cost is shown in Figure 9.16, where all points have been computed with $\gamma = 2°$. The design point is indicated by the white square. Figure 9.16 shows that the initial position can be varied over a very broad range. In fact, feasible solutions could be found over the entire range specified in Table 9.9. As a matter of fact, the propellant mass is nearly insensitive to the initial position, seeFigure 9.16(a). This is not too surprising, because the propellant usage is dominated by the time-to-go, which is identical for all these solutions. On the other hand, Figure 9.16(b) shows that the viewing cost is strongly sensitive to the initial position. The closer the initial position is to the landing site, the better the viewing conditions. The design point is located near the optimal region, and on the edge of the feasible envelope.

TABLE 9.9 Parameter-study ranges in the sensitivity analysis and the resulting feasible envelopes. The last two columns indicate the initially chosen design point and a proposed redesign for improving robustness.

	VARIED RANGE		FEASIBLE RANGE			
	LOWER	UPPER	LOWER	UPPER	DESIGN	REDESIGNED
r_x, m	940	1540	1065	1540	1240	1265
r_z, m	1350	1950	1350	1950	1650	1625
v_x, m/s	−117	−17	−72	−41	−67	−67
v_z, m/s	−135	−35	−71	−25	−55	−60
t_{go}, s	40	90	48	90	50	52
γ, °	−6	14	2	11	4	2
m_0, kg	777	1177	777	1084	977	—
I_{sp}, s	107	507	107	507	307	—
T_l, N	437	937	437	937	437	—
T_u, N	3129	3629	3296	3629	3629	—



(a) approach-phase propellant

(b) viewing cost

(c) approach-phase propellant

(d) viewing cost

FIGURE 9.16 Sensitivity of the propellant mass and viewing cost to the initial kinematic state.

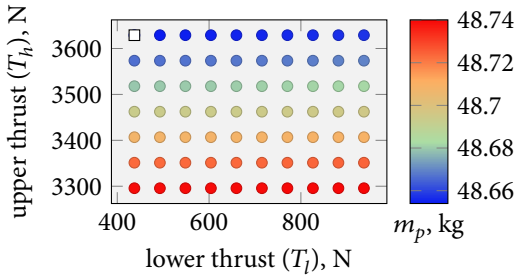


FIGURE 9.17 Feasible envelope of the thrust limits. The used propellant mass is nearly insensitive to the constraints.

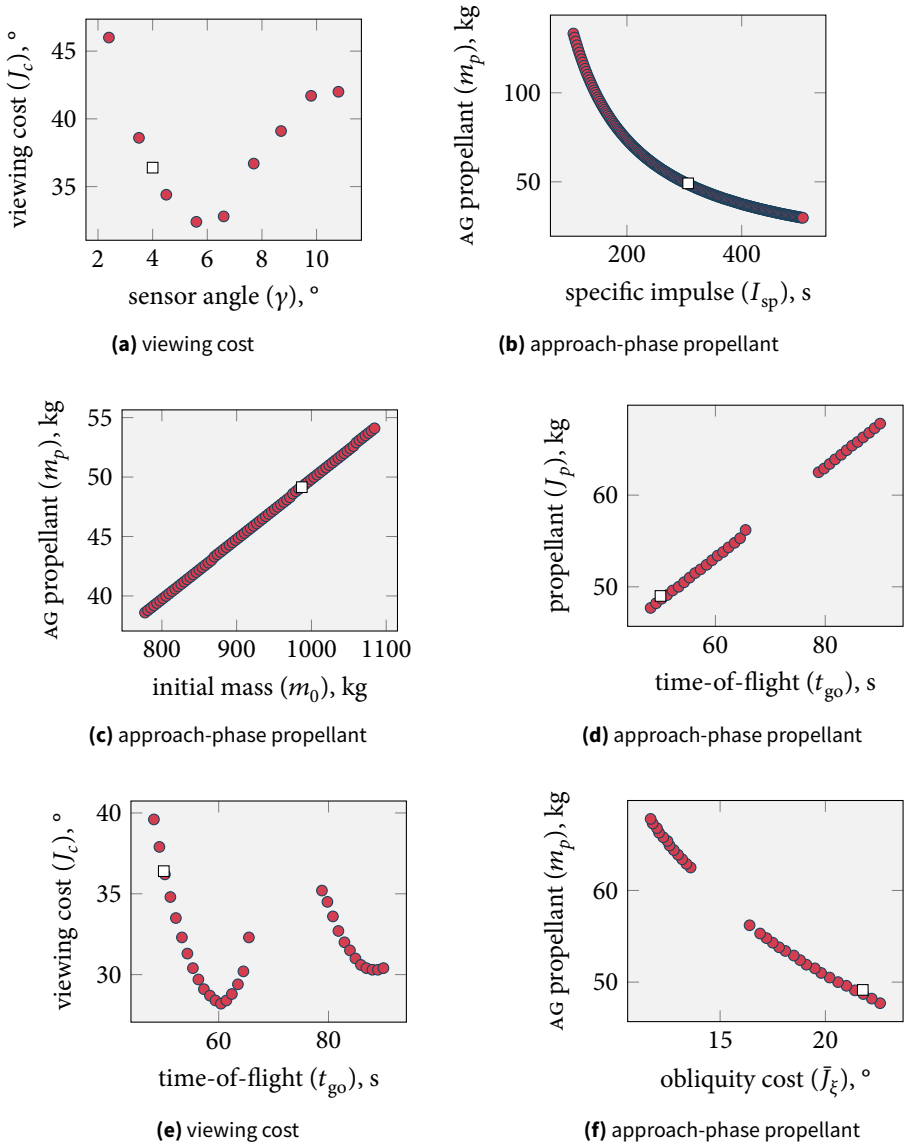
As shown in Figure 9.16(c), the initial velocity influences the propellant usage much more. A faster speed means more propellant. However, it must be stressed that this represents only the *approach-phase* propellant, not the overall consumption since PDI. The opposite is true for the viewing conditions, which is visible in Figure 9.16(d). Here, a faster initial speed means improved viewing conditions. Overall, Figure 9.16 shows that the initial conditions lie in areas that are neither the best for the viewing conditions nor for the propellant—which would be expected from a Pareto-optimal solution.

The needs of the propulsion system are documented in Figure 9.17. Clearly, the total propellant mass is hardly influenced by the applied thrust. As expected, though, the higher the thrust, the less propellant is required. As shown in Table 9.9, the trajectory is feasible for the full range of lower thrust bounds down to 437 N. The thrust may not get lower than 3296 N, or else the trajectory will become infeasible for this time-to-go. Note that the design point is in the top-left corner, because the thrust limits were not varied above or below the physical limitations of the propulsion system.

The graphs for the remaining important parameters are shown in Figure 9.18. First, the impact of the sensor angle γ is shown in Figure 9.18(a). Not surprisingly, the effect of the angle is very strong, as the amplitude between the best and worst cost is 13°. The design point is chosen originally at $\gamma = 4°$. A redesign is easily possible, because γ is just a vehicle parameter. A reduction of the cost by 5° should be possible "for free". However, this has to be done *after* the initial state has been changed. This is because the trajectory is so sensitive to γ.

The specific impulse I_{sp} and the initial mass m_0 bear no surprises when it comes to their impact on the propellant use, see Figure 9.18(b) and (c). The former acts exponentially on the mass, while the latter exhibits a linear relation.

The time-to-go demonstrates some more interesting behavior, see Figure 9.18(d). As a matter of fact, two "families" of solutions are found. For the first family, the lower feasibility limit is at 48 s and the upper at 68 s (Table 9.9). The design point at 50 s is on the lower end, which tends toward the propellant-optimal point. The second family starts at 79 s and extends beyond the investigated range. The nature of these families becomes clear when looking at the trajectories in Figure 9.19 and the viewing cost in Figure 9.18(e). There are two types of trajectories: direct approaches (first family) and "bumpy rides" that cancel all speed first, then rise again, just to descent vertically to the landing site (second family). There are no intermediate trajectories because these do not meet the viewing-angle constraint. The bump for the second family is caused



(a) viewing cost

(b) approach-phase propellant

(c) approach-phase propellant

(d) approach-phase propellant

(e) viewing cost

(f) approach-phase propellant

FIGURE 9.18 Sensitivity of the propellant mass and viewing cost to vehicle parameters and time-to-go.

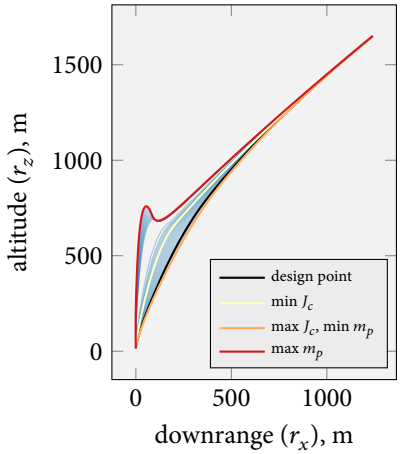


FIGURE 9.19 Trajectories corresponding to the range of time-to-gos investigated in Figure 9.18(e). There are two families of feasible flights.

because a long t_{go} is forced upon the algorithm. This causes it to "waste time" by first rising again.

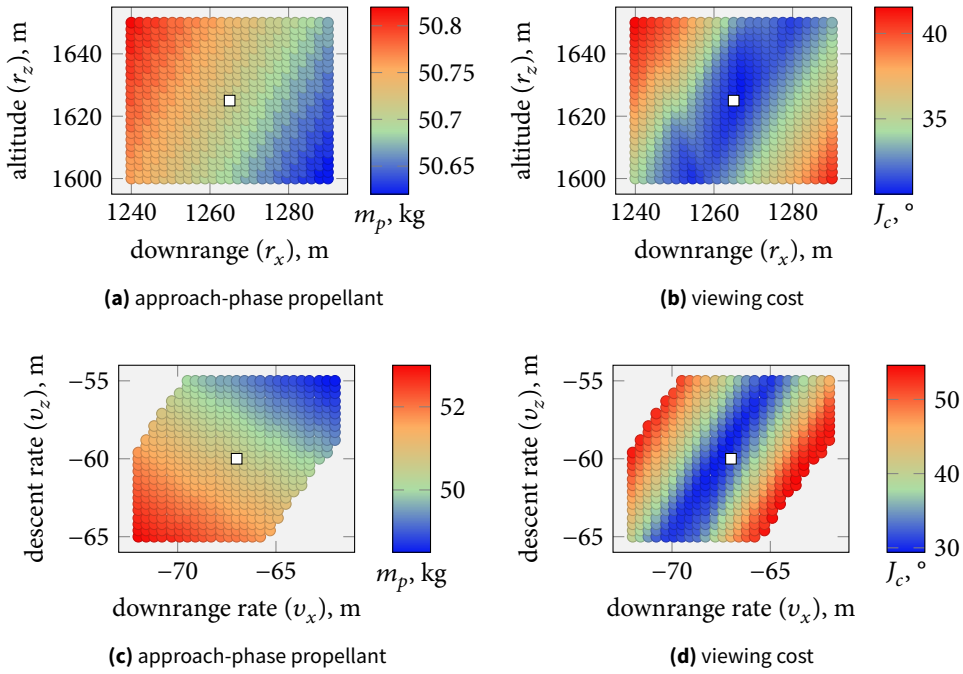
To summarize, the sensitivity analysis gives a first indication for the robustness of the algorithm, as initial conditions can be widely varied. Further confirmation of this will need to be given by an MC analysis. Moreover, the chosen design point tends to be near the boundary of the feasible envelope and typically *near*, but not directly *in* feasible regions. Possible minor adjustments are shown in the next section, which finalizes the design.

9.2.5 Final design

One important finding of the sensitivity analysis is that the design point typically lies near the boundary of the feasible envelope. While this leads to the Pareto-optimality, it also poses a certain risk. Consider Figure 9.16 again. The initial position is located at $r_x = 1240\,\text{m}$ and $r_z = 1650\,\text{m}$. At this point, the maximum allowable change in x- and z-direction is 25 m, or else the solution will become infeasible. Accordingly, the delivery error at AG may *never* exceed this value.

This strict requirement can be strongly relieved if only slight adjustments are made to the design. At the same time, the viewing cost can be further improved, while sacrificing only a minimal amount of propellant. Unfortunately, such a re-design is all but straightforward. It was found that all parameters strongly interact with each other, so that even a small change calls for the reiteration of an analysis cycle.

To give an example, changing the initial position to a more optimal point might embed the initial position deeper in the feasible envelope. However, this change will also affect the envelope of the velocity, where the initial point may now have moved toward the feasible boundary. Moreover, a new sensor angle could now be optimal. But if this is changed, the envelopes for the position and speed will change again. The bottom line is that tweaking the initial positions is a *highly iterative process*, that requires plentiful re-evaluations of the sensitivities. Sparing pages of plots, only the final results of these iterations are shown here.



(a) approach-phase propellant

(b) viewing cost

(c) approach-phase propellant

(d) viewing cost

FIGURE 9.20 Feasibility envelope around the finalized AG.

The following modification to the design point is proposed (see Table 9.9): $r_x = 1265$ m, $r_z = 1625$ m. This corresponds to moving slightly right and downwards in the design space, "embedding" the point deeper in the feasible space. The tolerable error at AG is now 50 m, which is an improvement by 100 %. Figure 9.20(a) and (b) show the sensitivities around the new design point, clearly showing that it is now in the center.

For much the same reasons, the speed is proposed to be moved to $v_x = -67$ m/s and $v_z = -60$ m/s (see Table 9.9). Previously, even a small speed error would have lead to infeasible solutions (see Figure 9.16(a) and (b)).[6] Now about 5 m/s should be tolerable at AG, as shown in Figure 9.20(c) and (d).

Finally, the sensor angle γ is changed to 2° for reasons explained in the previous section. This is a "free boost" to the viewing cost, which is now decreased by 15.4 % from $J_c = 36.4°$ to 30.8° (Figure 9.20(b), (d)). The time-to-go is increased very slightly to $t_{go} = 52$ s to move away from the edge of feasibility. Although this should seemingly come at a cost of $\Delta m_p = 1.4$ kg according to Figure 9.18(d), the total increase for main braking *and* approach phase is only 0.07 % from $J_p = 808.7$ kg to 809.3 kg (10 % w. r. t. payload mass). This is a small price to pay for a huge increase in robustness and viewing optimality. However, it is stressed that any such trade would have to be justified to a systems engineer in an actual mission study.

All other values are considered fixed and cannot be changed. For example, the initial mass and specific impulse are no design parameters when considering guidance, so no final adjustments are made to these. Such parameters would be traded on the system level.

6 Up to the precision of the grid sampling.

TABLE 9.10 Summary of the final AG design.

r_x, m	r_z, m	v_x, m	v_z, m	t_{go}, s
1265	1625	−67	−60	52
m_p (AG), kg	m_p (PDI), kg	J_p, kg	γ, °	J_c, °
50.7	758.6	809.3	2	30.8

The design of the AG conditions is now concluded and the results are summarized in Table 9.10. The following section will detail the design of the PDI conditions.

9.3 POWERED-DESCENT INITIATION DESIGN

The design of the PDI conditions follows a similar approach as the AG conditions, but is much more straight forward. In the next three sections, the design variables will be chosen first, the design space based on these will be explored, and finally a design point will be chosen.

9.3.1 Design variables

The design of the PDI initial conditions presents a much lesser challenge than finding the right AG. The primary reason for this is that far fewer parameters have to be considered and the optimization search-space is much smaller. The following parameters influence PDI:

Downrange The downrange is the most important trade parameter. It simply defines how far away from the landing site the D/L sequence is commenced, measured along the curvature of the Moon's surface.

Altitude Strictly speaking, the altitude should also be a trade-parameter. Delaune et al. [2010] did a sensitivity analysis to study the impact of the altitude on the required propellant. They found that per 5 km of altitude a penalty of 1 kg is to be expected. It would thus be ideal to fly just above the surface of the Moon. Of course, this is not feasible due to the terrain. The ESA LL baseline is 10 km for safety reasons, which is therefore also taken as reference value for this study [Diedrich 2011].[7]

Crossrange The crossrange is ignored because it could always be eliminated through a simple frame transformation.

Speed The initial speed is derived from orbital mechanics. The velocity vector is the local orbital velocity of the DOI 100 km × 10 km Hohmann-ellipse, which is considered a given mission constraint that cannot be changed.

Time-of-flight The t_{go} is a defining design criterion.

7 Note that in the days of Apollo, the perilune altitude of the transfer orbit was taken to be 15 km [Bennett 1970]. However, this was before the days of high-definition DEMs and a larger safety factor was thus considered.

TABLE 9.11 Search-space definition and resulting design for the PDI.

	VARIED RANGE		FEASIBLE RANGE				
	LOWER	UPPER	LOWER	UPPER	STEP	DESIGN	REDESIGNED
d, km	560	700	585	700	5	640	630
t_{go}, s	600	700	630	700	5	660	660

AG conditions The final BCS are also tradeable. However, they have already been defined in the previous section and are thus considered to be fixed.

In conclusion, only the downrange d and the time-of-flight t_{go} will be optimized to find the best PDI conditions. This will be done in the following section.

9.3.2 Design-space exploration

Before the search space can be sampled, the trade criterion has to be defined first. This is merely the propellant use for the main-braking phase, and viewing conditions are ignored for now. Even though absolute vision-based navigation is applied during this leg, the requirements are much less strict. The camera simply needs to stare down at the surface and match features. There is no requirement to point at a particular point such as the landing site. A viewing-angle constraint is therefore unnecessary.

To investigate the topography of the design space in terms of m_p, the downrange d and time-of-flight t_{go} are varied over the ranges specified in Table 9.11. The search space is small enough for such a simple grid search. The maximum number of iterations for the main-braking algorithm is set to ten and the convergence criterion is $\|\boldsymbol{X} - \boldsymbol{X}^{-}\| \leq 0.1$. The discretization time step is 5 s.

The results of the search are shown in Figure 9.21. As a pleasant surprise, the search space is not bounded at the ceiling of the parameters. Moreover, the optimal region is of very broad extent and flat, meaning that there is no distinct optimal point, but rather a range of points with nearly equal cost. A brief analysis reveals the (rather obvious) reason for this. For longer times-of-flight there are longer coasting arcs at minimal thrust.

9.3.3 Final design

The large optimal region gives the designer lots of leeway. As stated before, a system-level trade-off for the PDI conditions are beyond the scope of this thesis. In the coarse PDI sampling that has been performed in the AG analysis, the optimal design point had been identified as $t_{go} = 660$ s, $d = 640$ km (Table 9.11). Inspecting Figure 9.21 this point is not directly on the boundary of the feasible enevelope. Even a maximum error of 31 km would still be feasible. Would the delivery error be 10 km at PDI, the increase in propellant mass would be 6.2 kg. This is approximately the maximum expected error, corresponding to the 3σ value used in the MC analysis (Section 9.4.5).

Anticipating possible errors at the PDI delivery, an option would be to choose a nominal point that is slightly less optimal, but for which the increase in propellant mass would be significantly less if there are errors at PDI. If the nominal point would be at

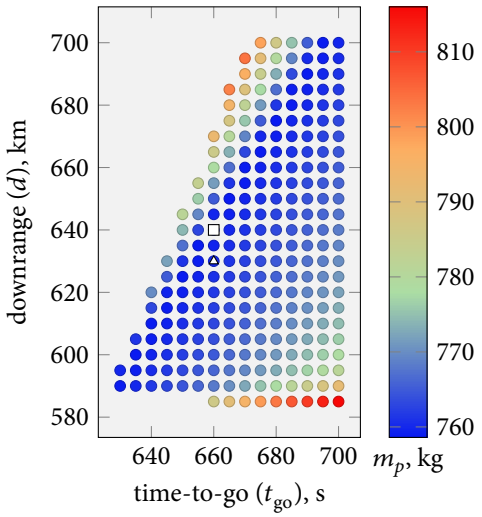


FIGURE 9.21 Feasible solutions on the search interval for the PDI. The square indicates the initial design point, the triangle the final design.

$t_{go} = 660$ s and $d = 630$ km, corresponding to a 10 km shorter fligt, the needed propellant would be 760 kg, up by 1.2 kg. However, if there was a 10 km error, the worst-case propellant need is now only 761.4 kg, which is 3.5 kg less than for the previous point. If MC analyses were run for both design points, it should be expected that the mean propellant mass would be significantly less for the second option. Moreover, this point should be even more robust, because the maximum tolerable error has now increased as well. For these reasons, the nominal PDI conditions are set to these values, see Table 9.11.

Even though a sensitivity analysis could be performed at this point, it would add little value. First of all, Figure 9.21 in fact already presented the sensitivities of the propellant mass to the time-to-go and the downrange. These are the most important parameters. Although cross-range and altitude could now be studied as well, they would not influence the design point either way. Some quick simulations have also shown that their impact on the trajectory is minimal. This makes sense, considering that any small crossrange error at PDI can be corrected over the course of 660 s. Just a small impulse at the start in the opposite crossrange direction can eliminate any such error. The sensitivities to mass, thrust, *etc.*, are the of the same physical nature as presented before. Overall, the main-braking phase is a much simpler and less interesting procedure than the approach phase. Rather than showing plots for all these parameters, robustness is therefore demonstrated by an MC analysis, which is the subject of Section 9.4.5. This concludes the design of the PDI conditions.

9.4 ANALYSIS OF THE CHOSEN DESIGN POINT

Having defined the mission scenario through the design of the PDI and AG conditions, it is the purpose of this section to analyze the resulting guidance commands and trajectory in detail. Instead of the previously performed discrete-time analysis, this is now

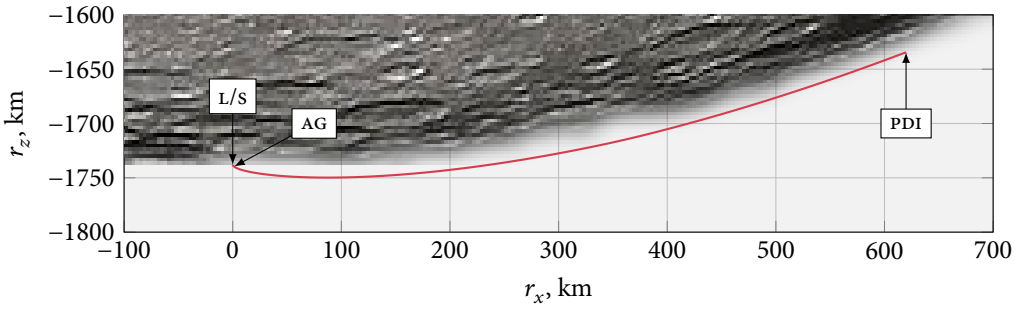


FIGURE 9.22 Nominal trajectory around the Moon from start to finish. The main-braking phase makes up the major portion of the flight.

TABLE 9.12 Event timeline of the designed scenario.

EVENT	t, s	h, km	d, km	Δv, m/s	m_p, kg
PDI	0	10.0	630.0	0	0
Throttle down	165	14.0	381.4	367	199
Throttle up	200	14.9	335.1	376	204
AG	660	1.6	1.3	1731	760
Throttle down	686	0.5	0.2	1829	792
Throttle up	697	0.2	0.1	1835	793
Relax FOV centering	702	0.2	0.1	1852	799
Relax camera FOV	708	0.1	0.0	1876	806
TG	712	0.0	0.0	1891	811

by integrating the commands in continuous time. To verify the expected performance an MC analysis is presented.

9.4.1 Outline of the scenario

The flight around the Moon starting at PDI is shown to scale in Figure 9.22. It is clear that the main-braking phase makes up by far the longest part of the flight, which is also shown in Table 9.12. In 660 s roughly 628.7 km is covered. The approach phase covers only 1.3 km downrange but takes another 52 s due to its slower speed. While this may seem obvious, it is impressive to see how short the approach phase is in the grand scheme of the mission. Even though this is the most critical phase in terms of mission success, propellant-wise the main-braking phase is far more important. It uses 93.7 % of the totally required propellant (Table 9.12). This is an important realization, that will be further discussed in the following section.

9.4.2 Nominal main braking

Having presented the main-braking trajectory in Figure 9.22 already, some additional plots are required to gain insight into how this flight profile comes to be and how it looks like in detail. To this end, the most critical guidance-commands, the pitch and

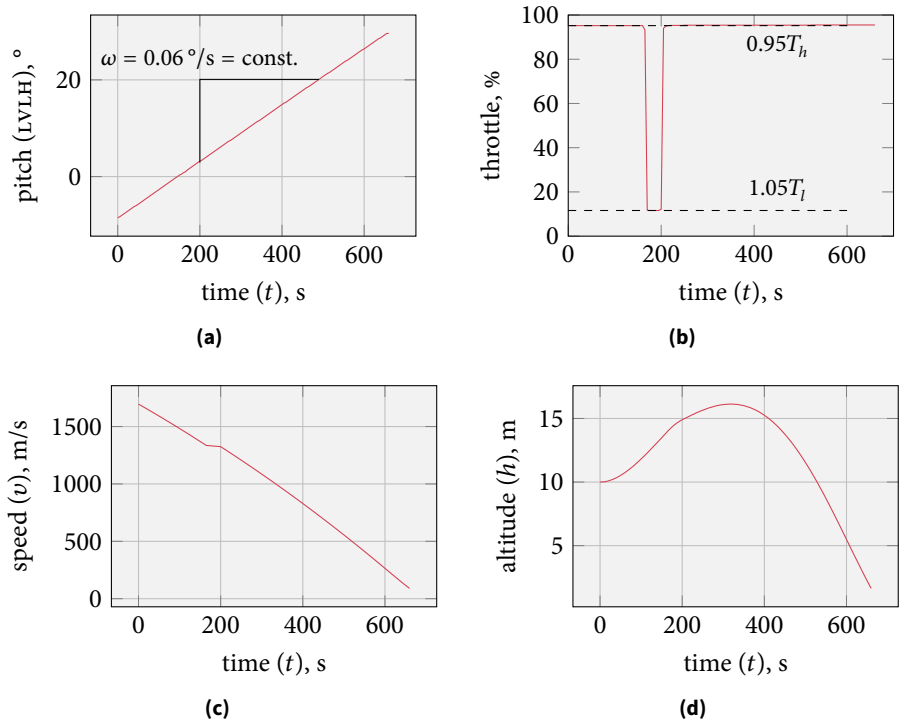


FIGURE 9.23 Guidance commands and resulting trajectory for the main-braking phase.

throttle level of the engines, are shown in Figure 9.23(a) and (b). The speed and altitude throughout this phase are shown in Figure 9.23(c) and (d) .

One peculiar observation from Figure 9.23(d) is that the lander actually first *rises* during the breaking burn before it descents again to its final altitude. This might seem unintuitive at first, because one could expect that the spacecraft would steadily decrease its altitude starting from PDI.

The most straightforward explanation would be that the initial speed at PDI is the perilune speed of an elliptical orbit, so that it is to be expected that the orbit first rises again. To show that this is indeed *not* the case, the speed has been plotted against the altitude in Figure 9.24. The line at the top of the graph indicates the local circular velocity. It is immediately clear that only a tiny fraction of the flight occurs at speeds faster than orbital speed. This explanation is therefore invalid.

Nevertheless, Figure 9.24 provides further important insight. The explanation for the behavior is that there is not enough thrust to break the spacecraft in such a way that it can descend right from the beginning. The lander needs to rise first such that it does not crash during the main-braking phase. If the T/W was higher, then it would rise by much less, or even descent directly. In the extreme case, if $T/W \to \infty$, there would be an impulsive shot. This theory is confirmed by plots shown in [Diedrich 2011], which have been prepared using a higher T/W.[8] Indeed, here the spacecraft only rises by about 1 km.

8 Unfortunately this author does not hold the permission to reproduce these figures.

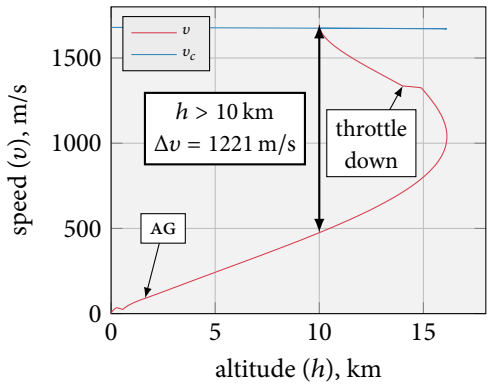


FIGURE 9.24 Reduction of the speed during the entire flight. The lander initially rises for a more efficient braking burn and eliminates 72 % of the speed above its initial altitude of 10 km.

Another, more exotic explanation would be that braking at higher altitudes is slightly more efficient. This explains why 72 % of the speed is cancelled above the initial altitude of 10 km. Intuitively, this makes sense if one realized that maneuvers to change orbits, such as perigee raising, is also more efficient higher apogees. Moreover, gravity losses are decreased at higher altitudes where the gravitational acceleration is less (0.7 % at 16 km wrt. 10 km). The speed is also decreased the fastest at the highest altitude. However, this theory is less likely to play a major role and would need to be further verified.

As a matter of fact, this rise in altitude is expected from purely theoretical derivations as well, although they do not give much physical insight. The linear-tangent steering law presented in [Perkins 1966] states that the optimal braking burn in a spherical gravity-field of constant magnitude is executed at maximum thrust and a constantly changing pitch rate. In simplified notation, this guidance law reads:

$$\tan\theta = a + bt \qquad [9.11]$$

where θ is the pitch angle, a and b are constants, and t is the time. This law only commands a change in the pitch angle because it assumes a constant, maximum, planar thrust. This precise theoretical optimum is in fact also the output of guidance in this scenario, as shown in Figure 9.23(a) and (b). The attitude changes at a nearly constant rate of $\omega = 0.06\,°/s$. One important difference between convex guidance and the linear-tangent steering law is the presence of the coasting leg of 35 s (Table 9.12), which occurs at the lowest possible thrust setting (Figure 9.23(b)). This leg is present because the final state can be controlled with convex guidance, which is *not* entirely possible with linear-tangent steering. The latter guidance law *only* commands an attitude for optimally eliminating speed, but cannot properly constrain the final state.

The purpose of the coasting leg and the reason why it occurs at this particular epoch is revealed when returning to Figure 9.24. In the first thrust leg some speed is already eliminated. However, a vertical component is also added to the speed to raise the trajectory. Before reaching the peak, the engines are throttled down to let the spacecraft coast to the maximum altitude. Near this maximum, the engines are engaged again to slow down at maximum efficiency.

These explanations are further backed up when investigating the speed profile in Figure 9.23(c). In the beginning, the slope of the curve is slightly less than after the

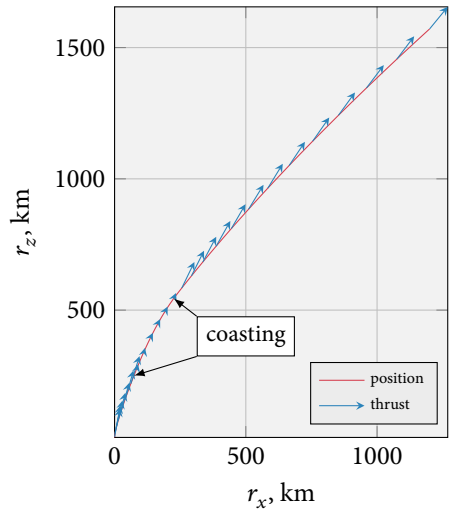


FIGURE 9.25 Approach-phase trajectory resulting from the guidance command. The realized thrust-vector is shown in ticks of 2 s.

coasting leg. This is because a small component is "wasted" to raise the altitude. After that the speed decreases at an accelerating rate, which is due to the lander getting lighter by burning propellant.

In essence, the coasting arc leaves the algorithm freedom in realizing the trajectory. If there was no period of throttling down, then there would also be no margin for error, as any small deviation from the reference would now mean that the final state cannot be realized.

In conclusion, the main-braking guidance command confirms the expectations and yields an optimal trajectory. The approach phase is treated next.

9.4.3 Nominal approach

Even though the approach-phase is the most critical landing phase, it bears little surprise when it comes to the resulting trajectory. The nominal flight and the guidance command are shown in Figure 9.25. To the greatest extent, the rationale for this path has already been given in Section 9.2. However, this section shall shed some more light on the detailed commands and performance.

As such, the most important variables are plotted in Figure 9.26. Again, the attitude changes at a constant rate, indicating optimality (see (d)). The throttle profile accommodates a brief coasting leg, which can be seen in (e) and Figure 9.25. The nature of this command can be deduced from (b). In the first coasting leg, a great deal of horizontal velocity is eliminated, which serves the purpose of targeting the landing site. In the second leg, the lander now coasts slowly toward the landing site. Finally, the remaining vertical speed is eliminated.

The viewing conditions during the approach are superb, which is shown in Figure 9.26(e) and (f). During no part of the flight does the viewing angle exceed 8°. Given that the FOVS for the LIDAR and camera are 40° and 70°, respectively, this means that the landing site is permanently visible by the sensors, even below 50 m where this re-

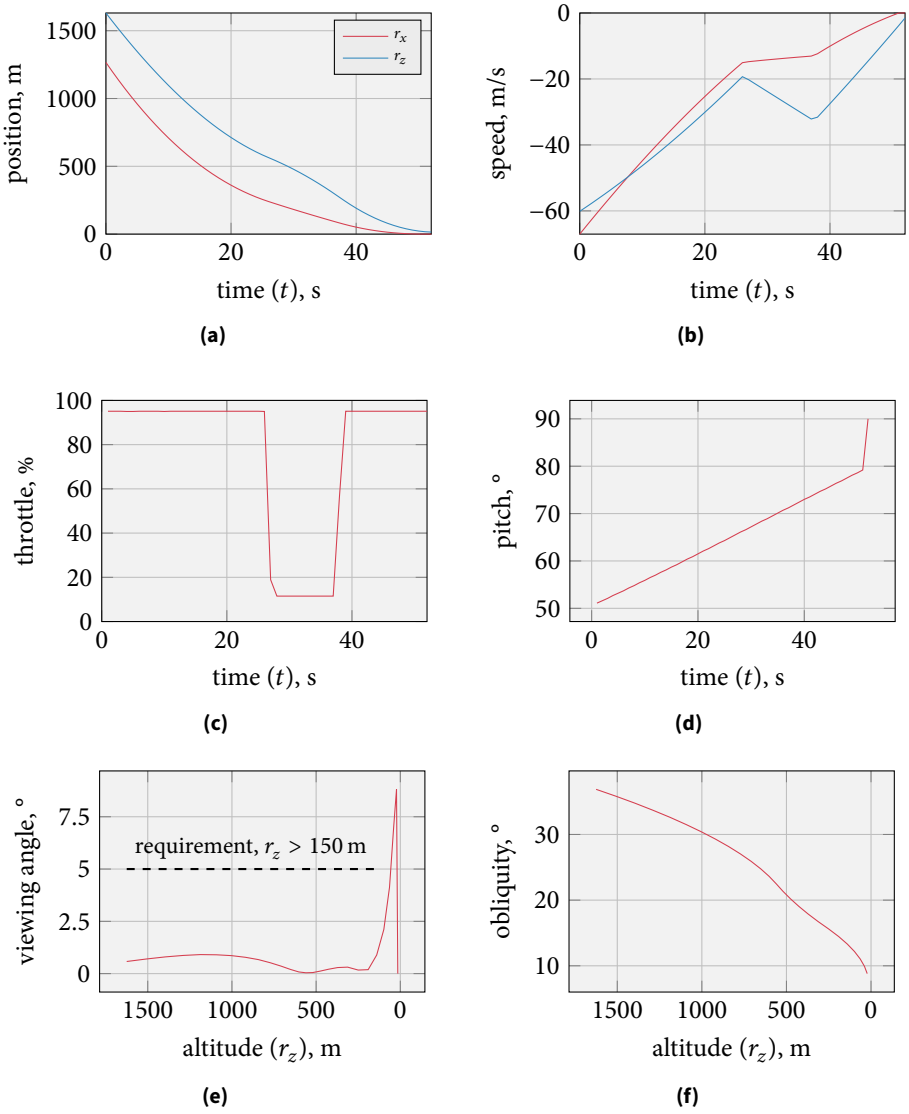


FIGURE 9.26 Guidance commands and resulting trajectory for the approach phase.

TABLE 9.13 Nominal guidance errors. Step sizes are 5 s, 1 s, and 0.01 s for the main braking and approach phase, as well as the continuous-time integration, respectively.

ERROR	r_x, m	r_y, m	r_z, m	v_x, m/s	v_y, m/s	v_z, m/s
AG	$-9.588 \cdot 10^{-2}$	$7.579 \cdot 10^{-16}$	-6.119	$4.631 \cdot 10^{-3}$	$-3.718 \cdot 10^{-19}$	$1.768 \cdot 10^{-1}$
TG	$1.126 \cdot 10^{-1}$	$-4.106 \cdot 10^{-17}$	$-9.977 \cdot 10^{-3}$	$4.488 \cdot 10^{-3}$	$-1.037 \cdot 10^{-18}$	$7.205 \cdot 10^{-2}$

quirement is actually relaxed. Moreover, the requirement of maintain a FOV error of less than 5° up to 150 m altitude is undershot by a large margin. The spike at the end of the flight can be explained by the final attitude, which is not exactly 90°. However, at TG the position vector points 90° upward, so that an increasing viewing angle error will build up.

As the flight progresses the obliquity also decreases, see Figure 9.26(f). On a positive note, the angle decreases ever further with decreasing altitude. This is beneficial for the quality of the sensor output, as the second retargeting phase is, in fact, planned between 300 m to 150 m in the ESA LL baseline scenario, see Figure 9.2.

On a final note, the pitch profile, Figure 9.26(d), would need some further consideration in future work. This is because the attitude jumps from 80° to 90° degree in the last two time steps, which is infeasible in reality. However, it is expected that using a pitch-up maneuver at TG this error should be correctable. This is to be studied in a future six-DOF simulation.

After discussing the control output of the convex guidance algorithm, an effort to verify its performance is essential as the next step. This is the subject of the upcoming section.

9.4.4 Performance analysis

The guidance performance can be analyzed for two aspects: the achieved landing precision and the needed propellant. Two more aspects will become relevant at later stages in the development, but cannot be considered at this point: the computational needs and the detailed HDA performance.

Starting with the landing precision, the errors occurring nominally at AG and TG are documented in Table 9.13. The errors documented in this table occur solely due to imperfections in the guidance algorithm itself, mostly resulting from the conversion to continuous time from discrete time. Any errors due to perturbations or the like have not been modelled, and the data in this table can therefore *not* be seen as the resulting final landing precision, but merely the error contribution of guidance.

Fortunately, it is apparent that the final errors are minimal. Considering the TG conditions, the largest position error occurring is 11.3 cm for the x-position. This is well below the requirement of 0.0 m ± 1.5 m (TG-1, Table 9.1 on Page 167). The altitude error is in the order of millimeters, and they are smaller than the downrange component because the altitude is chosen as TG trigger. Speed errors are also minimal, being no larger than several centimeter per second.

The nominal AG errors are larger, but still tolerably small. At 6.2 m, the figure for the altitude clearly stands out. However, this is not as bad as it seems. Main braking guidance is executed at a time-step of 5 s. For a time-of-flight of 660 s and a downrange

TABLE 9.14 Losses in comparison to a direct Hohmann transfer from 100 km down to the surface, a braking maneuver in the absence of gravity, and the actual scenario.

	HOHMANN	FREE SPACE	SCENARIO	LOSSES
Δv, m/s	1702.9	1703.0	1891.2	188.2
m_p, kg	750.9	750.9	810.7	59.8
t_{go}, s	0.0	591.3	712.0	120.7

of 630 km, such an error is very minimal. Moreover, it is larger than the downrange because r_x is taken as the AG trigger. Finally, there is plenty of room for correcting any errors at AG during the approach phase, so that the spacecraft can still be guided to the landing site. The speed error is also of sub-meter-per-second magnitude and therefore more than tolerable. Not surprisingly, all cross-range errors are vanishingly small, as there is nominally neither a cross-range speed nor position component.

Turning to the mass-wise performance, the total figure for the propellant need of 810.7 kg is to be verified. To do this, a fundamental test case is a comparison to the theoretical optimum: a Hohmann transfer from the 100 km LLO right down to the surface at 0 km. Note that only the braking burn at perilune is considered, because the DOI maneuver is not part of this study. Second, this can be compared to a braking burn in free space, in other words, a scenario with the same initial speed as the Hohmann transfer, but simulated in continuous time and the absence of gravity under finite thrust.

The results of this are printed in Table 9.14. The numbers show that at a needed propellant mass of 750.9 kg, the number for the designed scenario (810.8 kg) is off, but not by a huge margin. The numbers for the impulsive shot and the free-space burn are nearly identical. From the latter it can be deduced that no losses are modelled by the dynamics in the simulation, because the same functions have been used for integrating the scenario and the free-space burn. In consequence, the discrepancy can only be explained by gravity losses (see Equation [9.9], Page 172). Inspecting the time-difference needed for the free-space maneuver and the scenario, being 121 s, it appears as if there is little margin to decrease these losses: there is no other way to fly the trajectory in the gravity field around the Moon, but to spend extra time, which will necessarily result in gravity losses. Recall that gravity losses in terms of Δv are expressed as a time integral.

An interesting plot is shown in Figure 9.27, which shows the spent Δv against the burn time. Already in the beginning the two graphs for the ideal free-space burn and the scenario start to diverge. Later on the divergence increases in magnitude. This is a direct measure for the losses, which are also indicated graphically in the plot.

The only possible way to reduce gravity losses would be to increase the thrust-to-weight ratio, T/W, of the lander by either increasing the propulsive capabilities, decreasing the system mass, or both. This would allow for a shorter flight, a closer PDI to the landing site, and thus less losses. As a matter of fact, an impulsive shot models a maneuver at $T/W \to \infty$.

To compare the performance to some relevant studies rather than unrealistic ideal cases, the metrics of two references are listed in Table 9.15. Before these figures are discussed, it is stressed that it *cannot* be deduced from this table whether any of the scenarios is better! It is critical to realize this, which results from different assumptions such as varying vehicle baselines and system needs.

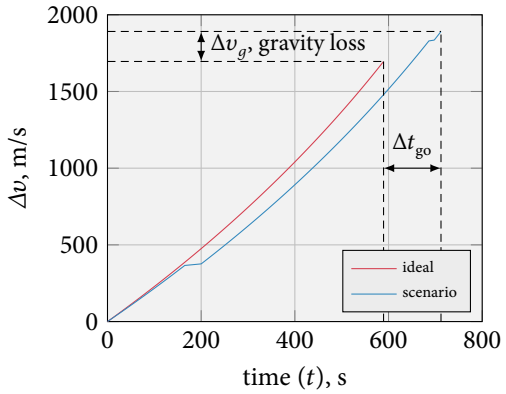


FIGURE 9.27 Growing discrepancy between an optimal, free-space braking burn and the designed scenario. Note the divergence right from the start, indicating small gravity losses.

TABLE 9.15 Performance comparison of the designed scenario to some references. Note the remarks below the table. These values are only *indicative* and *cannot* be considered a direct comparison! Sources: [Diedrich 2011], [Philippe 2014]

SCENARIO	m_0, kg	Δv, m/s	m_p, kg	t_{go}, s	I_{sp}, s	T_h, N
GERTH	1737	1891	809	712	306.7	3629
ASTRIUM	1544	1957	748	662	311 (PDI), 316 (AG)	3600
NGC-1	n/a	1840	n/a	687	n/a	3820

Even in spite of this limitation it is still clear that the figures for propellant mass – and especially the less biased parameter Δv – match rather closely. While the gravity losses were identified as 188 m/s, the values in Table 9.15 vary by 117 m/s. The designed scenario differs only by 51 m/s compared to the reference NGC-1. These are indications that the designed scenario does physically make sense and appears to be on par with industry.

Having studied the algorithm's performance and having found that convex guidance can deliver for the nominal scenario, its robustness to imperfect boundary conditions is yet to be studied. This is studied in the next section.

9.4.5 Robustness testing

Demonstrating the performance of convex guidance does not suffice to draw some final conclusions about its suitability for space applications. A decisive criterion yet to be tested is its *robustness*, which can be defined as:

Robustness The ability of a system to resist change without adapting its initial stable configuration. [Wieland and Wallenburg 2012]

In the scope of this analysis, this means that the guidance algorithm is to be stress tested under off-nominal initial conditions. An adequate means for this is an MC analysis. In an MC analysis, critical parameters are randomly dispersed according to realistically

TABLE 9.16 Dispersions at PDI considered in the MC analysis. Sources: [Philippe 2014], [Kerr et al. 2013]

	d, m	c, m	h, m	v_d, m/s	v_c, m/s	v_h, m/s	T, %	I_{sp}, %
DISPERSION (1σ)	3000	1500	600	1.5	1	0.6	0.667	1
DISTRIBUTION				— Gaussian —				uniform

TABLE 9.17 Pure guidance error statistics resulting from 1000 MC runs. 100 % of the runs were successful.

	AG			TG		
ERROR	WORST	MEAN μ	STD. 1σ	WORST	MEAN μ	STD. 1σ
r_x, m	$6.703 \cdot 10^{-1}$	$-3.344 \cdot 10^{-1}$	$1.848 \cdot 10^{-1}$	$1.159 \cdot 10^{-1}$	$1.128 \cdot 10^{-1}$	$9.881 \cdot 10^{-4}$
r_y, m	$4.532 \cdot 10^{-1}$	$3.213 \cdot 10^{-4}$	$1.161 \cdot 10^{-2}$	$8.225 \cdot 10^{-6}$	$-5.683 \cdot 10^{-8}$	$2.089 \cdot 10^{-6}$
r_z, m	$-1.253 \cdot 10^{1}$	-6.526	1.932	$1.551 \cdot 10^{-2}$	$-9.913 \cdot 10^{-3}$	$2.329 \cdot 10^{-3}$
v_x, m	$9.572 \cdot 10^{-3}$	$3.354 \cdot 10^{-3}$	$2.858 \cdot 10^{-3}$	$4.552 \cdot 10^{-3}$	$4.491 \cdot 10^{-3}$	$1.958 \cdot 10^{-5}$
v_y, m	$5.463 \cdot 10^{-5}$	$1.072 \cdot 10^{-7}$	$1.176 \cdot 10^{-5}$	$1.490 \cdot 10^{-7}$	$-1.010 \cdot 10^{-9}$	$3.773 \cdot 10^{-8}$
v_z, m	$2.466 \cdot 10^{-1}$	$1.646 \cdot 10^{-1}$	$3.977 \cdot 10^{-2}$	$7.397 \cdot 10^{-2}$	$7.187 \cdot 10^{-2}$	$2.292 \cdot 10^{-3}$

defined probabilities. Substituting these into a continuous-time simulation, it is then verified whether guidance can find solutions that meet the requirements.

To this end, dispersions are defined in Table 9.16. These correspond to the values used in the study of the ESA LL [Philippe 2014]. The execution of the MC analysis is simple. A loop is run N times, where at the beginning of each run the variables are dispersed. The results are then recorded. Previous studies have chosen $N = 200$ [Kerr et al. 2013] and $N = 100$ [Diedrich 2011]. Because the runtime admits a larger survey in this study, $N = 1000$ is chosen to be on the safe site statistically-speaking. This took 16 hrs to complete spread over two cores. The nature of the dispersions has been verified in the same manner as Figure 9.8, the results of which would be useless to plot here.

First of all, the algorithm was able to handle all dispersions and was able to find realizable solutions in 100 % of the cases. The most important results from 1000 runs are shown in Table 9.17, which lists the guidance error distributions and worst cases at AG and TG. Comparing the figures to Table 9.13 (Page 200), only marginal differences in the mean errors are observable. The most notable figure is the mean altitude error at AG, which has increased by 0.4 m with respect to to the nominal error. This also has a large spread resulting from taking the downrange as the AG trigger. Another interesting observation is that the mean errors for all crossrange components are minimal, even though there is a dispersion of up to 4.5 km (3σ) in this direction at PDI. This demonstrates that convex guidance can correct crossrange errors with ease. Overall, the final errors are fortunately very small, showing that guidance can correct for dispersed initial conditions.

Histograms for the resulting distributions are plotted in Figure 9.28 and Figure 9.29. It is noticeable that the errors are normally distributed, except for the downrange error at AG. This is again because it serves as AG trigger, and there will thus never be a positive error at this point.

The required propellant mass and Δv are shown in Figure 9.30. The mean propellant need is found to be 810.8 kg, which differs only marginally from the nominal 810.7 kg. In the best case, 806.7 kg can be expected, and 815.0 kg worst-case. The spread is therefore relatively small. The distribution does not appear to be Gaussian, but rather flat with few outliers that occur much less frequently than nominally. The same is true for the Δv, which can be seen as an equivalent of the propellant mass.

In conclusion, the MC campaign finishes with positive results. Not only are the errors small, but it has also been found that no excessively bad surprises need to be expected in terms of the propellant needs. Thus, formulating the verdict that convex guidance is robust is not unreasonable.

9.5 RECOMMENDATIONS

Having defined an end-to-end design of a D/L scenario, several recommendations can be made in retrospect. The most important conclusion is that the design is challenging and of necessarily highly iterative nature. This leads to several recommendations (in no particular order):

- For future studies, a *faster implementation* is of the essence. This will greatly ease design efforts by allowing for quicker analysis and iteration. This will also benefit from a better search algorithm.
- Vision-based *sensor models* should be included in the loop to assess navigation and HDA performance in detail.
- In the same manner, safe-site *diversions* (retargetings) must be included.
- Simulation *fidelity* should be increased overall by adding perturbations, navigation, and control to a six-DOF mode of the spacecraft.
- Following up on this, attitude *interfaces* at the gates must be studied in more detail.
- Finding an *analytical relation* between $\boldsymbol{v}_0$, $\boldsymbol{r}_0$, and γ would greatly ease and speed up design efforts—if possible at all.
- It would be useful to further study the *interaction of AG and PDI*.
- The addition of an inertial, *pointable sensor-platform* on the spacecraft, that can point at the landing site, has the potential of greatly improving viewing conditions and robustness while retaining more propellant optimality. Such a trade has to happen on system level.
- Other scenario *design-options*, such as adding hovering phase(s) for hazard mapping (HM) surveys shall be considered.

With these recommendations the chapter is concluded. Overall, a robust mission scenario has been designed using the available information. However, this thesis can only present the first step toward deeper analysis in the future.

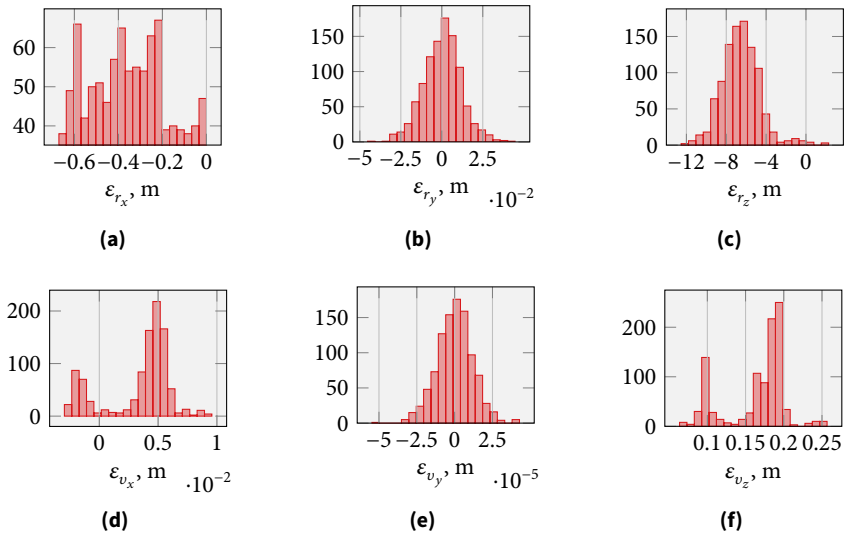


FIGURE 9.28 Distribution of state errors at AG resulting from 1000 MC runs.

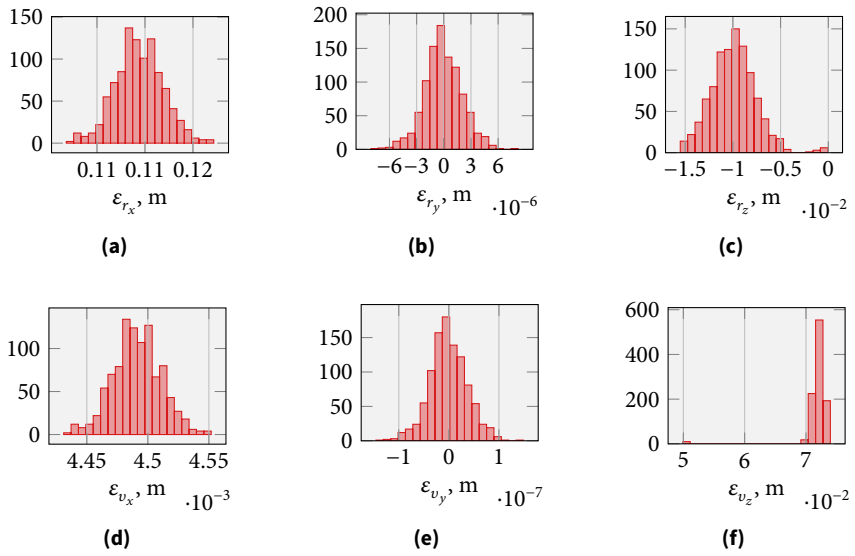


FIGURE 9.29 Distribution of state errors at TG resulting from 1000 MC runs.

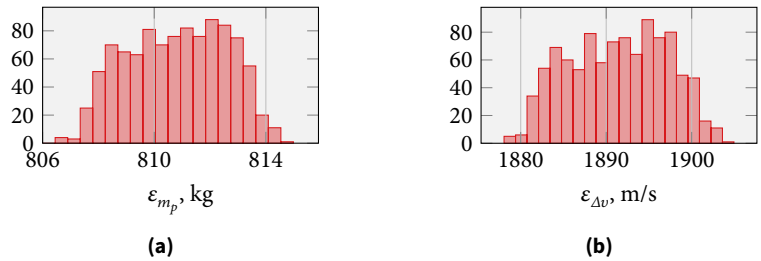
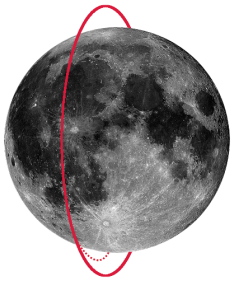


FIGURE 9.30 Histogram of the needed propellant mass and Delta V. The mean propellant need is 810.8 kg, or 1891.4 m/s of Delta V.

Chapter 10

CONCLUSIONS AND RECOMMENDATIONS



> "The true delight is in the finding out rather than in the knowing."
>
> *Isaac Asimov*

This thesis set out to answer the following research question:

> *Can convex optimization theory be used to derive a guidance algorithm that is globally propellant-optimal, can constrain the viewing conditions in the approach phase, and unify main braking and approach-phase guidance into a single algorithm?*

This question can be answered in the affirmative. Throughout the previous chapters the convex guidance law by Açıkmeşe and Ploen [2007] has been extended by several constraints and made compatible with Lunar flight. Based on this extensive work, several important conclusions will be drawn in this chapter, alongside recommendations for future work.

10.1 CONCLUSIONS

There are three important novelties in this work, that lead to the conclusion that convex guidance is potentially applicable for guiding Lunar landers all the way from the beginning of the braking phase up to the final few meters of vertical descent to the surface. These are:

1. A viewing-angle constraint has been successfully implemented by repeatedly solving the same optimization problem. In each iteration, the state history of the previous solution is used to determine the optimal pointing-direction for the spacecraft attitude. In this way it is possible to center the spacecraft's field of view (FOV) onto the landing-site up to a specified tolerance. Typically six iterations are required for this. In 6500 sample runs with a tolerance of 5°, it was found that feasible trajectories can be found for nearly any initial position, given an appropriate velocity vector and a suitable time-to-go.

   It was determined that the most important parameter for the feasibility of trajectories is the sensor angle, that determines under what angle an instrument is installed on the spacecraft. This was found to be very sensitive, and the correct angle is absolutely crucial for finding feasible trajectories. Only about 5 % of the 6500 runs were successful, where most failures can be traced to an unsuitable sensor angle. However, once the correct angle is found, the algorithm can robustly find solutions. The sensor angle is therefore not a limiting factor, but must be well defined.

2. Several new gravity-field models have been implemented, that allow for applying the algorithm to the main braking phase. This has previously been impossible. Next to this new application, it was found that a spherical-gravity is even necessary for the approach phase, which leads the lander from about 2 km downrange to the landing-site. If a flat Moon is assumed in this phase, errors in the order of 1 m to 2 m must be expected. The new models are therefore crucial additions for landing safety and requirements satisfaction.

   The first algorithm implements a full spherical-gravity model. It is based on solving a linear time-varying system (LTV) and requires to solve the optimization problem repeatedly. This way, it becomes possible to compute the nonlinear gravitational magnitude at each time step based on the state history of the previous iteration. For a scenario starting at 500 km downrange from the landing site and taking 600 s, convergence is typically achieved in six iterations or less.

   However, a flaw in this model – that has been taken from the literature – has been identified. The LTV is formulated in the inertial frame, whereas most constraints work in the landing-site frame. For this reason, a second model was developed. This handles gravity in the same way as a control-input vector. It is computed in the inertial frame, but the gravity vector is transformed to the landing-site frame. It is therefore possible again to solve the problem in the landing-site frame. As such, all constraints can be retained, while modelling spherical gravity at the same time. Moreover, the problem now uses a linear time-invariant system (LTI), which greatly reduces the number of floating-point operations required compared to an LTV.

   A third model assumes a spherical Moon and takes into account the curvature of the gravity field, but does not model the difference in magnitude while descending. This is based on solving the LTV only once, using the initial conditions. Compared to the LTI, this reduces the error by an order of magnitude. The full spherical model reduces the error by five orders of magnitude to the order of centimeters in position and millimeters per second in terms of speed, making it ignorable for practical applications.

   The form of the solution resulting from these models corresponds to the theoretical optimum that was expected from the bi-linear tangent steering-law, but convex guidance can fully constrain the final state, which some other steering laws cannot.

3. Finally, the guidance algorithms for the approach and main-braking phases differ only by minor modifications in the optimization problems. They share the same algorithmic framework, the same solver, and are based on the same theoretical foundations. It may therefore be concluded that the algorithms for main-braking and approach phases could successfully be consolidated.

One unfortunate fact of life derived from the need for iteration is that the time-to-go for the different phases can likely not be optimized on-board any more. This is because this also requires iteration, which would likely make the algorithm too slow. Initially, the time-to-go was to be optimized using a golden-section search on-board. To improve upon previous uses of this method in guidance applications, it was proposed to add parabolic interpolation to the golden-section search, which can speed up the

process. However, it was found that optimal and robust results can be achieved even with a fixed, pre-computed time-to-go, which is also done in other studies. In this work, the time-to-go is thus fixed based on pre-mission simulations. Future work could look into alternative methods, such as tabulating appropriate flight times.

Next to the new viewing-angle constraint mentioned above, two additional constraints from the literature have been implemented: a no-sub-surface-flight and a glide-slope constraint. In fact, the glide-slope constraint can take up the task of avoiding subterranean flight by itself. This was demonstrated in a scenario that would fly sub-surface in the unconstrained case, touch the surface with the no-sub-surface-flight constraint, but could safely approach the landing-site by staying above the ground with a glide-slope of 4°. An innovation in the scope of glide-slope constraints is their use for hazard avoidance. Large boulders can be avoided by choosing a glide-slope angle that leads to trajectories passing above them.

The algorithm was studied in detail for vehicle and scenario parameters based on the European Space Agency (ESA) Lunar Lander (LL). The first important conclusion from this study is that convex guidance is not compatible with the propulsion configuration of this vehicle, which relies on the successive shutdown of non-throttleable, not hot-restartable engines. Because of the globally optimal bang-bang (max-min-max) profile that is the output of convex guidance, vehicles applying this algorithm do require either hot-restartable or throttleable engines.

Having slightly modified the propulsion-system baseline of the LL by assuming that the main engines are pulseable up to a lower limit of 83.3 N, the robustness of the algorithm could be confirmed in an extensive scenario-design study based on a 6500-run Monte Carlo search. An *exemplary* design option was studied, that lead to a total Δv requirement of 1891 m/s, which is of similar magnitude as reference scenarios. Because of different assumptions, however, it is not possible to say with certainty whether this number is better or worse. The viewing angle is successfully constrained to remain below 5°, and never even exceeds 7.5°. The robustness of the algorithm could be successfully verified under realistic dispersions, that have been directly derived from the ESA LL mission. The initial state, thrust, and specific impulse have been dispersed at the start of the main-braking maneuver. 100 % out of 1000 runs were successful. Guidance only causes tiny landing errors, with the worst cases being 1.2 cm in position and 7.2 mm/s in speed.

Turning to the practical side, few conclusions on the on-board suitability can be drawn. It is known from the literature that convex guidance has the potential for on-board applications, and is being tested by the National Aeronautics and Space Administration (NASA) using terrestrial demonstration-vehicles. However, in this thesis the algorithm was only implemented using a prototyping environment in MATLAB. This does not allow to draw direct conclusions on the speed with which the algorithm might work on-board. Nevertheless, it could be qualitatively concluded that largest burden to the on-board computer would be the dynamical model. As the number of floating-point operations and memory use scale with with the square of the number of time steps, it is essential to have an efficient implementation. For this reason, an algorithm has been proposed that makes use of the matrices' sparsity properties and minimizes the number of operations. In this way, the memory footprint could of the dynamical model could be reduced from the order of hundreds of megabyte to merely hundreds of kilobytes. As the algorithm also works with a solver tailored for embedded systems,

there is thus strong evidence to assume that it could be on-board suitable for future missions.

In conclusion, a promising guidance algorithm with the potential for future on-board implementations on Lunar missions has been derived. However, further development efforts are required down the road to validate its final applicability. The most important next steps for this are presented in the following section.

10.2 RECOMMENDATIONS

Even though the foundations of the unified guidance algorithm have been laid in this report, there are still many challenges ahead. The following recommendations are the most important ones and suggested to be considered in future efforts:

- The optimization problem should be formulated in the canonical form to get rid of the intermediate modelling-language layer (cvx in this case). In combination with a more efficient implementation in a lower-level language, such as Ansi-C, the true computational efficiency of the algorithm should be investigated. Chebychev polynomials should be investigated to decrease the number of solution variables and speed up the process This should ideally be done with relevant hardware in-the-loop. Hardware tests by NASA indicate that the algorithm is on-board suitable, but independent validation of this is a must—especially with the modifications made in this thesis.

- Several further enhancements to the basic algorithm should be made. First, the addition of an attitude-rate constraint seems possible. This would solve occasional problems resulting from instantaneous jumps in the propulsion (and thus attitude) commands. Second, the rotational speed of the central body can be added directly to the equations of motion, which would be beneficial if more complex environmental models are used, as guidance works in a Lunar-fixed frame. Third, variations in the specific impulse due to engine clustering and throttle settings could be taken into account, because the command history of previous iterations is already known. Finally, the results from the iterations could be used to improve the linearization of the thrust bounds, as accurate values for the current vehicle mass could be substituted into this constraint. These improvements appear to be relatively simple to implement.

- Future simulations should significantly increase the sophistication of the simulation framework. To verify the algorithm's compatibility with the overall guidance, navigation, and control (GNC) system needs, it should be embedded into a closed-loop GNC simulation. Hazard detection and avoidance (HDA) should be added to the loop to test the diversion capabilities of the algorithm. Finally, detailed force models including perturbations should be taken into account to be able to judge the achievable landing precision.

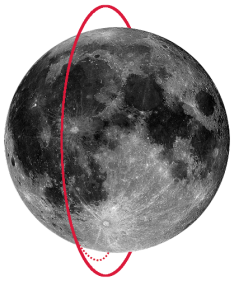
- Efforts for a detailed verification study comparing the applicability and optimality with regard to other guidance algorithms is necessary. Even though the mathematics proof that convex guidance is globally propellant-optimal, that gives no indication for how much better this is compared to other algorithms. Moreover,

some optimality is sacrificed in practice by choosing suboptimal – and fixed – time-of-flights to increase robustness. All these factors should be considered in a detailed trade-off, which is a daunting task.

- Future developments would benefit from deeper theoretical insight into the problem. Especially the relation between the initial position, velocity, and the resulting requirements for the sensor angle should be investigated. At this point, no obvious relation has been found, and the sensor angle has to be established by trial and error. Design efforts could be greatly sped up by having a good initial guess. Moreover, it would be useful to characterize the relation between the approach gate and the powered-descent-initiation point.

- It is highly recommended to develop a multidisciplinary design-optimization framework that assess the vehicle and trajectory design concurrently. It was found that the interaction between the different design parameters is significant and highly nonlinear. For example, the sensor angle, that dictates the viewing conditions, needs to be optimized for both, robustness and minimizing the FOV off-centering during approach. The propulsion-system configuration should be part of the loop as well.

- Related to the previous point, more options for landing scenarios should be considered in future studies. In this thesis, the ESA LL baseline scenario has been assumed, which is itself derived from Apollo. However, different architectures – such as direct descents or including hovering phases for hazard mapping – can be considered as well. Moreover, the impact of adding a pointable sensor-platform to the lander should be investigated, which would simplify the design of sensor-optimal trajectories (at added dry mass).



BIBLIOGRAPHY

Açıkmeşe, B. and L. Blackmore [2011]. "Lossless convexification of a class of optimal control problems with non-convex control constraitns". In: *Automatica*, vol. 47 [2].

Açıkmeşe, B.; L. Blackmore; D. P. Scharf, and A. Wolf [2008]. "Enhancements on the convex programming based powered descent guidance algorithm for mars landing". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. AIAA-08-6426.

Açıkmeşe, B.; J. Carson, and L. Blackmore [2013]. "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem". In: *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2114.

Açıkmeşe, B. and S. R. Ploen [2005]. "A Powered Descent Guidance Algorithm for Mars Pinpoint Landing". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. AIAA-05-6288.

– [2007]. "Convex Programming Approach to Powered Descent Guidance for Mars Landing". In: *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366. DOI: 10.2514/1.27553.

Allione, M. S.; A. L. Blackford; J. C. Mendez, and M. M. Whittouck [1968]. *Guidance, Flight Mechanics and Trajectory Optimization. Volume VI – The N-Body Problem and Special Perturbation Techniques*. Tech. rep. NASA CR-1005. George C. Marshall Space Flight Center, NASA.

Antsaklis, P. J. and A. N. Michel [2006]. *Linear Systems*. Birkhäuser.

Ball, A. J.; J. R. C. Garry; R. D. Lorenz, and V. V. Kerzhanovich [2007]. *Planetary Landers and Entry Probes*. Cambridge University Press.

Bennett, F. V. [1970]. "Apollo Lunar Descent and Ascent Trajectories". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. AIAA-70-25, pp. 503–509.

Ben-Tal, A. and A. Nemirovski [2001]. *Lectures on Modern Convex Optimization. Analysis, Algorithms, and Engineering Applications*. MPS-SIAM Series on Optimization.

Bertsekas, D. P. [1999]. *Nonlinear Programming*. Athena Scientific.

Betts, J. T. [1998]. "Survey of Numerical Methods for Trajectory Optimization". In: *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2. DOI: 10.2514/2.4231.

Blackmore, L.; B. Açıkmeşe, and D. P. Scharf [2010]. "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization". In: *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 4, pp. 1161–1171. DOI: 10.2514/1.47202.

Borrelli, F.; A. Bemporad, and M. Morari [2013]. *Predictive Control for Linear and Hybrid Systems*. UC Berkeley.

Boyce, W. E. and R. C. DiPrima [2008]. *Elementary Differential Equations*. Wiley.

Boyd, S. and L. Vandenberghe [2004]. *Convex Optimization*. Cambridge University Press.

Brent, R. P. [2013]. *Algorithms for Minimization Without Derivatives*. Dover Publications.

Bryson, A. E. and Y.-C. Ho [1975]. *Applied Optimal Control. Optimization, Estimation, and Control*. Taylor & Francis.

Carpenter, J. D.; O. Angerer; M. Durante; D. Linnarson, and W. T. Pike [2010]. "Life Sciences Investigations for ESA's First Lunar Lander". In: *Earth Moon Planets*, vol. 107, pp. 11–23. DOI: 10.1007/s11038-010-9375-y.

Carpenter, J. D.; R. Fisackerly; D. De Rosa, and B. Houdou [2012]. "Scientific Preparations for Lunar Exploration with the European Lunar Lander". In: *Planetary and Space Science*, p. 51.

Cheng, R. K. [1964]. *Surveyor terminal guidance. NASA-CR-57550*. Tech. rep. Jet Propulsion Laboratory.

Cherry, G. W. [1964]. "A general, explicit, optimizing guidance law for rocket-propelled spaceflight". In: *AIAA/ION Astrodynamics Guidance and Control Conference*. AIAA-64-638. DOI: 10.2514/6.1964-638.

Citron, S. J.; S. E. Dunin, and H. F. Meissinger [1964]. "A terminal guidance technique for lunar landing". In: *AIAA Journal*, vol. 2, no. 3.

Conway, B. [2010]. *Spacecraft Trajectory Optimization*. Cambridge University Press.

Cornelisse, J. W.; H. F. R. Schöyer, and K. F. Wakker [1979]. *Rocket Propulsion and Spaceflight Dynamics*. Pitman.

Crane, E. S. and S. M. Rock [2012]. "Influence of Trajectory on Accuracy of Hazard Estimation During Lunar Landing". In: *AIAA Guidance, Navigation, and Control Conference*. AIAA-12-4554.

Crawford, I. A.; M. Anand; C. S. Cockell; H. Falcke; D. A. Green; R. Jaumann, and M. A. Wieczorek [2012]. "Back to the Moon. The scientific rationale for resuming lunar surface exploration". In: *Planetary and Space Science*, vol. 74 [1], pp. 3–14. DOI: 10.1016/j.pss.2012.06.002.

De Rosa, D.; R. Fisackerly; A. Pradier; C. Philippe; B. Houdou; J. Carpenter, and B. Gardini [2011]. "ESA Lunar Lander Mision". In: *International ESA Conference on Guidance, Navigation & Control Systems*.

Deb, K. [2002]. *Optimization for Engineering Design. Algorithms and Examples*. Prentice Hall.

Delaune, J.; D. D. Rosa, and S. Hobbs [2010]. "Guidance and Control system design for Lunar Descent and Landing". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. August.

Diedrich, T. [2011]. *Design Definition File. LLB1-MS-TN-AST-0005*. Tech. rep. Astrium GmBH Bremen, p. 133.

Domahidi, A.; E. Chu, and S. Boyd [2013]. "ECOS: An SOCP Solver for Embedded Systems". In: *European Control Conference*, pp. 3071–3076.

D'Souza, C. N. [1997]. "An optimal guidance law for planetary landing". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. AIAA-97-3709. DOI: 10.2514/6.1997-3709.

ECSS – European Cooperation for Space Standardization – Secretariat [2004]. *ECSS-E-60A – Space Engineering/Control Engineering*. ESA-ESTEC, Requirements & Standards Division. 76 pp.

Engelbrecht, A. P. [2007]. *Computational Intelligence*. Wiley.

Fisackerly, R.; A. Pradier; B. Gardini; B. Houdou; C. Philippe; D. De Rosa, and J. Carpenter [2011]. "The ESA Lunar Lander Mission". In: *AIAA SPACE 2011 Conference & Exposition*. AIAA-11-7217. DOI: 10.2514/6.2011-7217.

Gerth, I. [2013]. "Guidance and Control Methods for Safe and Precise Lunar Landing". Literature Study. Delft University of Technology.

Gerth, I. and E. Mooij [2014]. "Guidance for Autonomous Precision Landing on Atmosphere Bodies". In: *AIAA Guidance, Navigation, and Control Conference*. AIAA-14-0088.

Grant, M. and S. Boyd [2008]. "Graph implementations for nonsmooth convex programs". In: *Recent Advances in Learning and Control*. Ed. by V. Blondel; S. Boyd, and H. Kimura. Lecture Notes in Control and Information Sciences. http://stanford.edu/~boyd/graph_dcp.html. Springer, pp. 95–110.

– [2014]. *CVX: Matlab Software for Disciplined Convex Programming, version 2.1 beta*. URL: http://cvxr.com/cvx (Retrieved: March 19, 2014).

Grumman – Grumman Aerospace Corporation [1971]. *Apollo Operations Handbook. Lunar Module*. LM 10 and Subsequent. Volume I – Subsystems Data. Tech. rep. NAS 9-1100. Exhibit E – Paragraph 10.4.

Hall, R. C. [1977]. *Lunar Impact. A History of Project Ranger*. NASA History Series.

Harvey, B. [2006]. *Soviet and Russian Lunar Exploration*. Springer.

Heppenheimer, T. A. [1999]. *Countdown. A History of Space Flight*. John Wiley & Sons.

Hindi, H. [2004]. "A tutorial on convex optimization". In: *American Control Conference*. Vol. 4, pp. 3252–3265.

Hoffmann-Wellenhof, B.; H. Lichtenegger, and E. Wasle [2007]. *GNSS: Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer.

Huertas, A.; Y. Cheng, and L. Matthies [2007]. "Real-time Hazard Detection for Landers". In: *NASA Science Technology Conference*.

Huntress Jr., W. T. and Y. M. Marov [2011]. *Soviet Robots in the Solar System*. Springer.

Iserles, A. [2009]. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press.

Johnson, N. L. [1995]. *Reach for the Moon. The L-1 and L-3 Manned Lunar Programs and the Story of the N-1 "Moon Rocket"*.

Kerr, M. L.; M. Hagenfeldt; J. A. Ospina; J. M. Ramón; L. F. Peñín; M. Mammarella; A. Bidaux, and P. Colmenarejo Matellano [2013]. "ESA Lunar Lander: Approach Phase Concept and G&C Performance". In: *AIAA Guidance, Navigation, and Control Conference*. AIAA-13-5018.

Klumpp, A. R. [1971]. *Apollo Guidance Navigation and Control*. Tech. rep. R-695. Boston: Massachusetts Institute of Technology.

Kos, L. D.; T. P. Polsgrove; R. R. Sostaric; E. M. Braden; J. J. Sullivan, and T. T. Le [2010]. "Altair Descent and Ascent Reference Trajectory Design and Initial Dispersion Analyses". In: *AIAA Guidance, Navigation, and Control Conference*. AIAA-10-7720.

Lakdawalla, E. [2014]. "China lands on the Moon". In: *Nature Geoscience*, vol. 7, no. 2, p. 81.

Lambert, J. [1991]. *Numerical Methods for Ordinary Differential Equations. The Initial Value Problem*. John Wiley & Sons.

Larson, W. J. [2009]. *Applied Space Systems Engineering*. Ed. by W. J. Larson; D. Kirkpatrick; J. J. Sellers; L. D. Thomas, and D. Verma. McGraw-Hill.

Liu, X. and P. Lu [2013]. "Solving Non-Convex Optimal Control Problems by Convex Optimization". In: *AIAA Guidance, Navigation, and Control Conference*.

Löfberg, J. [2004]. "YALMIP: A Toolbox For Modelling and Optimization in MATLAB". In: *CACSD conference*.

Lueneberger, D. G. [1979]. *Introduction to Dynamic Systems. Theory, Models, and Applications*. Wiley.

Mathworks – The MathWorks, Inc. [2014]. *fminbnd. Find minimum of single-variable function on fixed interval*. URL: http://www.mathworks.nl/help/matlab/ref/fminbnd.html (Retrieved: February 27, 2014).

Meditch, J. [1964]. "On the problem of optimal thrust programming for a lunar soft landing". In: *IEEE Transactions on Automatic Control*, vol. 9 [4], pp. 477–484.

Mittelmann, H. D. [2008]. *Newer SDP/SOCP-codes on the 7th DIMACS Challenge problems*. URL: http://plato.asu.edu/ftp/dimacs_sdp.html (Retrieved: September 6, 2013).

– [2012]. "The State-of-the-Art in Conic Optimization Software". English. In: *Handbook on Semidefinite, Conic and Polynomial Optimization*. Ed. by M. F. Anjos and J. B. Lasserre. Vol. 166. International Series in Operations Research & Management Science. Springer US, pp. 671–686. DOI: 10.1007/978-1-4614-0769-0_23.

– [2013]. *MISOCP and large SOCP Benchmark*. URL: http://plato.asu.edu/ftp/socp.html (Retrieved: September 6, 2013).

Montenbruck, O. and E. Gill [2000]. *Satellite Orbits. Models, Methods, and Applications*. Springer.

Mooij, E. [1997]. *The Motion of a Vehicle in a Planetary Atmosphere*. Series 08: Astrodynamics and Satellite Systems. Delft University Press.

– [1998]. "Aerospace-Plane Flight Dynamics. Analysis of Guidance and Control Concepts". PhD thesis. Delft University of Technology.

Mulder, J. A.; W. H. J. J. van Staveren; J. C. van der Vaart; E. de Weert; A. C. in 't Veld, and E. Mooij [2011]. *Flight Dynamics*. Lecture notes. Delft University of Technology.

NASA – National Aeronautics and Space Administration [1966]. *Surveyor I Mission Report*. Tech. rep. TR 32-1023. Jet Propulsion Laboratory.

NASA – Mission Evaluation Team [1969]. *Apollo 11 Mission Report*. Tech. rep. MSC-00171. Manned Spacecraft Center. 348 pp.

NASA – – [1971]. *Apollo 14 Mission Report*. Tech. rep. MSC-04112. Manned Spacecraft Center. 268 pp.

NASA – – [1972]. *Apollo 16 Mission Report*. Tech. rep. MSC-07230. Manned Spacecraft Center. 407 pp.

Orloff, R. W. [2001]. *Apollo by the Numbers*. NASA SP-2000-4029. NASA History Series.

Paschall, S.; T. Brady; T. Fill, and R. R. Sostaric [2009]. "Lunar Landing Trajectory Design for Onboard Hazard Detection & Avoidance". In: pp. 1–20.

Perkins, F. M. [1966]. *Derivation of Linear-Tangent Steering Laws*. Tech. rep. Aerospace Corporation.

Pfeiffer, C. G. [1968]. *An Analysis of Guidance Modes*. Tech. rep. NASA Electronics Research Center.

Philippe, C. [2012]. "Discussion on Requirements". Personal communication.

– [2014]. "Communication on approach gate data". Personal communication.

Pólik, I.; J. J. Cochran; L. A. Cox; P. Keskinocak; J. P. Kharoufeh, and J. C. Smith [2010]. "Conic Optimization Software". In: *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc. DOI: 10.1002/9780470400531.eorms0182.

Press, W. H.; S. A. Teukolsky; W. T. Vetterling, and B. P. Flannery [2007]. *Numerical Recipes 3rd Edition. The Art of Scientific Computing*. Cambridge University Press.

Rea, J. R. [2009]. "An Investigation of Fuel Optimal Terminal Descent". PhD thesis. University of Texas at Austin.

Ribarich, J. [1968]. "Surveyor spacecraft landing accuracy". In: *Journal of Spacecraft and Rockets*, vol. 5, no. 7, pp. 768–773.

Rugh, W. J. [1995]. *Linear System Theory*. Prentice Hall.

Schaub, H. and J. L. Junkins [2009]. *Analytical Mechanics of Space Systems*. AIAA Education Series.

Schoenemaekers, J. [2011]. *Lunar Lander Phase B1: Consolidated Report on Mission Analysis. LLB1-CREMA-ESA(HSO)-0001 (MAS WP 569)*. Tech. rep. ESA ESOC.

Schulte, G. [2004]. *400 N Engine Qualification Test Report*. Tech. rep. 400N-RILAM-TRP-0006. EADS SPACE TRANSPORTATION.

Shuster, M. D. [1993]. "A Survey of Attitude Representations". In: *The Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517.

Soppa, U. [2011]. *Mission Analysis and Definition Document. LLB1-MS-TN-AST-0004*. Tech. rep. Astrium GmbH Bremen, p. 195.

Spall, J. C. [2003]. *Introduction to Stochastic Search and Optimization. Estimation, Simulation, and Control*.

Stengel, R. F. [1986]. *Optimal Control and Estimation*. Dover Publications.

Strandmoe, S. E.; T. Jean-Marius, and S. Trinh [1999]. "Toward a vision based autonomous planetary lander". In: *AIAA Guidance, Navigation and Control Conference*. AIAA-99-4154.

Sun, Z.; Y. Jia, and H. Zhang [2013]. "Technological advancements and promotion roles of Chang'e 3 lunar probe mission". In: *Science China Technological Sciences*, vol. 56, no. 11, pp. 2702–2708.

Sutton, G. P. and O. Biblarz [2010]. *Rocket Propulsion Elements*. John Wiley & Sons.

Topcu, U.; J. Casoliva, and K. D. Mease [2005]. "Fuel Efficient Powered Descent Guidance for Mars Landing". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. AIAA-05-6286.

Vallado, D. A. [2007]. *Fundamentals of Astrodynamics and Applications*. Microcosm Press.

Wertz – Members of the Technical Staff, Attitude Systems Operation. Computer Sciences Corporation. [1979]. *Spacecraft Attitude Determination and Control*. Ed. by J. R. Wertz. Kluwer Academic Publisher.

Widnall, W. S. [n.d.]. *Optimal Control in the Lunar Module Digital Autopilot*. Report for internal use. Provided by Christian Philippe (ESA). Massachusetts Institute of Technology. 17 pp.

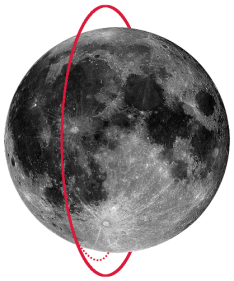
Wieland, A. and C. M. Wallenburg [2012]. "Dealing with supply chain risks: Linking risk management practices and strategies to performance". In: *International Journal of Physical Distribution & Logistics Management*, vol. 42, no. 10.

Williams, L. [2013]. *JPL, Masten Testing New Precision Landing Software*. NASA. URL: http://www.jpl.nasa.gov/news/news.php?release=2013-247 (Retrieved: September 7, 2013).

Wong, E. C.; G. Singh, and J. P. Masciarelli [2006]. "Guidance and Control Design for Hazard Avoidance and Safe Landing on Mars". In: *Journal of Spacecraft and Rockets*, vol. 43, no. 2, pp. 378–384.

Zuber, M. T.; D. E. Smith, and G. A. Neumann [2013]. *Clementine Gravity and Topography Data*. Ed. by R. E. Arvidson. URL: http://pds-geosciences.wustl.edu/lunar/clem1-gravity-topo-v1/cl_8xxx/gravity/glgm2sh.tab (Retrieved: June 24, 2013).

GLOSSARY



▶ A

Absolute Vision-Based Navigation

: The task of estimating the spacecraft's translational state by tracking features on the Lunar surface and comparing these to an on-board catalogue (feature matching). Applied during the coasting arc of the Hohmann transfer.

Altitude

: Distance between the spacecraft and the sub-spacecraft point along vector planet's center to the current position of the spacecraft.

Approach Phase

: The flight segment following up on the main-braking phase, starting at high gate up until terminal gate.

Attitude Control

: The task of attaining and stabilizing the vehicle in the attitude configuration for the guidance system.

Attitude Parameter

: A set of coordinates that completely describe the orientation of a rigid body relative to some reference frame.

▶ B

▶ C

Coordinate System

: A mathematical description of vectors with respect to a given axis system.

Crossrange

: Distance, measured on the planet surface, perpendicular to the downrange direction.

▶ D

Descent Orbit

The leg from the DOI to the main-braking phase, the Hohmann transfer orbit itself.

Descent Orbit Insertion

The burn lowering the perilune of the LLO for beginning a Hohmann transfer toward the landing site.

Divert Maneuver

A guidance phase during which the lander alters its trajectory to fly to a revised target selected at the retargeting event.

Downrange

The distance, measured on the planet surface, from the current sub-spacecraft point to the origin of the landing-site frame.

▶ E

Elastic Body

A mathematical concept for modelling the attitude motion of a body of finite dimensions, where structural dynamics are considered and the mass distribution is variable.

▶ F

Failure

The inability of a system, subsystem, component or part to perform its required function within specified limits under specified conditions for a specified duration.

Failure Tolerance

The ability to continue to operate in the presence of anomalies or failure. The number of failures, which can be allowed without disruption of nominal functional performance.

Final Target Acquisition (segment)

The second segment of the approach phase, positioning the lander on top of the target after final retargeting. During this phase, the target is not necessarily in the center of the HDA-sensor FOV anymore.

Final Terminal Descent (segment)

The third segment of the approach phase, consisting of a vertical descent towards the final target.

▶ G

Guidance

The task of calculating and executing a realizable acceleration profile, which will cause the trajectory of the space vehicle to attain desired end conditions.

Guidance Mode (or algorithm)

A policy for calculating the parameters and functions which will accomplish the guidance task.

▶ H

Hazard

A feature that can compromise or make impossible a safe landing.

Hazard Avoidance

The function deciding whether a divert should be initiated or not, by evaluating the overall risk map and choosing a new landing site. Also known as piloting.

Hazard Detection

The function fusing the hazard maps to create an overall risk map of the landing area.

Hazard Detection and Avoidance

A term comprising hazard mapping, hazard detection, and hazard avoidance as a single function.

Hazard Map

A set of data points spread over the landing area, that associates a landing risk for a certain criterion to each of these points.

Hazard Mapping

The function that evaluates sensor data of the landing area and its vicinity, that associates a quantitative risk to each measurement point for a number of different hazards, and thus generates hazard maps from this information.

High Gate

The position vector associated to the point in time that the landing site comes into the field of view of the camera. (High gate may be encountered before approach gate.)

▶ I

Inelastic Body

A mathematical concept for modelling the attitude motion of a body of finite dimensions, assuming an inelastic structure and variable mass.

▶ L

Landing Area

The region around the nominal landing site with conditions that permit a landing.

Landing Target

The currently selected point on the Lunar surface to which the lander is guided.

Landing-Site Evaluation (phase)

The phase during which hazard detection and hazard mapping evaluate the landing area.

Low Gate

The position vector associated to the point in time of the where the visual to the landing site is lost.

Low Lunar Orbit

Refers to a 100 km Lunar parking orbit.

▶ M

Main-Braking Phase

The period from starting the full thrust burn at perilune of the LLO to initiate the final landing phase up to the moment of throttling back.

Miss Distance

The distance between finally selected target and actual touchdown point. (Not the absolute landing error.)

▶ N

Navigation

The task of estimating the state of the space vehicle system from sensed data.

Nominal Landing Site

The landing coordinates specified prior to the mission.

▶ P

Piloting

see hazard avoidance.

Pinpoint Landing

The capability of precisely landing at a predefined location, typically defined as less than four hundred meters of absolute landing error for the Moon.

Point Mass

A mathematical concept for modelling the translational motion of a body, which assumes that it is of infinitesimal dimension and constant mass.

Precision Landing

A landing within 200 m of the nominal landing site.

Primary Guidance

The task of generating the acceleration profile to be to be flown by the spacecraft.

▶ R

Reference Frame

A set of axes that serve as reference for a coordinate system, which are aligned to a defined origin and orientation.

Relative Vision-Based Navigation

The task of estimating the spacecraft's translational state by tracking features in the landing area, that do not necessarily need to be known in advance. Applied during the approach phase.

Retargeting (event)

The point in time that hazard avoidance decides to command a divert to a safer landing site.

Retargeting Area

The (circular) area about the current target with permissible conditions, considered by the hazard avoidance function.

Rigid Body

A mathematical concept for modelling the attitude motion of a body of finite dimensions, assuming an inelastic structure and constant mass.

Robustness

The ability of a system to resist change without adapting its initial stable configuration.

Roughness

Maximum absolute terrain height deviation from the mean plane under the lander footprint caused by topographic features (such as boulders, craters, crevasses, scarps).

▶ S

Safe Landing Site

A site which is compliant with all landing-site requirements.

Simplified Guidance

The task of generating acceleration profiles to estimate the propellant consumption needed to reach potential landing sites, thus creating a propellant map.

Soft Landing

A landing compliant with the requirements.

Sub-spacecraft point

The projection of spacecraft position on planet's ground.

▶ T

Target Pointing Segment

The first segment of the approach phase with pitch coordinated descent during landing site evaluation. During this approach, the active target is in the center of the HDA sensor FOV. Diverts as a consequence of retargetings will be made from this baseline trajectory.

Terminal Gate

The position vector associated to the point in time where the terminal (vertical) descent is initiated.

Throttle Back/Approach Gate (event)

The position vector associated to the point in time after main braking that the engines are throttled back and the approach-phase guidance is initiated.

Touchdown

The position vector associated to the point in time where all of the vehicle's landing legs come to a rest on the central body's surface.

▶ V

Validation

The process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

Verification

The process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.

Appendix A

OVERVIEW OF REFERENCE SCENARIOS

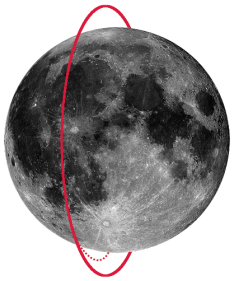
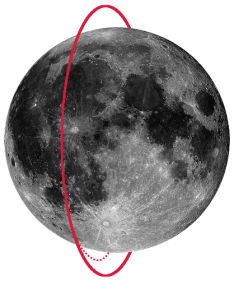


TABLE A.1 Descent and landing parameters of reference scenarios.

PARAMETER	DEIMOS	ASTRIUM	NGC-1	NGC-2	DELAUNE	GERTH
ACTIVITY	ESA LL phase-B1 guidance, control, and HDA concept	ESA LL phase-B1 guidance, control, and HDA concept	ESA moon-landing HDA-concept activity	ESA robust EDL activity	MSC thesis at ESTEC	MSC thesis at TU DELFT, with ESTEC
SOURCE	[Kerr et al. 2013]	[Soppa 2011]	[Philippe 2014]	[Philippe 2014]	[Delaune et al. 2010]	—
DOI LLO	100 km × 10 km	100 km × 10 km	100 km × 10 km	100 km × 12.5 km	100 km × 10 km	100 km × 10 km
PDI, km	700	506.9	583	??	500	630
GUIDANCE, DOI	??	NLP	PEG	gravity turn	bi-linear tangent	convex
GUIDANCE, AG	NLP (TBC)	NLP	Apollo	??	Apollo	convex
MAX. THRUST, N	3820	3600	3820	??	3500	3629
I_{SP}, s	??	311	??	??	320	306.7
t_{PDI}, s	740	539.7	613	??	??	660
t_{AG}, s	90 (incl. diverts)	90	74	??	80	52 (no diverts)
m_0, PDI, kg	1722	1544	??	??	1500	1737
m_0, AG, kg	??	858	??	760	856	977
m_0, TG, kg	??	788	??	??	??	926
r_x, AG, m	2500	1306	200	1600	1500	1265
r_z, AG, m	1500	1655	1000	1570	2000	1625
v_x, AG, m/s	90	78.9	5	110	40	67
v_z, AG, m/s	10	37.5	30	40	50	60



Appendix B

DISCRETIZATION OF SYSTEM MATRICES

This appendix extends the discussion in Section 6.6.1.2, that treats the error analysis of the linear time-invariant system (LTI) gravity dynamics. For th mathematical background see [Antsaklis and Michel 2006].

B.1 VERTICAL EQUATIONS OF MOTION

To give a simple example of an LTI state-space system and its discretization, consider the vertical motion of a point mass in a constant gravity field. This is the one-dimensional, constant-mass case of the same problem as considered in the following section.

The equation of motion (EOM) is:

$$\ddot{r} = g \tag{B.1}$$

where r is the position (altitude) and g the gravitational acceleration. This can be cast to a continuous-time state-space model as:

$$\underset{\dot{\boldsymbol{x}}}{\begin{pmatrix} \dot{r} \\ \ddot{r} \end{pmatrix}} = \underset{A_c}{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}} \underset{\boldsymbol{x}}{\begin{pmatrix} r \\ \dot{r} \end{pmatrix}} + \underset{B_c}{\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}} \underset{\boldsymbol{u}}{\begin{pmatrix} 0 \\ g \end{pmatrix}} \tag{B.2}$$

with the standard notation as introduced in Section 6.3.2. In this case, g is treated as if it was a constant control input, although it really is not. Because this vector multiplying B_c could also be replaced by an acceleration control-vector, the term and notation "input matrix" and B_c are kept here nonetheless.

The discretization can be performed by carrying out by the following manipulations:

$$\Psi(\Delta t) = \sum_{j=0}^{\infty} \frac{\Delta t^j}{(j+1)!} A_c^j \tag{B.3}$$

$$A_d = \mathsf{I} + \Delta t A_c \Psi(\Delta t) \tag{B.4}$$

$$B_d = \Delta t \Psi(\Delta t) B_c \tag{B.5}$$

First compute the discretization matrix Ψ:

$$\Psi(\Delta t) = \mathsf{I}_{2\times 2} + \frac{\Delta t}{2} A_c + \underbrace{\frac{\Delta t^2}{6} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \cdots}_{0} \tag{B.6}$$

$$= \begin{bmatrix} 1 & \Delta t/2 \\ 0 & 1 \end{bmatrix} \tag{B.7}$$

Then A_d can be determined as:

$$A_d = \mathsf{I}_{2\times2} + \Delta t A_c \Psi(\Delta t) = \mathsf{I}_{2\times2} + \underbrace{\Delta t \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & \Delta t/2 \\ 0 & 1 \end{bmatrix}}_{0} \tag{B.8}$$

$$= \mathsf{I}_{2\times2} \tag{B.9}$$

The input matrix B_d is:

$$B_d = \Delta t \Psi(\Delta t) B_c = \Delta t \begin{bmatrix} 1 & \Delta t/2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{B.10}$$

$$= \begin{bmatrix} 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix} \tag{B.11}$$

With these matrices, the discrete-time system can be written as:

$$\begin{pmatrix} r(k+1) \\ \dot{r}(k+1) \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} r(k) \\ \dot{r}(k) \end{pmatrix} + \begin{bmatrix} 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix} \begin{pmatrix} 0 \\ g \end{pmatrix} \tag{B.12}$$

An important observation can be made if these matrix multiplications are carried out and are compared to the analytical solutions:

$$r(k+1) = r(k) + \tfrac{1}{2} g \Delta t^2 \quad \Leftrightarrow r(t) = r_0 + v_0 t + \tfrac{1}{2} g t^2 \tag{B.13}$$

$$\dot{r}(k+1) = \dot{r}(k) + g\Delta t \quad \Leftrightarrow v(t) = v_0 + gt \tag{B.14}$$

This shows that in this case the discrete-time solution of the LTI is indeed identical to the analytical solution *at the temporal nodes*. Therefore, the discrete-time solution actually does not introduce any errors. However, the fact that no accurate value is available in between nodes is not remedied by this, and can only be mitigated by choosing smaller steps-sizes.

B.2 FULL LINEAR SYSTEM DYNAMICS

The discrete-time system and input matrices of the full system dynamics used in convex guidance can be derived with the same methodology as applied in the previous section. The state-space model is:

$$\begin{pmatrix} \dot{\boldsymbol{r}}(t) \\ \ddot{\boldsymbol{r}}(t) \\ \dot{z}(t) \end{pmatrix} = \begin{bmatrix} 0_{3\times3} & \mathsf{I}_{3\times3} & 0_{3\times1} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{r}(t) \\ \dot{\boldsymbol{r}}(t) \\ z(t) \end{pmatrix} + \begin{bmatrix} 0_{3\times3} & 0_{3\times1} \\ \mathsf{I}_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & -\alpha \end{bmatrix} \begin{pmatrix} \boldsymbol{\tau}(t) \\ \sigma(t) \end{pmatrix} + \cdots$$

$$\cdots \begin{bmatrix} 0_{3\times3} & 0_{3\times1} \\ \mathsf{I}_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & -\alpha \end{bmatrix} \begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix} \tag{B.15}$$

For the discretization, the first step is to compute the matrix $\Psi(\Delta t)$, which requires the expression for A_c^j. It is easy to show that for all $j > 1$ it holds:

$$A_c^j = \begin{bmatrix} 0 & \mathsf{I} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \mathsf{I} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdots = 0, \quad \forall j > 1 \tag{B.16}$$

where the matrix indices have been dropped for clarity of the notation. Refer to Equation [B.15] for the correct dimensions. Now it is easy to show that:

$$\Psi(\Delta t) = \sum_{j=0}^{\infty} \frac{\Delta t^j}{(j+1)!} = \mathsf{I} + \frac{\Delta t}{2} A_c \tag{B.17}$$

With this, the discrete-time system matrix can be derived:

$$A_d = \mathsf{I} + \Delta t A_c \Psi(\Delta t) = \mathsf{I} + \Delta t A_c (I + \frac{\Delta t}{2} A_c) = \mathsf{I} + \Delta t A_c \tag{B.18}$$

where it has been made use of the property that $A_c^2 = 0$ as shown above.

The discretized input matrix is determined as follows:

$$B_d = \Delta t \Psi(\Delta t) B_c = \Delta t \left(\mathsf{I} + \frac{\Delta t}{2} A_c \right) B_c = \Delta t \left(B_c + \frac{\Delta t}{2} A_c B_c \right) \tag{B.19}$$

The matrix product $A_c B_c$ evaluates as:

$$A_c B_c = \begin{bmatrix} 0 & \mathsf{I} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ \mathsf{I} & 0 \\ 0 & -\alpha \end{bmatrix} = \begin{bmatrix} \mathsf{I} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{B.20}$$

Therefore, evaluating Equation [B.19] leads to the final matrix:

$$B_d = \begin{bmatrix} (\Delta t^2/2)\mathsf{I} & 0 \\ \Delta t \mathsf{I} & 0 \\ 0 & -\Delta t \alpha \end{bmatrix} \tag{B.21}$$

Although the basic matrices are now derived, it is useful to derive two further variations. In Section 6.4.1, it was shown that stacked solution matrices are required for the second-order cone-problem (SOCP). These are:

$$S_x = \begin{bmatrix} A_d^0 & 0 & \cdots & \cdots & 0 \\ 0 & A_d^1 & \ddots & \cdots & \vdots \\ 0 & \ddots & A_d^2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & A_d^N \end{bmatrix} \quad S_u = \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ B_d & 0 & \cdots & 0 \\ A_d B_d & B_d & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_d^{N-1} B_d & \cdots & \cdots & B_d \end{bmatrix}$$

Analytical expressions for the matrix elements can be derived. The elements of S_x are:

$$A_d^2 = (\mathsf{I} + \Delta t A_c)(\mathsf{I} + \Delta t A_c) = \mathsf{I} + 2\Delta t A_c$$
$$A_d^3 = (\mathsf{I} + 2\Delta t A_c)(\mathsf{I} + \Delta t A_c) = \mathsf{I} + 3\Delta t A_c$$
$$\vdots \qquad [B.22]$$
$$A_d^j = \mathsf{I} + j\Delta t A_c \qquad [B.23]$$

In S_u, the elements are:

$$A_d^j B_d = (\mathsf{I} + j\Delta t A_c)B_d = B_d + j\Delta t \begin{bmatrix} 0 & \mathsf{I} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} (\Delta t^2/2)\mathsf{I} & 0 \\ \Delta t \mathsf{I} & 0 \\ 0 & -\Delta t\alpha \end{bmatrix}$$
$$= \begin{bmatrix} (\Delta t^2/2)\mathsf{I} & 0 \\ \Delta t \mathsf{I} & 0 \\ 0 & -\Delta t\alpha \end{bmatrix} + \begin{bmatrix} (j\Delta t^2/2)\mathsf{I} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} (j+1/2)\Delta t^2 \mathsf{I} & 0 \\ \Delta t \mathsf{I} & 0 \\ 0 & -\Delta t\alpha \end{bmatrix} \qquad [B.24]$$

In a similar manner as in the previous section, the state propagation of the system under the influence of gravity can be investigated by completing the matrix product of the discrete LTI. As this also helps to visualise the structure of the matrices, this is shown next. The full system equation – without control input – is:

$$\begin{pmatrix} r_x(k+1) \\ r_y(k+1) \\ r_z(k+1) \\ \dot{r}_x(k+1) \\ \dot{r}_y(k+1) \\ \dot{r}_z(k+1) \\ z(k+1) \end{pmatrix} = \begin{bmatrix} 1 & & & \Delta t & & & \\ & 1 & & & \Delta t & & \\ & & 1 & & & \Delta t & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix} \begin{pmatrix} r_x(k) \\ r_y(k) \\ r_z(k) \\ \dot{r}_x(k) \\ \dot{r}_y(k) \\ \dot{r}_z(k) \\ z(k) \end{pmatrix} + \cdots$$
$$\cdots \begin{bmatrix} \Delta t^2/2 & & & & & & \\ & \Delta t^2/2 & & & & & \\ & & \Delta t^2/2 & & & & \\ \Delta t & & & & & & \\ & \Delta t & & & & & \\ & & \Delta t & & & & \\ & & & \Delta t & & & -\alpha\Delta t \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ g \\ 0 \end{pmatrix} \qquad [B.25]$$

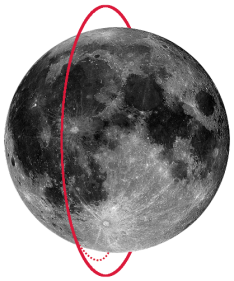
where blank elements represent zeros in these sparse matrices. For the example of r_z, the propagation from step k to $k+1$ is thus:

$$r_z(k+1) = r_z(k) + \dot{r}_z(k)\Delta t + \tfrac{1}{2} g \Delta t^2 \qquad [B.26]$$
$$\dot{r}_z(k+1) = \dot{r}_x(k) + g\Delta t \qquad [B.27]$$

Thus, as with the one dimensional case, the error made at the nodes vanishes with respect to the analytical solution.

Appendix C

GOLDEN-SECTION SEARCH ALGORITHM



This appendix gives a full listing of Brent's method, that combines the golden-section search algorithm with parabolic interpolation [Brent 2013]. For a detailed explanation of the algorithm, see Section 6.5 starting on Page 80.

❐ ALGORITHM 1 Finding the minimum of a unimodal function by parabolic interpolation and golden-section search.

```
1:  procedure GOLDEN_SECTION_SEARCH(a, b, eps, tol, J)
2:      φ ← (3 − √5)/2                                   ▷ golden section
3:      v ← w ← x ← a + φ(b − a)                         ▷ initialize
4:      e ← 0                                            ▷ size of larger half of interval
5:      m ← 0.5(a + b)                                   ▷ mid point
6:      t2 ← 2(eps · |x| + tol)                          ▷ absolute and relative tolerance
7:      fv ← fw ← fx ← f(x)                              ▷ initial trial
8:      while |x − m| > t2 − 0.5(b − a) do               ▷ main loop
9:          p ← q ← r ← 0
10:         if |e| > tol then                            ▷ fit parabola
11:             r ← (x − w)(fx − fv); q ← (x − v)(fx − fw)
12:             p ← (x − v)(fx − fw)q − (x − w)r
13:             q ← 2(q − r)
14:             if q > 0 then
15:                 p ← −p
16:             else
17:                 q ← −q
18:             end if
19:             r ← e; e ← d
20:         end if
21:         ▷ parabolic interpolation
22:         if |p| < |0.5qr| ∧ p < q(a − x) ∧ p < q(b − x) then
23:             d ← p/q
24:             u ← x + d                                ▷ next trial point
25:             ▷ do not evaluate f too close to a and b
26:             if u − a < t2 ∨ b − u < t2 then
27:                 if x < m then
28:                     d ← tol
29:                 else
30:                     d ← −tol
```

```
            end if
        end if
    else                                              ▷ golden-section step
        if x < m then
            e ← b − x
        else
            e ← a − x
        end if
        d ←ce
    end if
    if |d| ≥ tol then                  ▷ f must not be evaluated close too x
        u ← x + d
    else if d > 0 then
        u ← x + tol
    else
        u ← x − tol
    end if
    fu ← f(u)                                         ▷ evaluate current trial
    if fu ≤ fx then                                   ▷ update preserving variables
        if u < x then
            b ← x
        else
            a ← x
        end if
        v ← w; fv ← fw; w ← x; fw ← fx; x ← u; fx ← fu
    else
        if u < x then
            a ← u
        else
            b ← u
        end if
        if fu ≤ fw ∨ w = x then
            v ← w; fv ← fw; w ← u; fw ← fu
        else if fu ≤ fv ∨ v = x ∨ v = w then
            v ← u; fv ← fu
        end if
    end if
    m ← 0.5(a + b)                                    ▷ mid point
    t2 ← 2(eps · |x| + tol)                           ▷ update tolerance
end while
return x, fx                          ▷ location of minimum, minimum
end procedure
```